A NOVEL INTRUSION DETECTION SYSTEM (IDS) ARCHITECTURE

ATTACK DETECTION BASED ON SNORT FOR MULTISTAGE ATTACK SCENARIOS IN A MULTI-CORES ENVIRONMENT

JULES FERDINAND **PAGNA DISSO DE MUILA**

PhD

UNIVERSITY OF BRADFORD

2010

Abstract

Recent research has indicated that although security systems are developing, illegal intrusion to computers is on the rise. The research conducted here illustrates that improving intrusion detection and prevention methods is fundamental for improving the overall security of systems.

This research includes the design of a novel Intrusion Detection System (IDS) which identifies four levels of visibility of attacks. Two major areas of security concern were identified: speed and volume of attacks; and complexity of multistage attacks. Hence, the Multistage Intrusion Detection and Prevention System (MIDaPS) that is designed here is made of two fundamental elements: a multistage attack engine that heavily depends on attack trees and a Denial of Service Engine. MIDaPS were tested and found to improve current intrusion detection and processing performances.

After an intensive literature review, over 25 GB of data was collected on honeynets. This was then used to analyse the complexity of attacks in a series of experiments. Statistical and analytic methods were used to design the novel MIDaPS.

Key findings indicate that an attack needs to be protected at 4 different levels. Hence, MIDaPS is built with 4 levels of protection. As, recent attack vectors use legitimate actions, MIDaPS uses a novel approach of attack trees to trace the attacker’s actions. MIDaPS was tested and results suggest an improvement to current system performance by 84% whilst detecting DDOS attacks within 10 minutes.

Acknowledgement

First things first, I would like to thank God for making this project come through. I also would like to thank both my supervisors John Mellor and Andrea Cullen for their great support as I was working through this research project. Great thanks go to my direct family and in-laws who have been of a great moral, financial support. I would like to mention my wife Jaigar, my daughter Jessie and Mum and Dad, and my auntie Maguy Sardin.

To my colleagues at Syphan, I would like to say a big thank you as they have provided an environment for me to learn and grow. On the same note, I would like to thank all my friends who have contributed to the accomplishment of this work by their direct or indirect input.

I will conclude this by thanking the University of Bradford who allowed me to carry out my research despite the problems and I had over the course of this work. In advance, I would like to thank the examination board (external and internal) examiners for taking the time to examine my work.

Last but not least, a special thanks to Diana Sinclair, Anthony Fomuso, and many other who has help minimising the number of typos and grammatical error in this thesis.

To God be the Glory for His mercies endure forever.

Table of contents

Table of figures

List of tables

1 Introduction

This chapter will start by defining the terms and expressions specific to the thesis and by giving an account of the current state of Internet attacks. The thesis will then look into some of the technological advances that Intrusion Detection Systems (IDS) can benefit from i.e. multi-core processors. The motivation, research aim and objectives will be presented. The research methodologies used throughout the thesis will be explained to add meaning to the results obtained. The original contributions of this thesis will be presented. This chapter will conclude by presenting the outline of the whole thesis.

1.1 Terms and Expressions

1.1.1 Abbreviations

AMD: Advance Micro Devices

DDOS: Distributed Denial of Service

DMZ: Demilitarised Zone

IANA: Internet Assigned Numbers Authority

IDS: Intrusion Detection Systems

IPS: Intrusion Prevention

MIDaPS: Multistage Intrusion Detection and Prevention System

NIC: Network Interface Card

NIDS: Network Intrusion Detection Systems

1.1.2 Terms and expressions

Bot: a piece of software that is programmed to execute a number of predefined tasks and usually await order of execution for a master computer.

Botnet: A botnet is a group of bot infected PCs that are all controlled by the same "command and control centre". A botnet, also known as a zombie army, is a computer connected to the Internet that has been set up to forward transmissions (including spam or viruses) to other computers on the Internet, without the knowledge of the computer owner.

Countermeasure: countermeasure is a process put into place to address a vulnerability in order to reduce the probability of attacks hence reducing the possible impact of a threat.

DDOS attacks: a DDOS attack is a distributed denial of service. It is a denial of service that is performed in an orchestrated manner using multiple attackers against one victim.

Denial of Service: a denial of service is an attempt to make computer irresponsive so that they stop delivering the services they intended for to those having right of access.

Distributed Denial of Service: it is an alternative way to call DDOS attack

False negative: We speak of false negative when an alert is not generated when it was supposed to be generated. In fact the system is generally thinks it is not attack when it is.

False positives: we speak of false positive when attack did not occurs but the system generate an alert as if they was a security breach

Flood attacks: flood attacks refer to any attack that is perform against a computer system by overloading the system resources. This can be a flood of request, response, or unwanted messages. Flood attacks generally lead to Denial of Service

Denial of Service: denial of service occurs when legitimate users are prevented from accessing and using their resources. This is generally achieved through DDOS, DOS, and flood attack.

Flow: a flow in an exchange of message between two host from the SYN to the ACK after a FIN/AC as shown in Figure 1-1

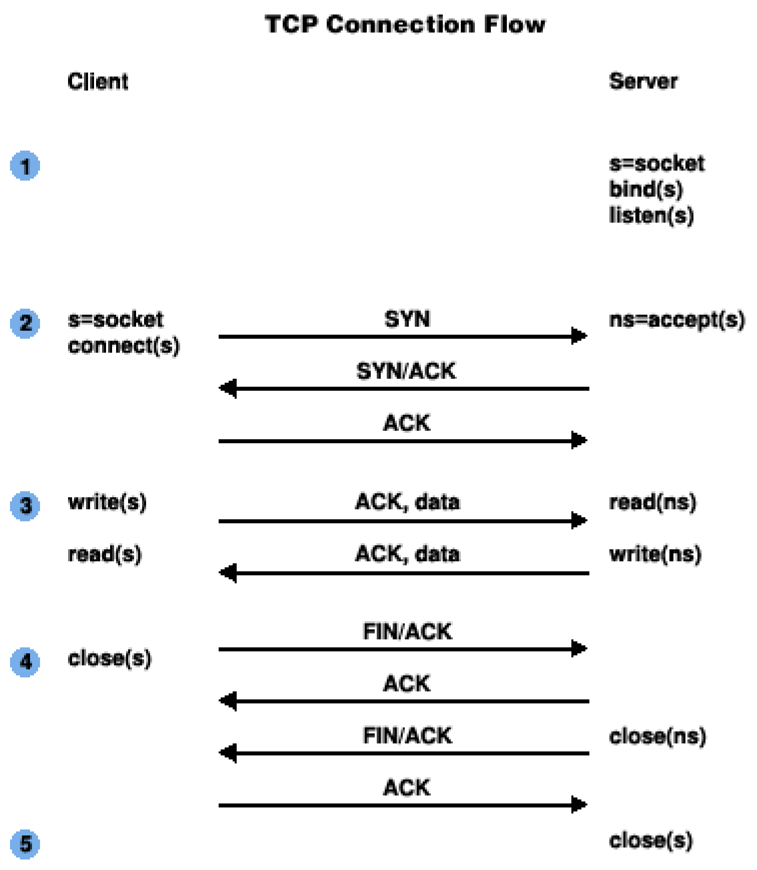


Figure 1-1: TCP Flow

Fragmentation Attack: it is also known as the overlapping fragment attack. Fragmentation refers to the IP datagram broken down into smaller packets and over the network via different types of network media. These smaller packets are generally reassembled at the receiving end. They are different type of fragmentation attacks: Ping O' Death Fragmentation Attack, The Tiny Fragment Attack, and The Teardrop Attack.

Hardware based IDS: is it an IDS that has been implemented on chip. Here the IDS is embedded into the hardware.

Hybrid IDS: a Hybrid IDS is an IDS that works both as a Network IDS and a Host IDS.

Intrusion Detection Systems: An Intrusion Detection System (IDS) is a system, software or hardware that listens to incoming and outgoing traffic and reports any evidence of attacks or policy violation. An extensive definition of IDS is given in chapter two.

Intrusion Prevention System: it is an Intrusion Detection System with the capability to react against malicious packets. The usual reaction is a simple block of the malicious packet.

Multicore : multicore will generally be used as a short form of multicore processors

multi-core processors : A multi-core processor is an integrated circuit (IC) to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks [1].

Multicore technologies : it refers to technologies that use multicore processors.

Multistage attacks: a multistage attack is an attack that is performed in multiple steps. For instance, the attack can start by the user clicking on a link from an unknown email. From the link a malicious program will then be downloaded unto the computer. The malicious program will then start to communicate with the master attacker in order to open door to attacks

Parallel Programming: It is now common to have computers i.e. laptops or desktops with two or four CPUs also called cores. Multiple cores are equivalent to multiple CPUs. To take full advantage of this evolvement, the source code can be parallelised and its execution distributed across all the available CPUs. Not too long ago, concurrent programming was only possible from a low-level manipulation of threads and locks. With the recent technology and tools like visual studio 2010, it is possible to write codes that will be executed across the different processors with very little effort. Multiple tasks are executed at the same time

Pre-processor / Preprocessor: preprocessors are blocks of code organised in a way that the block can be turned on or off. They are two major pre-processors. One adds another layer of analysis to Snort. This layer of detection is intended to do complex tasks when rules cannot be used to detect attacks. When an attack cannot be expressed into a rules (based on a signature), a pre-processor can be written for that purpose. In the other hand, pre-processor can be used to shape the traffic to make it easy for the detection engine. For instance, obfuscated URL go through a pre-processor to transform the URL into regular URL so that matching can be done by the matching engine.

Regular expressions: Regular expressions are also known as regex, or regexp. Regular expressions are expressions that are represented using a sort of compression language to shorten many expressions into one using similar groups of expression. For example, ab? Would match ab plus any single character (aba, abc, abd, ab1, abe, etc.)

Sequential Implementation: a sequential implementation is an implementation where instructions are executed one by one, one after the other one

Signature: Signature will always make reference to Intrusion Detection System signature in this thesis. In that sense, a signature is a pattern, a string that was written to match attack behaviour here identified as string in packets.

Single core application: a single core application is an application that has been writing without taking on board more than one CPU. This type of application can still be a parallel application by using threats and dead locks.

Social networking: social networking is a group of people with common interested. In this thesis, social networking will refer to group of people coming together with common interest using the Internet.

SPAMS: SPAMS refer to the use of email to send huge amount of unsolicited emails.

TCP Conversation: TCP conversations represent a complete communication between two ends during a full session. This communication can be broken into multiple packets. One conversation is sometimes refers as one session.

Threats: it is the possibility to take advantage of vulnerability and turn it into attack against the vulnerable system.

Trace file: it is a file in which activity related to a user or a program is recorded

Traffic generator: a computer program use to generate traffic with predefined conditions.

Virus: computer program written with malicious intend to harm computer system.

Vulnerabilities: A vulnerability is a flaw or weakness in system security procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the system's security policy. It is also considered as the existence of a weakness, design, or implementation error that can lead to an unexpected, undesirable event compromising the security of the computer system, network, application, or protocol involved

Worms: a worm is a malicious piece of code, software, is able to replicate itself without any human interaction and propagate itself across networks, using email, share folders etc.

Zombies: a zombie is a computer who has been affected by a malicious program and made part of a botnet.

1.2 Background

Without question, the Internet today influences almost every single aspect of life. Social networking has proved to be a great tool in bringing the world together: MySpace in 2003, Facebook in 2004, and Twitter in 2006 [2], [3] and other similar social websites have seen unprecedented growth. However, malicious users have taken the opportunity to create applications that have been used to recruit computers (also called zombies) into an army of computers controlled remotely to serve in an attack at a later stage: a Distributed Denial of Service (DDOS) attack [4] . Almost every single possible opportunity has been exploited by malicious users to gain illegal access to computers or to have control over them. Recent events revealed that these types of attack, targeted at social networking users, have been used against businesses and as a cyber-war tool [5]. Important governmental and business websites have been forced to close for hours, days or even months by attackers sending excessive amounts of data to their servers causing them to stop delivering their intended services. Not too long ago, Internet security was a concern mainly for business users. In today's world, Internet security has become a matter of National Security, forcing governments to take active part in the game [6].

DDOS attacks are performed by exhausting resources of other computers without the consent and knowledge of legitimate owners. Computers participating in a DDOS attack are affected in speed and overall performance as excessive unplanned resources are used. In order to attack others computers, the attacking computer is force to use extra resources. This put the attacking computer in the position of being victim of DDOS attack if its resource used for the attack exceeds or approach the limit of resources needed for a normal operation. DDOS, more than any other Internet attacks have raised a high level of awareness in the world. A great deal is spent by governments to try and solve the problem [7], [8]. In addition, research groups have had sponsorship to dedicate more time researching ways and methods to stop malicious users and their activities. A well know open source project, Emerging Threats, has benefited from one of these grants. They have been carrying out research on how best to secure computer systems using Snort IDS [9]. Our research work is based around Snort. There is a great need to secure online activities as demonstrated by the multiple actions and grants towards making the Internet more secure [10].

IDS have been increasingly used by communities around the world to detect and protect against attacks as they are free of charge and involve a wide community of experts. However the increased speed of communications, the pace at which attacks are performed and the sophistication of recent attacks make detecting and mitigating them very difficult. Despite their popularity, IDS have failed to offer a level of security that would protect against recent attacks as they grow in speed, bandwidth and sophistication, hence a move towards multi-core systems [11], [12]. As shown in Figure 1-2 there has been a growing interest in multi-core technologies and a slight decrease in specialised processors. From 2003 to 2007 there has been an increased interest in finding an alternative to single core. This indicates that researchers have found that converting existing single core applications is not efficient [13]. As shown in Figure 1-2 research efforts in single core applications have been significantly reduced to the benefit of research in other field such as multicore. This is an indication that building an IDS that is capable of multicore is more relevant that working on a single core.

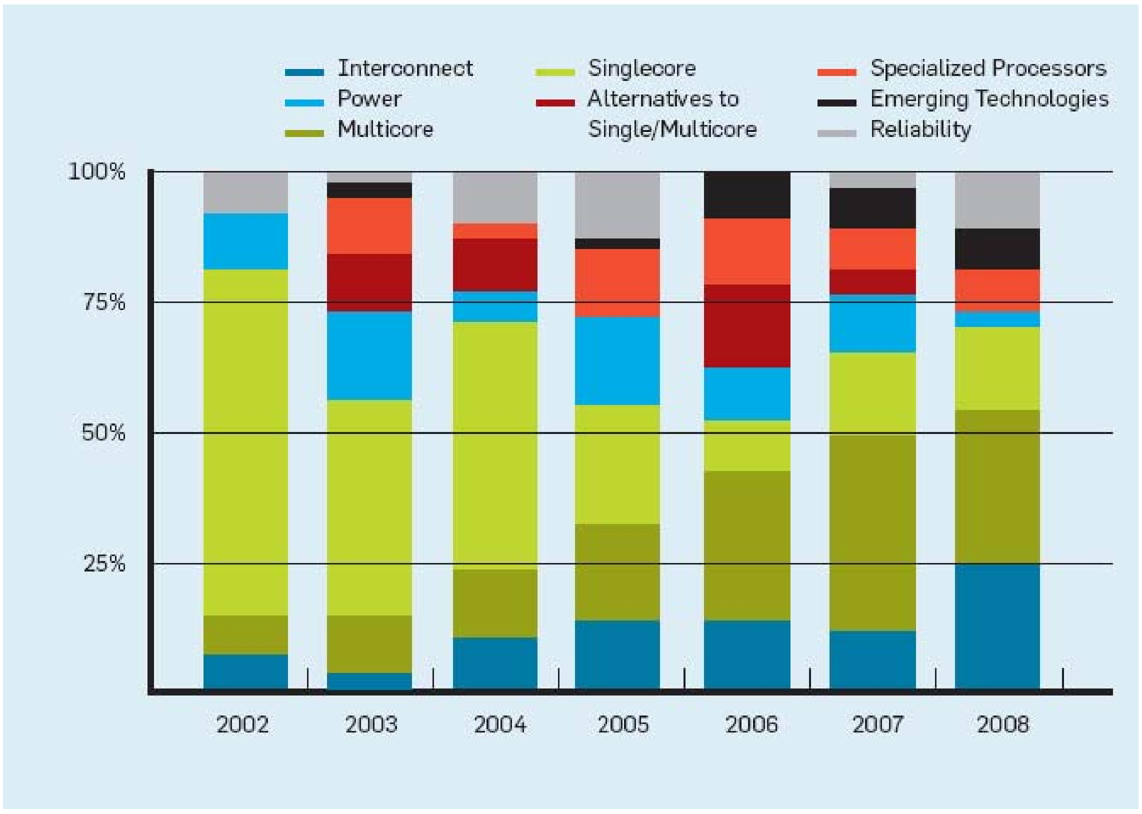


Figure 1-2: Evolution of research areas [13]

Many studies around improving IDS have led to scrutinising how multi-core technologies would benefit the implementation of such systems [14]. As preliminary research results around this topic show, there is great potential in performance improvement by using such systems but no system to date has been successfully designed and implemented [10] . In addition, successful implementations of network based applications around multi-cores technologies need to be examined thoroughly. Some researchers argue that

> "*Parallelizing legacy code is widely viewed as a deadend, but building compelling addons to existing applications that take advantage of multicore, and then "bolting on" these features to legacy codes is possible"* [13]

A successful implementation of IDS would be able to keep up with the technological advancements and the level of sophistication of attacks. Lately using multi-core processors to improve speed and performances of systems has become a popular subject and has drawn a lot of attention [13]. There are many benefits from using multi-core processors that have yet to be exploited. However, producing a good parallel system from a traditional serial implementation remains a challenge as well as redesigning existing systems so that they could be compliant with multi-core system of today and in the future. There have been a few attempts [15], [16], [14] but these researches generally look at a particular aspect inside the very complex structure of IDS without looking at the consequences that their partial solution could have on the whole structure of IDS systems or they use parallelism to prove that systems can run faster but with no indication of the improvement in attack detection. Other researchers have looked at improving current IDS systems based on a particular hardware that does not necessarily comply with the evolution of processors as stated by Moore's Law [17]. Others still have looked at parallel IDS that do not address recent attacks. As a consequence such systems would need to be readapted for new hardware requirements. In this research, the author focuses first on identifying why IDS systems fail against the latest attacks by looking at IDS components. The author then proposes a new IDS architecture that is compliant with the current technological evolvement and that would not need to be redesigned based on any particular hardware. The architecture is targeted at using available hardware on the market without any prior change to the hardware structure. In addition, the architecture of the new IDS will partly focus on DDOS detection and mitigation as DDOS has been identified as compulsory feature for a security system protection.

1.3 Motivations

The motivation of this research came from the industrial challenges of producing a security system that would ensure availability, confidentiality and integrity in regards of the current state of Internet security. After several years of industrial experience and while working on the design of a security system for a 10GB appliance, some interesting ideas raised few questions marks. A recent move by innovators to hardware based IDS implies a high speed and millions of packets processed at the same time. However, hardware based solutions have limitations in their capabilities to execute particular software functions. For instance, there is no regular expression system fully implemented in hardware. Currently, Snort rules contain 65% of rules with regular expression. In addition, hardware based IDS have a lot of memory problems as there is hardly any dynamic memory allocation. While the hardware improves speed, the software implementation of IDS has more features even though there are currently many attacks that go undetected. Securing Internet systems is more and more challenging. In addition, a good number of companies rely on IDS and IPS to protect their systems [18].

After investigation, it appears that there is a gap to fill. Hardware based IDS deliver speed but are very expensive; software solutions, even though they do not offer the conviviality of speed offered by the hardware IDS, they are able to support a wider range of functionalities. From these conclusions, the author started working on a system that will not only increase the packet processing rate but also offer deeper analysis in order to enable the maximum detection and mitigation possible. This will be achieved by adding additional analysis capabilities to what currently exist in Snort. Further discussions in the coming chapters will go into details how this can be achieved.

On the other hand, the author has a strong interest in contributing to the open source as it is an environment where experts meet, discuss and work together. In that regard, some of our work has been published to the open source community managed by Google [19] where the author has produced an HTTP code generator intended to test IDS resistance against spam sources.

This work was started from the perspective of having a system that works i.e. a system that is able to cope with recent attacks and technology advancement, a system that will deliver granularity in analysis, a system that produces less false positive while improving the detection rate.

1.4 Research Aim and objectives

This research aims at producing a new architecture design for Network Intrusion Detection Systems that will not only take full advantage of multi-core processors as they continue to evolve but moreover, an architecture that is able to stand against the most dangerous attack faced by the Internet today i.e. Denial of Service attacks (DDOS) and multistage attacks i.e. attacks distributed amongst packets and flow. The new architecture aims to be a multi-dimensional parallel framework that will use readily available components i.e. widely available software and hardware.

In striving to produce such a system, the following objectives would be achieved:

- Redesign IDS based on Snort using a multi-dimensional approach and ready for multicores architecture
- Identify Snort specific engines component weaknesses
- Produce a system that will improve the detection rate whilst reducing the false positive
- Design a multistage attack detection system
- Keep up to date with technological advancement by suggesting a parallel implementation of our architecture.
- Extend rules format to enable flow tracking
- Contribute to the state of the art of Internet Security

1.5 Research Methodologies

Different methodological approaches have been used, but mainly an iterative experimental approach has been followed. In this an idea has been developed and then tested, the results of experimentation have been considered and used to refine the idea or lead to the development of new approaches. Critical evaluation of the stages and then ultimately of the overall system confirm its novelty and expose areas for improvement.

1.6 Identifying the state of the art

A number of ongoing research studies have been identified by using the literature review. The literature review which follows in the next chapter was used as the starting point of our critical analysis which helped us to identify the limits of current evolvement of Intrusion Detection Systems. Our literature review was mainly based, but not limited to researches published in:

- Institute of Electrical and Electronics Engineers (IEEE) which is the world's leading professional association for the advancement of technology;
- Science Direct which is one of the most extensive sites that provides online access to scientific and technical access.
- Association for Computing Machinery, the world’s largest educational and scientific computing society which delivers resources that advance computing as a science and a profession
- Google scholar and windows live academic which is a central point of researching through various online academic databases
- Vern Paxson webpage. Vern is Associate Professor of Computer Science at the University of California, Berkeley. His main active research areas are Bro and CCIED (the NSF-sponsored Collaborative Center for Internet Epidemiology and Defenses, a joint effort with UC San Diego). The main topics of CCIED are botnets and Internet worms, including their network telescope project which is also what they are interested in. He has been awarded the Association for Computing Machinery’s Grace Murray Hopper Award for his work in measuring and characterising the Internet [20]. On various occasions, there have been email exchanges with Vern to discuss some points related to his publications. He is the founder of the second most popular IDS: BRO IDS.

- The Intel® Parallel Programming and Multi-Core Community: The Intel® Parallel Programming and Multi-Core Community have been very helpful in providing good technical information on how to go about programming for multi-core processors. Also, this thesis has gained from the later community by learning how to avoid common errors and how to optimise the code written to improve IDS
- OPENMP
- Microsoft Parallel Pages

1.7 Dataset Sources

Most of the experiments in this thesis have been performed offline. Data used in our experiments have been collected in various ways and from different sources to test the limit of Snort and to validate our model and architecture.

1.7.1 Evilfingers

Evilfingers is a community portal for information security. They make packets i.e. recorded traffic from real events available that can be used for various purposes. Some of the packets are tagged with a Snort signature and others are not.

EVILFINGERS data were merged with the 2009 Inter-Service Academy Cyber Defense Competition data to form the main source of data used in this research. They organise their files in such a way that one file/capture target one particular signature in Snort when a signature has already being identified. Using Linux command line, all the files were merged together so that all the attacks would run at the same time

```
Mergecap –w  *.pcap allpcap.pcap
```

1.7.2 PCAPR

PCAPR organises data by categories. Similarly to EvilFingers, the files are organised in a way that each file will target one specific objective. Some of the files captured important events worth analysing without necessarily being attacks. In our research, this thesis has been mainly interested in files that were capture from attacks.

1.7.3 The Metasploit Project

The Metasploit project is collection of proof of concepts that have been packaged in order to help in penetration testing, IDS signature development, and exploit research. It has been used in many other researches [21-24]. In this research, Metasploit framework was used to generate attack traffic. Every time a signature was triggered, the packets that triggered the signature were isolated from the full packets capture file. At a later stage, all the files that were collected after signatures were triggered were joined to form a bigger file used test Snort resistance against recent attacks. Metasploit was used in conjunction with Fragroute and Fragrouter.

1.7.4 Fragroute and Fragrouter

Fragroute and Fragrouter are two very similar programs that are used for attack such as “Man in the middle attack”. They offer extensive fragmentation capabilities.

1.7.5 FX-HTTP-TRAFFIC-GENERATOR

Fx-HTTP-Traffic generator is a program written during the course of this research. It has been used to test IDS against their resistance to a well know database of SPAM source: the URL blacklist service

1.7.6 URL blacklist service

This is a commercial URL Blacklist service. They are over 3165041 URL and domain entries. The database is updated regularly and provides an efficient way to block bad URLs and to write security policies. In the context of this research, the URL blacklist service was used to generate HTTP traffic to test IDS against their awareness and resistance for spam.

1.7.7 SSL Black List

Snort by default does not analyse encrypted traffic. Yet there are simple steps that can be taken to offer a level of security against bad SSL traffic. The work carried out in this research will use this list for the latter purpose.

1.7.8 "2009 Inter-Service Academy Cyber Defence Competition" [25]

This dataset was used in lieu of the DARPA dataset that have been long used as the standard dataset for IDS. This dataset is the result of the Military Cyber Defense Exercise between the National Security Agency (NSA) and all of the different service academies. Many efforts were made to make this dataset as accurate as possible by using state of the art attack tools such as Nessus, WebScarab and Nikto while a skilled team of 30 people generated background noise traffic by interaction each with three virtual stations. This interaction included activities listed as, but not limited to, browsing the web, sending emails, downloading and uploading files, and chat. The experimental testbed is represented in Figure 1-3

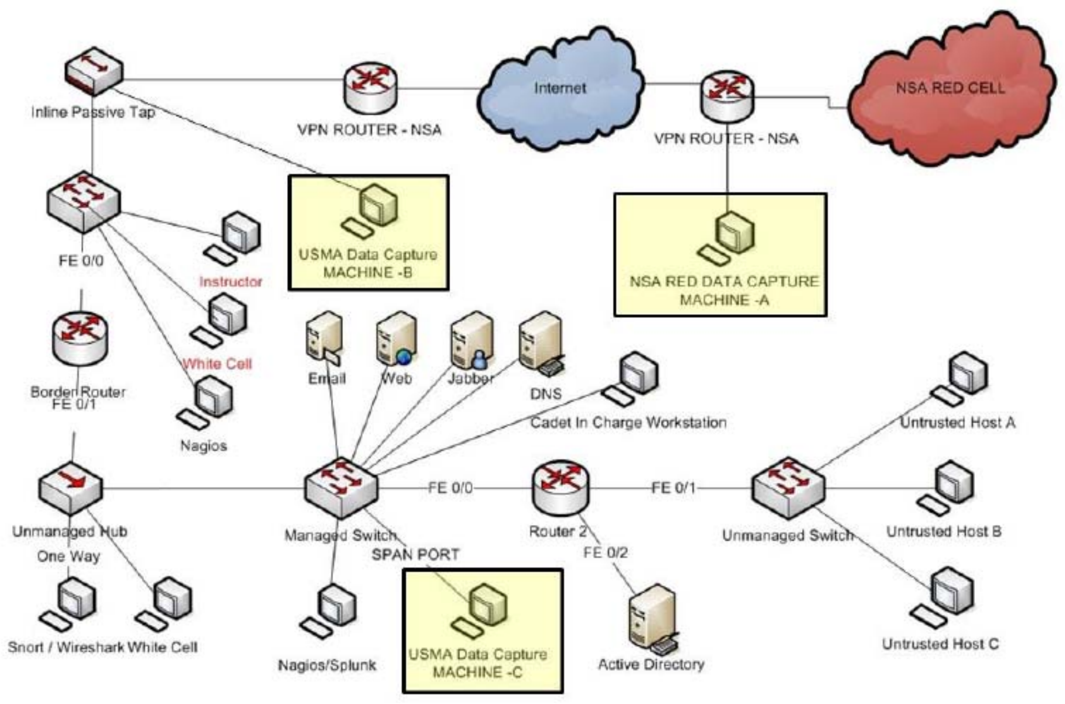


Figure 1-3: Experiment testbed for CDX 2009

1.8 Thesis original contribution

The contributions the author has made are as follows

1. The author has designed a new IDS architecture that improves the overall performance of such systems. The core elements of our new architectures are:
    a. The multistage intrusion detection and prevention systems which is an hybrid intrusion detection system i.e. a mixture of network Intrusion Detection System and Host based Intrusion Detection System
    b. An extension to Snort rules to enhance the detection engine for all patterns relevant to protect their system. This research led to an increase of 84% on Snort performance.

   c. A DDOS detection and mitigation engine that leverages an exceptional level of mitigation and detection and is able to detect attacks within twelve minutes
   d. A four level framework of visibility of attacks
2. The IDS is designed following a uniquely extensive evaluation of current security threats including thorough experimentation with real threat scenario and data.

1.9 Outlines the rest of Thesis

Chapter 2 is about the literature related to the subject explored in this thesis. Various subjects covered in this chapter are the background and state of the art for IDS, Evolution of Intel cores, Botnets, Fragmentation attacks, parallel programming, multi-core implementations and their related problems, high speed networks.

In Chapter 3 the thesis focuses on performance analysis. It starts by stress testing Snort. The thesis then moves on to test Snort against known vulnerabilities. The tests aim at showing how well Snort resists attacks within high speed networks and also how Snort protects against recent attacks. Further tests will be performed to identify what component inside Snort is failing during the tests and the reasons why those components are failing. This chapter helps define the essence of the architecture that will be proposed as the original contribution of this research. The nature of components inside Snort that are failing and the reasons why they are failing would be vital information required to build the new architecture.

Chapter 4 will present the novel multistage detection methodology that the author has designed in order to reduce the false positive but to detect more attacks and have a complete knowledge that will enable us to give more meaning to the individual alerts.

Chapter 5: The proposed DDOS engine. This chapter will present the DDOS engine as a complete unit that will later be integrated with the whole system.

Chapter 6: Our overall architecture. In this chapter, the author presents the different parts of our research as a whole.

Chapter 7 will conclude with a description of the work achieved and make recommendations for further research by addressing the extension to our work to a multi-core architecture.

After the conclusion, the references will be presented.

Conclusion

This chapter has discussed the background information related to developing a new IDS architecture that will take full advantage of multi-cores processors. The research aim and objectives have been presented. In addition to introducing the novel architecture, the research methodologies that have helped achieving our objectives were discussed. This chapter ends by presenting an overview of what will be discussed throughout the course of this research.

2 Literature review

2.1 Intrusion detection System

The work described in this thesis is based on Intrusion Detection Systems. An Intrusion Detection System (IDS) is a system, software or hardware that listens to incoming and outgoing traffic and reports any evidence of attacks [26], [27]. There are three major types of Intrusion Detection Systems: host based, Network based IDS and hybrid IDS. The work conducted here will be focusing on network based IDS. The author will be looking at Snort and Bro IDS. Snort is a modern network security application that can be used to monitor, save and report incidents as they happen on the network [28]. Bro is also a Network IDS which differs from Snort in that Snort has based its architecture around static keyword matching whereas the Bro architecture is based on events and algorithms [29].

Intrusion Detection Systems can be organised either by the type of detection they perform or by where they sit on the network.

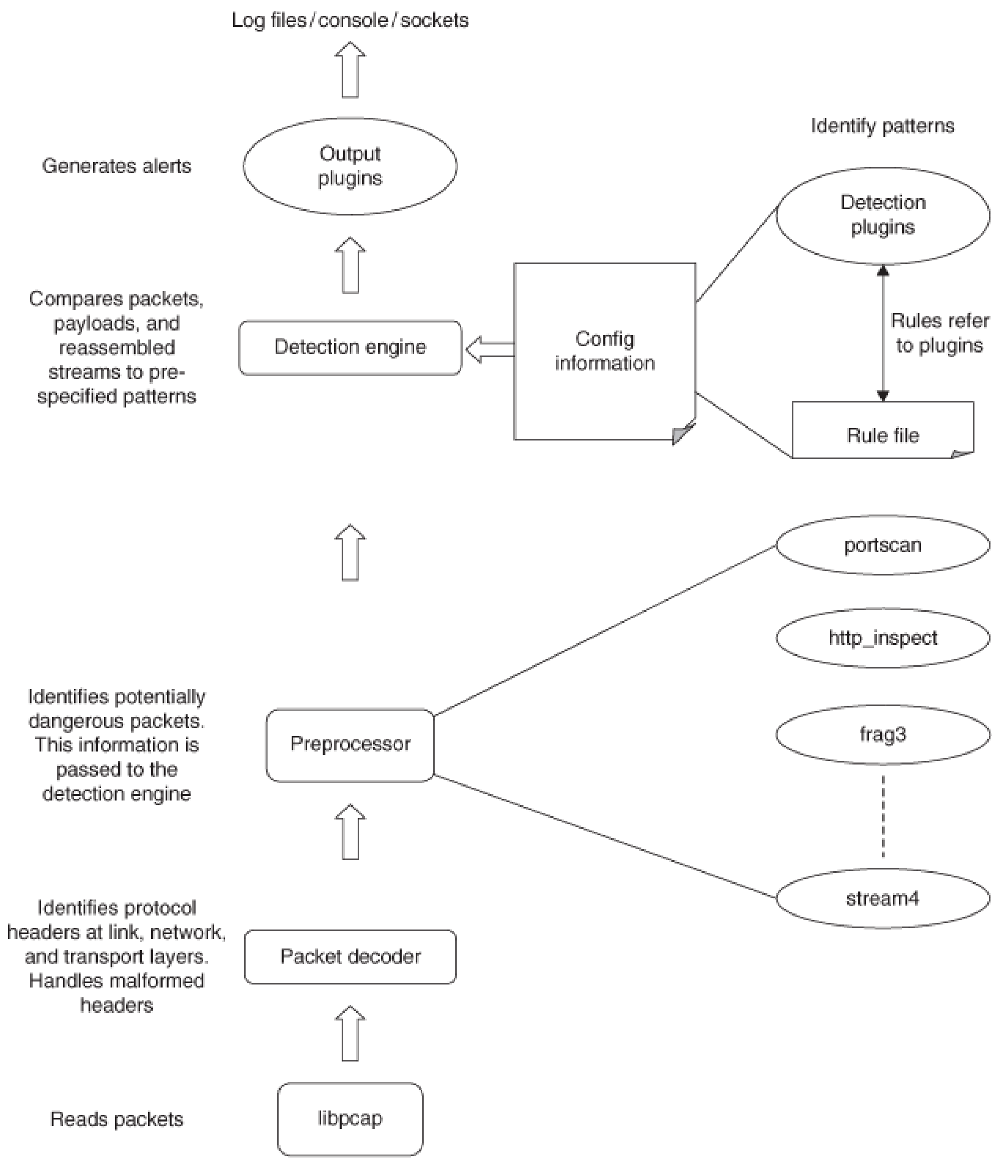
2.2 Signature based IDS – the case of SNORT

Signature based IDSs perform intensive string comparison. Keywords used in signatures are generally based on software vulnerabilities or on packet capture of a suspicious behaviour, or on packet capture of a successful attack. The live traffic, incoming and outgoing traffic is compared to a database of phrases that have been previously used by an attack or phrases that can be used by attackers based on vulnerability. The problem of signature based IDS is that they rely in most cases on the fact that the attack has to happen at least once. After analysis of the packets captured from attacks keywords are extracted from the attack to make the signature. However, this approach could give enough time for hackers to perform malicious activities between the time the vulnerability is discovered and the time an appropriate signature is published [30]. In order to defeat attackers honeypots are generally used to identify the techniques used to perform attacks [31], [32]. Honeypots can be set to auto-generate signatures based on the packets they have captured. Alternatively, signatures would be manually written. In their work, [33] suggest a system to auto generate signatures based on honeypot captures.

The use of regular expressions in IDS signatures have improved the potential that signature matching systems offer as one regular expression can contain numerous variations of an attack. However, with recent advancement of network speed and a shift to hardware based IDS and IPS, improving keyword identification remains a very engaging topic of research [34-37].

Snort has been used by millions of users and is the de facto standard of IDS. Snort is built around keyword matching and its architecture is as follows.

As shown in Figure 2-1 the packet capture handles packets at the NIC level. Snort uses an external library to capture packets. Snort uses WinPcap Under Windows OS and Libpcap under unix systems. Once the packets have been captured, they are sent to the packet decoder that will identify the different parts of the packet headers; at this stage, the malformed packets can be dropped depending on the configuration in the configuration file: snort.config. The pre-processor will do a preliminary analysis of packets to detect any potential packets. Again, the detection options depend on the settings made by a user. One would chose to ignore scans and another user will chose to be alerted on every possible alert. The packets are then passed to the detection engine that will look into the packet headers and packet payloads in order to detect any possible trace of attacks. Once the packets are analysed and depending on the results of the analysis, an alert will be generated. Various alerts system can be added to Snort by the means of plugins. Snort processing schema is represented in Figure 2-1



Log files / console / sockets
Generates alerts
Output plugins
Identify patterns
Detection plugins
Compares packets, payloads, and reassembled streams to pre-specified patterns
Detection engine
Config information
Rules refer to plugins
Rule file
portscan
http_inspect
frag3
stream4
Identifies potentially dangerous packets. This information is passed to the detection engine
Preprocessor
Identifies protocol headers at link, network, and transport layers. Handles malformed headers
Packet decoder
Reads packets
libpcap

Figure 2-1 Snort Processing Schema

The author introduces more layers of security and suggests a parallel implementation rather than a sequential implementation as is the case for Snort. For example, Snort is not configured by default to prevent IANA reserved addresses to appear in the traffic. This is justified by the fact that Snort generally sits inside the network. However, when Snort sits at the network border, there is no security feature in place to control and prevent IANA reserved addresses usage. Also, with the recent advancement in activities aiming at fighting BOTNETs, up-to-date IPs that have been found to participate in bot activities are available in a list format. This list will be used as part of the first line of defence. This will reduce the load of the detection engine. Also, during the routine rules analysis, Snort engines do not verify whether a rule is relevant to the system being protected. If Snort was to classify and check only rules that are relevant to the system in which it runs, the time spent to perform string matching will be reduced by up to 84%. The author will demonstrate this at a later stage through some experiments.

2.3 Anomaly based IDS – the case of BRO

Anomay IDS analyze every byte of traffic without in advance necessarily expecting a specific attack [38]. However, when attacks are already known, security features will be put in place accordingly. The anomaly based IDS needs a certain knowledge of the system being protected. During the learning period, the anomaly IDS will gather enough information to form the baselines, the normal behavior. In addition to deep protocol analysis, a border line is then defined as the normal behavior. When a network activity is detected and not mapped to the normal behavior action is taken. This can sometimes generate false alert indicating the false presence of attacks activities: false postive. In contrast to Snort, Bro offers a complex detection mainly based on anomalies rather than keywords. Keyword detections have their advantages that to some extent the number of false positives does not change when a network behaviour changes. As for anomaly based IDS, the detection mechanism needs to go through a perpetual learning curve . Depending on the activities of the network or of the time of year, or even the period of time during the day, activities can vary significantly [39]. There is a need for well written algorithms that will adapt to the changes without generating too many false positives.

One the biggest problems that anomaly based IDS faces is detecting attacks that fall into the category of normal behavior. The directory transversal is a web attack that does not breach any protocol definition or specification. This attack can easly go undetected by anomaly based system as it is performed under normal behaviour [40], [41].

In opposition to Snort, Bro was not built with the intention of being a system ready out of the box. Rather, it was built for research purposes in the field of IDS and traffic analysis(Paxson, 1999). Bro is built on events and its architecture is as follows:

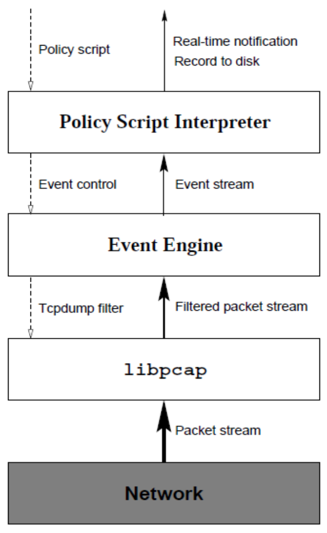


Figure 2-2: Bro Architecture [42]

As shown in Figure 2-2, Bro is built for real-time network analysis. Fundamentally, Bro provides a real-time network. At the Bottom of its architecture, Bro listens to network communication passively and sends a copy of the network traffic as it been captured to the libpcap library that will parse the traffic. Once the traffic has been organised, it is then sent to the event engine that checks the packet integrity. Once packets have been certified as valid, a hash key is created based on the flow information if not already in existance [42]. The Event engine will then generate events based on the analysis done. These events are then reviewed by the policy script interpreter. The appropriate action is taken from the policy script interpreter. These actions vary and could be as simple as logging an alert, sending an alert to an external system such as syslog, or blocking the packets.

Research around Bro IDS has evolved and a new architecture for multicore processors has been suggested. This architecture is discuss in more details in 2.9 below. One of the strengths of BRO IDS is that every single packet and flow is analysed and has to get a go ahead before it is released [43]. By scrutenising every single flow, the problem of packet fragmentation is very well addressed as BRO IDS ensures that every single byte is analysed. However, with the advancement of recent attacks, analysing flow independently is not proven to be enough to detect multistage attacks. In the architecture designed in this work, the author has introduced some flow management in order to corolate information between flows.

It is virtually impossible to have a system that will ensure 100% detection rate as well as 0% false positives. Combining the anomaly and signature based intrusion detection system has proved to be much better. Bro integrates a signature matching engine as well as maintaining an anomaly network system analysis. The system proposed in a later stage of this research will integrate both anomaly and signature to combine the power of signature for known attacks and to detect unknown attacks – the zero day attack. This is generally achieved by creating a baseline based on a “normal” behaviour that has been recorded during a learning curve. Based on a history, a behaviour profile is then defined. Anything that falls out of that behaviour would be considered as anomaly.

2.4 Intruvert Network

Intuvert Network was created after a series of Denial of Services (DoS) hit Yahoo and CNN and other websites in 2000. The objectives that Parveen Jain and Ramest Gupta, creator of Intruvert Network, had was to provide a novel approach that would provide a reliable protection to fight a wide range of Network security problems. The product from Intruvert Network was then called IntruShield. IntruShield performs a deep packet inspection on every packet that crosses the network [44]. IntruShield claim to deliver cost-effective appliances offering high-performance and reliability for various segments of network independently of their location on the network. IntruShield is relatively simple to use and to set up; a web based interface has been provided for its management. IntruShield offers a reasonable performance over network with a bandwidth up to 10 Gb. A number of security problems are addressed by IntruShield such as zero-day, DoS, DDoS, SYN flood, and encrypted attacks, and threats like spyware, VoIP vulnerabilities, botnets, malware, worms, Trojans, phishing, and peer-to-peer tunneling. IntruShield detection system is based on signature, shell-code detection algorithms, DDOS detection and prevention [45] [46]. IntruShield is able to parse about 100 protocols with over 3,000 signatures.

2.4.1 Intruvert architecture

IntruShield architecture as represented below Figure 2-3

One of the key implementation of the deep packet analysis used by IntruShield is based on packet reassembly. Packet reassembly could be problematic in high speed network. The next section will address issue relating to Intruvert Security performances.

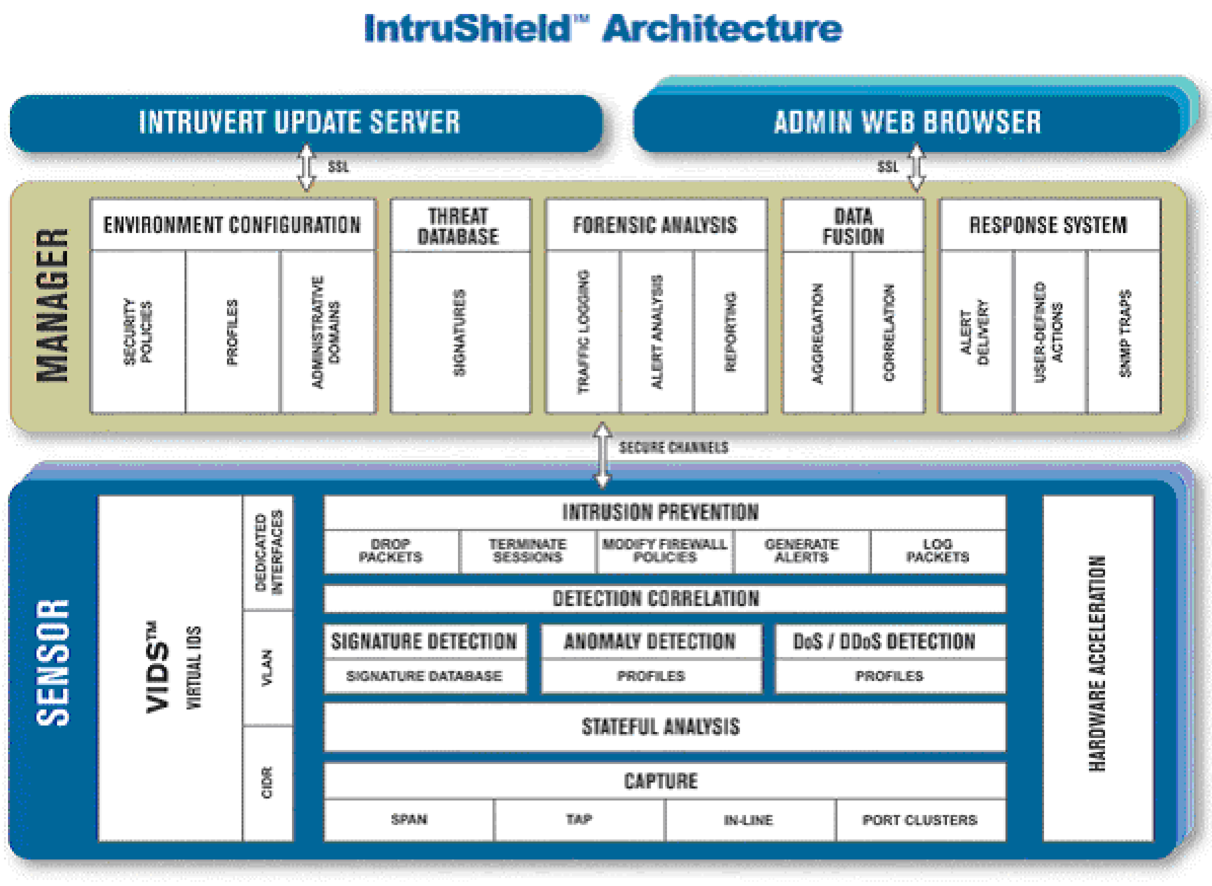


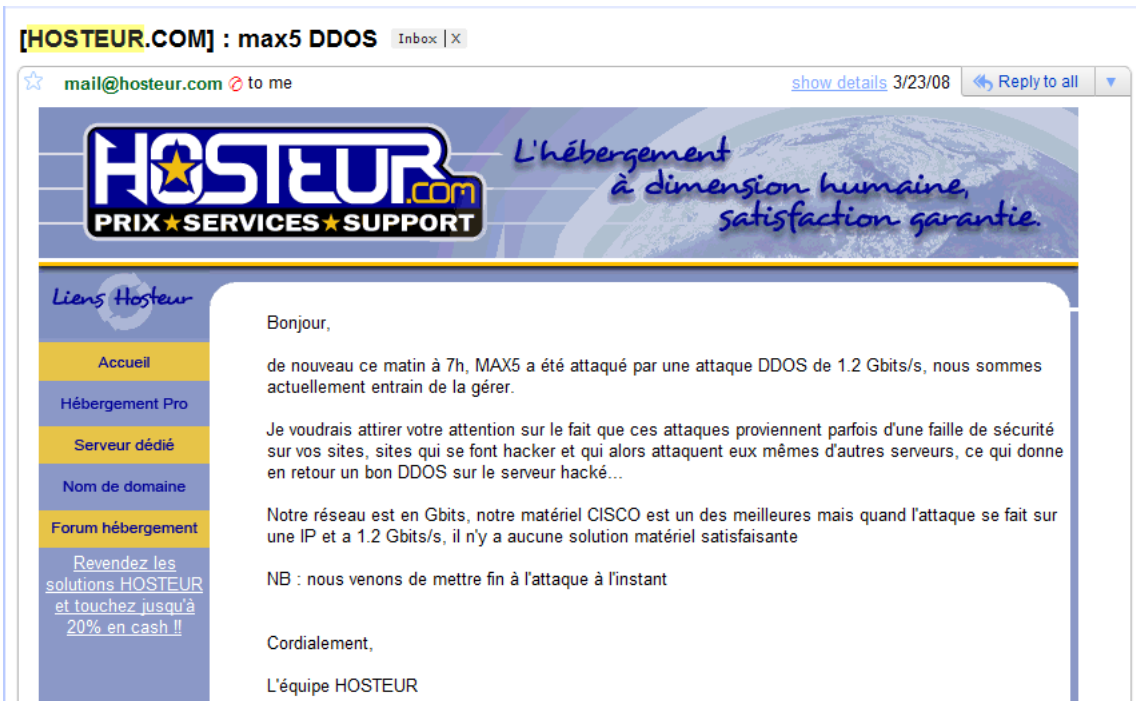
Figure 2-3: Intruvert Architecture [45]

2.4.2 Intruvert Security limits and problems

Intruvert clearly display a good range of security features Figure 2-3. However, Intruvert was not the choice of this research as it was not possible to have access to the source code for a deeper analysis of the performance of each of its components and suggest an improvement. For instance, one of the drawbacks in Intruvert is that the signatures are not available to be changed. The IntruShield appliances are based on a custom hardware platform yet. This research aims at looking at systems that are widely available and not restricted by a certain platform. Working on a hardware specific platform would mean that anyone willing to use the results produced in this research will be forced to have the same platform. While they use standard Intel processors for general management, they include network processors, ASICs and FPGAs to speed-up computing intensive tasks (e.g., signature matching and SSL-decoding) [47]. Unfortunately, McAfee does not provide concrete details about the system's internals. From its architecture, it is not clear or rather non-existent the way IntruShield will address multi-stage attacks.

In this thesis, Intruvert was not physically tested as the author could not afford to acquire it as it is a commercial product and expensive. However, In the recent attack (Figure 2-4) that Hosteur [a French web hosting company] has experimented, Cisco IDS and IPS security system was subject to a live test and failed to prove its efficiency. The French company ended up blaming a client who was running a game, Warez, on his web site. The French company has not revealed the exact model of Cisco equipment they are using but they claim to use one of the latest Cisco security device.

Intruvert is based on packet reassembly which would pose a number of issues related to the performance. At 5GB, there is literally no time for packet reassembly. The model that we propose later in this thesis will give a possible solution to the problem of reassembly.


[HOSTEUR.COM] : max5 DDOS

mail@hosteur.com to me 3/23/08

Bonjour,

de nouveau ce matin à 7h, MAX5 a été attaqué par une attaque DDOS de 1.2 Gbits/s, nous sommes actuellement entrain de la gérer.

Je voudrais attirer votre attention sur le fait que ces attaques proviennent parfois d'une faille de sécurité sur vos sites, sites qui se font hacker et qui alors attaquent eux mêmes d'autres serveurs, ce qui donne en retour un bon DDOS sur le serveur hacké...

Notre réseau est en Gbits, notre matériel CISCO est un des meilleures mais quand l'attaque se fait sur une IP et a 1.2 Gbits/s, il n'y a aucune solution matériel satisfaisante

NB : nous venons de mettre fin à l'attaque à l'instant

Cordialement,

L'équipe HOSTEUR

Figure 2-4: attack reported by hosteur

2.5 GRIDS - a graph based intrusion detection system for large networks

The concept presented by GRIDS is very interesting as it moves away from the traditional detection methodology in that the detection and the reporting are both based on a grap. One the great things of GRIDS is that it builds the network architecture in which it is installed. However, GRIDS only runs on Unix hosts connected by IP nets [48]. Also, this system assumes that the networks belongs to single organisation which have autonomous departments. However, departments in reality have many interdependences and generally share resources. It is difficult to picture how this system would work in a modern enterprise environment. Last but not least, this system assumes that no part of the network is actively hostile. The author did not understand why an IDS would would be designed to work in a non hostile environment. The paper that presents this modele [48] was purely based on principle and no experiments was done. No other work related to this IDS was identified. This system was not consider as important in this research as the author belived that the design wass not mature enough for further consideration.

2.6 Sguil: The Analyst Console for Network Security Monitoring

Sguil was writing a tcl/tk programming language [49]. This limits Sguil to linux like systems. However, using a Unix like integrated environment for Windows platform like Cygwin, Sguil is able to run on windows platform. Howeer, there there will be communcation between the Unix like integrated environment and the host Windows operating system. Sguil is an engine that is based on many other tools to perform collection, analysis, and escalation of indications and warnings to detect and respond to intrusions [50]. Sguil is based on Snort and Snort rules to perform the detection. One of the differences between Snort and Sguil is the graphical interface that Sguil offers. the later is very user friendly and make the analysis easier. The auther did not feel this was the tool to consider as it pure a management of many tools put together. Hence this system was not considered for this thesis.

2.7 Intrusion detection System and their current level of protection

Reports on the Internet show that the number of attacks is still very high [51-54] and continue to rise (Figure 2-5). Despite great efforts, secure transactions and communications over Internet security is not guaranteed. Being intrigued by the current state of Internet security and despite the many efforts accomplished in making the Internet more secure, a decision was made to investigate why intruders and malicious Internet users are still able to bypass or to bring security systems down. Figure 2-5 shows that in 2008, Symantec had created over 60% of their entire malicious signatures database to date. There has been an increase of about 150% in malicious activities [55].

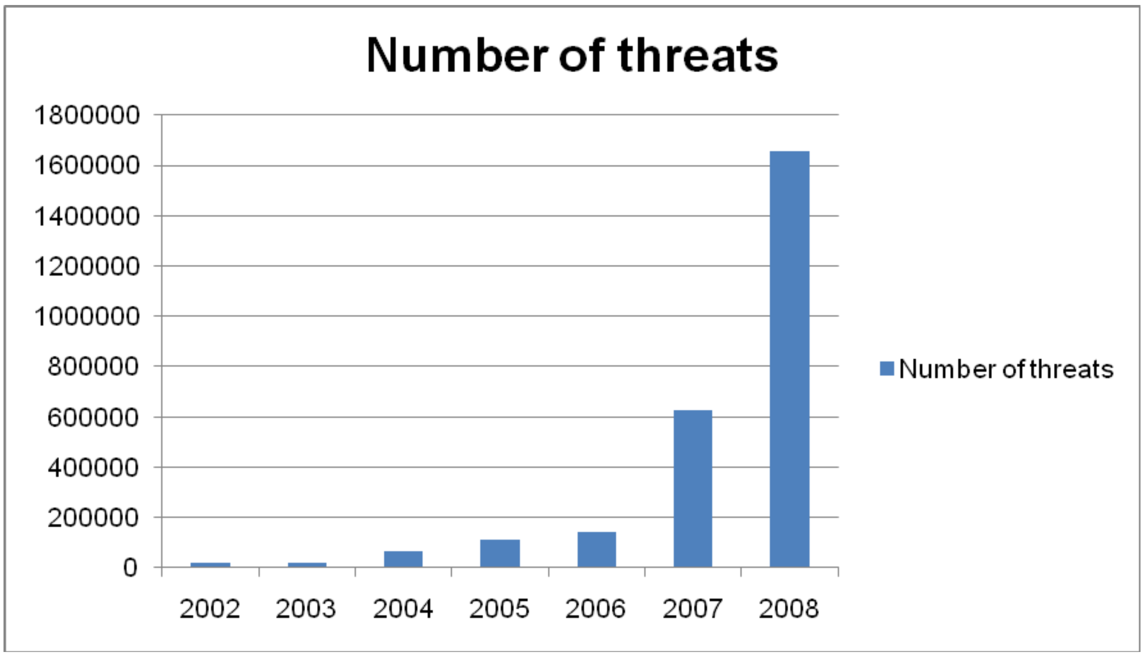


Figure 2-5: New malicious code signatures [55]

Our efforts started with a quick review of the current state of the Internet regarding network IDS. Statistics reveals that Flood-Based attacks are the most common vector attacks. Flood-Based attacks are aimed at overflowing the network resources so that targeted systems become unavailable. They are also known as Denial of Service Attacks (DOS) or Distributed Denial of Service Attacks (DDOS).

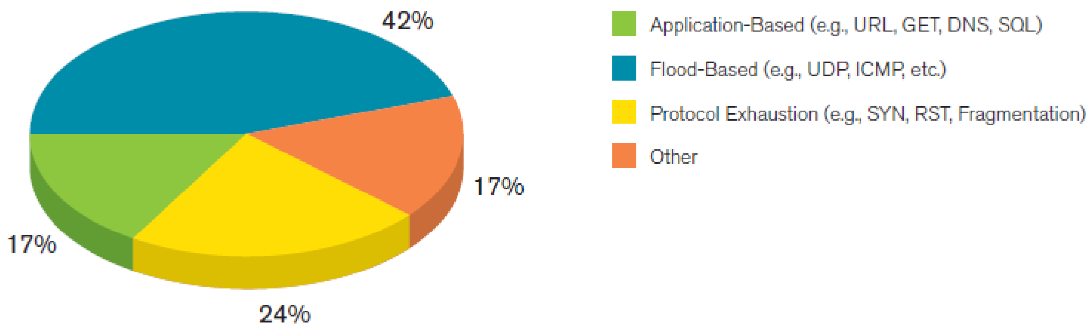


Figure 2-6: Attack Vectors [56]

Looking at Figure 2-7, during the year 2008 there was a serious increase in the speed at which attacks are performed. This implies that security systems have to be able to perform at high speed. One of the biggest consequences of not being able to perform at this pace would be that attacks will not be reported, actions will not be taken hence the system protected will crash. In fact, most traffic will go without being analysed causing many attacks not to be detected.

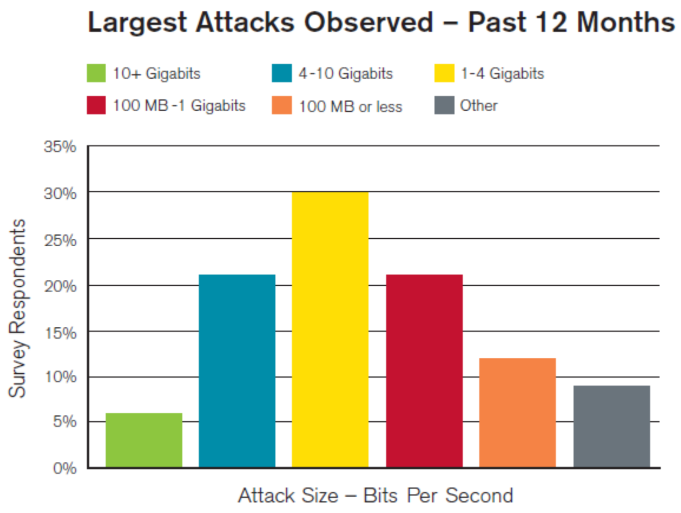


Figure 2-7: Attacks based on speed ([92])

Figure 2-7 confirms that the most serious threats are based on botnets which are networks of infected computers ready to execute commands from a bot master, the commanding computer. One of the largest botnets to date is evaluated at

1,5millions computers (Sanders, 2005). In regards to internet security, it is possible to flood almost any network from such a powerful botnet by only sending 1Mbs/host. Sending 1Mb/s per host or bot would mean sending 1.5 million Mb/s. Currently, there are very few systems that support 10Gb/s i.e. 10,000Mb/s. This shows that botnets can easily flood networks. Traditionally, botnets have been used to send spam i.e. up to 3 billion spams per day [57]. Lately, they have been used not only to send spam but to install malware, Trojans, delete data from computers and flood networks. Botnets have also been recently used in cyber war [58]

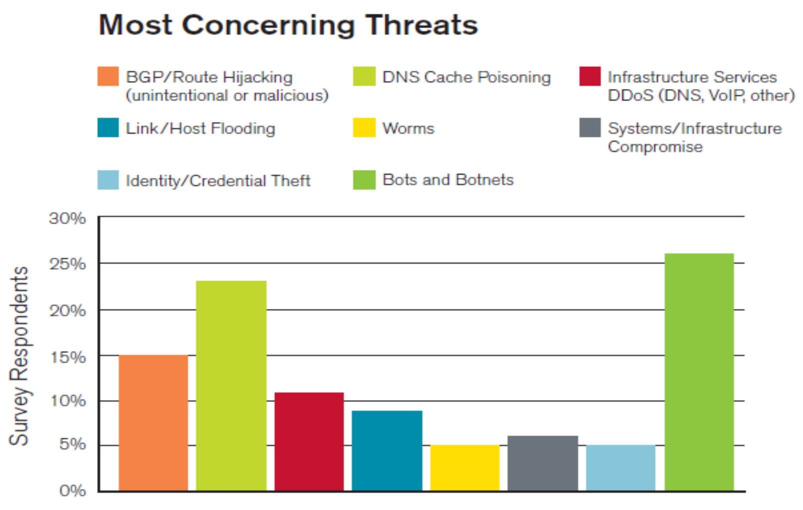
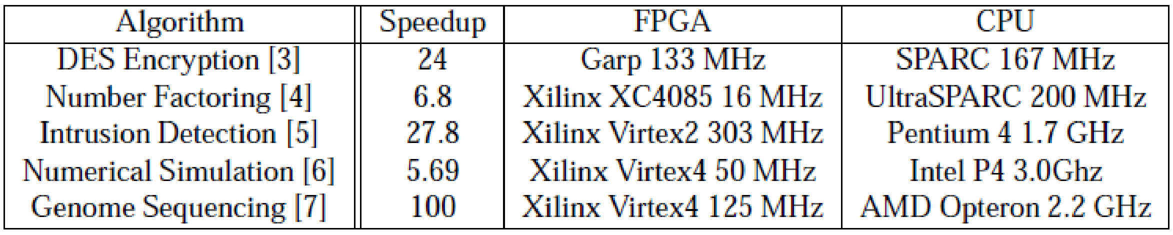


Figure 2-8: Most Concerning Threats [56]

In an effort to fight recent attacks there has been a general tendency of moving towards hardware based IDS instead of improving software based solutions. This could be due to the rapid evolution of bandwidth and the speed at which attacks are performed in today's Internet. Figure 2-9 shows that IDS overall speed could be improved about 28 times if solutions were implemented in hardware.



Algorithm	Speedup	FPGA	CPU
DES Encryption [3]	24	Garp 133 MHz	SPARC 167 MHz
Number Factoring [4]	6.8	Xilinx XC4085 16 MHz	UltraSPARC 200 MHz
Intrusion Detection [5]	27.8	Xilinx Virtex2 303 MHz	Pentium 4 1.7 GHz
Numerical Simulation [6]	5.69	Xilinx Virtex4 50 MHz	Intel P4 3.0Ghz
Genome Sequencing [7]	100	Xilinx Virtex4 125 MHz	AMD Opteron 2.2 GHz

Figure 2-9: speed improvement of hardware over software

This improvement is significant as the number of packets analysed can be boosted considerably whilst the number of packets queuing to be analysed will also be reduced. Later in this thesis the author demonstrates that Snort does not perform well in a high speed environment. If Snort is able to process x packets during a period of time using a software implementation, Snort would be able to process 27.8 x more packets. The study will show that Snort does not process many packets at high speed and instead drops them without analysing them. When this happens the chances are that attacks will not be detected hence there is a very high chance for these attacks to be successful. This implies that all systems protected by Snort at high speed might be as transparent to attacks as systems with no protection at all.

Speed definitely matters when it comes to securing Internet based systems. Recent research as in Figure 2-10 [59] shows a serious increase in bandwidth usage in the UK. In 2008, an estimate of 16.46 million UK households has been using the Internet which represent 65 per cent of households and an increase of 1.23 million households since 2007 [60]. The current average download speed of broadband in the UK is currently 3.6 Mbs [61]

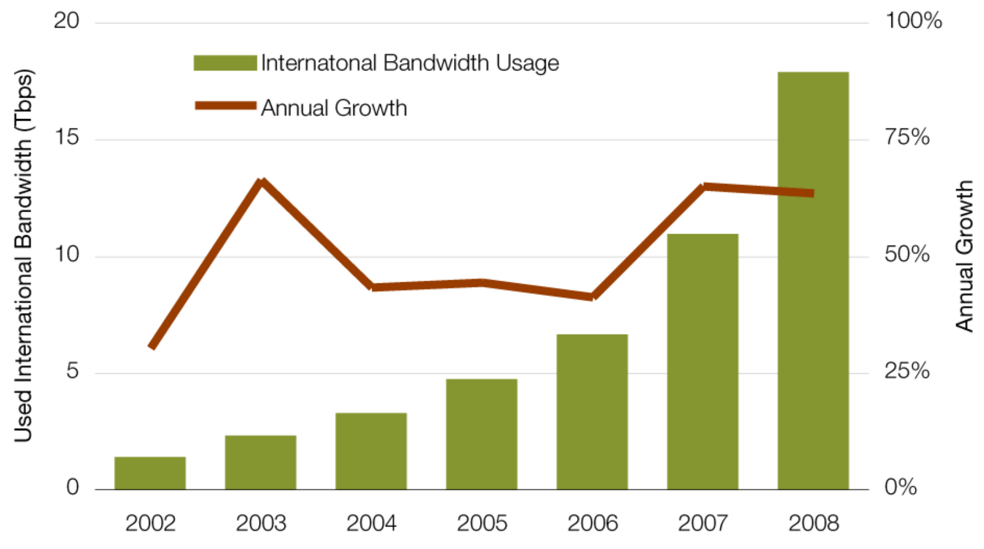


Figure 2-10: Bandwidth usage growth

Internet security systems must keep up with the latest advancements in technology. Research shows that attacks are performed at higher speed hence using more bandwidth. Recent tests performed on Snort show that Snort has a very weak ability at detecting attacks at high speed. Yet hardware based IDS are expensive and are not in the reach of most companies or organisations. Snort can take advantage of multi-core processors widely available in home based computer systems. One of the tasks of this thesis is to investigate why Snort components fail to perform under high speed, and how this situation can be revoked. Snort can be improved in many ways. These include but are not limited to: improving detection rate; improving the number of packets processed; reducing false positive, etc. Each of these aspects could be addressed differently. However, the improvements of each of these elements separately would not necessarily ensure the overall performance as improving one aspect could generate other issues. During our tests, Snort did not detect all the attacks.

In recent years, multi-cores technologies have become more and more common. Intel Corporation has modified Snort IDS to run on a multi-core platform. As a result Snort processed the same number of packets 6 times faster. This demonstration suggests that multi-core processors can be used to speed up Snort. However, Intel Corporation did not make any changes on the detection mechanism. As a consequence, even if Snort was to process packets quicker, there would still be some unresolved issues. As discussed in our section “related work”, current work on improving Snort is mostly based on the processing speed. This research looks at improving the detection mechanism as well as the processing speed.

2.8 Multi-core evolution

The constant evolvement of technology has resulted in the need for better computers. Looking at home users, people need better systems to handle the latest video and picture quality. In addition users will also need to download bigger files. In general, they will need better systems to benefit from the latest technological advancements. From an industrial perspective, there is a need to better manage the work environment; a need to facilitate network sharing resources; a need to collaborate with partners around the globe, and a need for faster communications. The first response for these needs was the AMD64 processor architecture [62]. Since then it has been possible for Advanced Micro Devices (AMD) to support multiple cores in one processor. The benefits are listed, but not limited to: less power consumption; concurrently executing programs that are processor intensive such as database searching, image processing, ripping and burning audio and video CDs or DVDs and downloading heavy files from the Internet. As for multi-threaded capabilities, computers have been able to perform concurrently multiple tasks also called threads. Some of these tasks include data mining, heavy mathematical calculations, and heavy repetitive tasks. As shown in Figure 2-11, multi-core technologies have become the standard for Intel^{TM} processors as the single core processors could not respond to the market demand and users expectations.

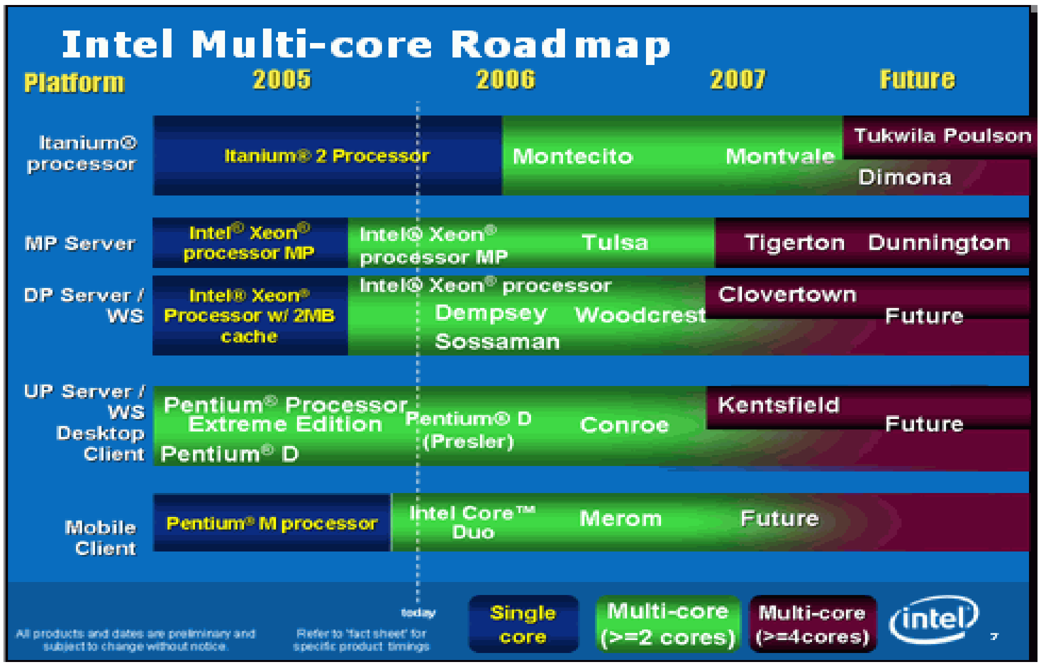


Figure 2-11: Intel Multi-core Road Map [63]

Intel^{TM} has been developing processor micro architectures with the objective of reducing the power consumption. For that reason, the processor's clock speed depends on 2 factors, the clock speed and the number of instructions per cycle [63]. The performance can be computed by

[Performance] = [Clock speed] x [Number of instructions per cycle]

Another important indirect factor of the overall performance of CPU is the power consumption. As predicted by Moore's law, Figure 2-12, the number of transistors has been growing.

In 1965, Moore stated that the number of transistors on a chip will double about every two years. Intel has kept that pace for nearly 40 years. However much was not said about on how the transistor power would scale.

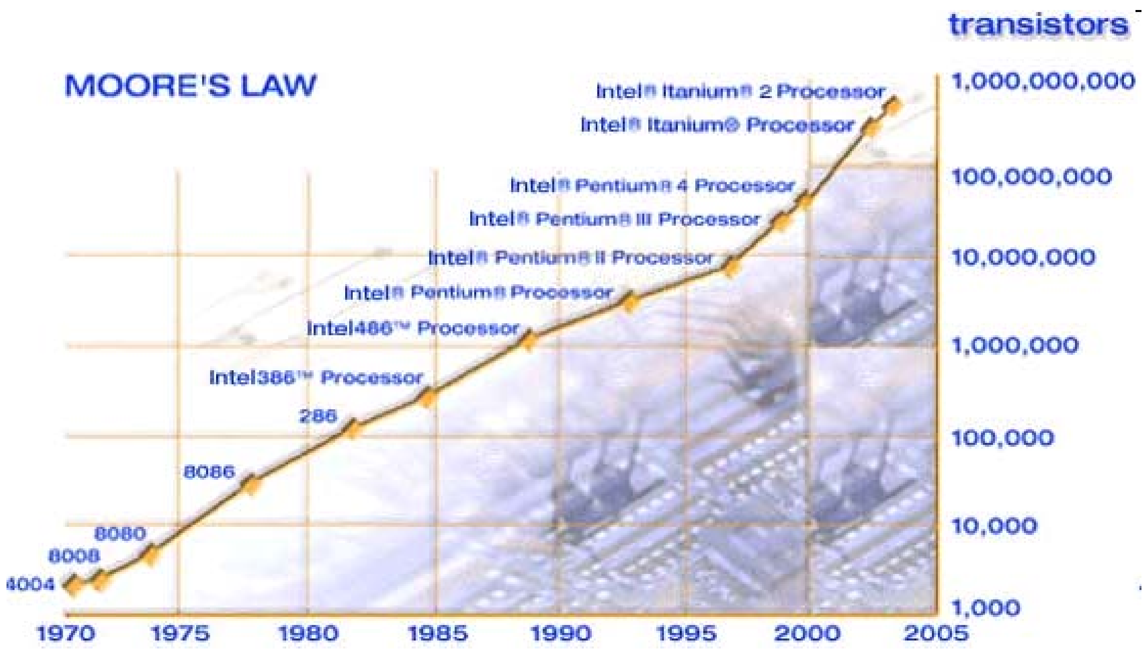


Figure 2-12: The Moore Law

The processing power which is measured in millions of instructions per second (MIPS), has steadily risen because of increased transistor counts. But Moore's Law can also mean decreasing costs. As silicon-based technology gains in performance, they becomes less expensive to produce, more plentiful and powerful, and more seamlessly integrated into our daily lives

Figure 2-13 show that a better performance is reached when there is less power consumption.

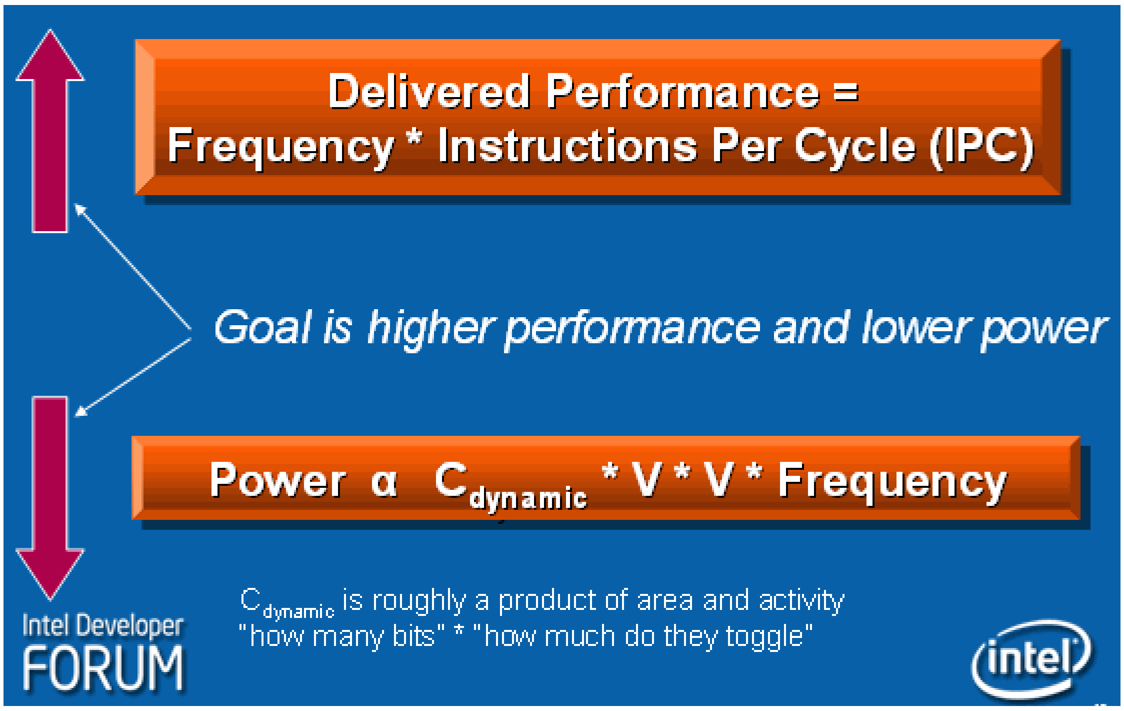

Delivered Performance =
Frequency * Instructions Per Cycle (IPC)
Goal is higher performance and lower power
Power α $C_{dynamic}$ * V * V * Frequency
$C_{dynamic}$ is roughly a product of area and activity
"how many bits" * "how much do they toggle"
Intel Developer FORUM
intel

Figure 2-13: Performance Over Power Consumption

The formula for the power consumption would be:

[Power consumption] = [Dynamic capacity] x [Voltage] x [Voltage] x [Clock speed]

The multi-core technology clearly offers many advantages. However, changing current implementation of network application is challenging. Few approaches in the analysis of parallelising network application need to be considered: Independent process on each core, pipelining which divides application into various stages and the symmetric multi-processing which runs identical process in parallel with a load balancer to equally share the tasks load amongst the different cores [64]. These parallel approaches will be discussed further in the architecture design and implementation.

The advancement in microchip has caused a shift to IDS hardware [65] away from the traditional software.

2.9 Related work

In their work Wheeler, P. and E. Fulp [14] propose a framework to parallelise Network Intrusion Detection Systems (NIDS). They suggest 3 levels at which parallelism could occur: the node level, the component level and the sub-component. For the node level, they suggest that multi identical systems to run in parallel where rules are taken from their original group and spread across all the running nodes. Snort organise rules into groups and each of these groups is generally identified by its filename. For instance “pop.rules” will refer to all the rules related to POP protocol, the Post Office Protocol. Incoming packets are duplicated by a packets duplicator across all the nodes at the same time, identical rules are sent to the different nodes. This suggests that one packet will go through the same inspection many times. This method clearly suffers from repetition. They also propose a variance for which when a packet is sent to a node, the node will check if there is a rules in relation with that packet. Even though Wheeler and Fulp [14] do not give details about how the check would be done, the Author argues that there are many inconveniences with the Node level. Firstly, the node level would work only if all communications are considered to be stateless which is unrealistic with today’s attacks described by [66-68]. Secondly, there are endless repetitions. In Addition, there is no correlation between the packets sent in multiple frames. Also, this method does not take into consideration fragmentation which is one of the latest techniques used by malicious users to overflow systems as current IDS do not handle fragmentation at high speed [69]. This has also been proven by some of our results that show that Snort detection rate of fragmented packets will drop about 95% when speed changes from 0.1 Mb to 10 Mb refs. The architecture that proposed in the work carry out in this research will consider dividing packets without repeating them. Also, the author has introduced a flow correlation for attacks spread over multiple flows. This will be presented under the shape of context record management that will help correlate detection across the multiple parallel processes.

At the component level, Wheeler and Fulp suggest that specific functions such as defragmentation might be parallelised. This could be interesting. However, there is not a clear definition on how this will fit into the overall system. There is a risk of creating a bottleneck at this level if a top level classification is not done in order to separate fragmented packets with complete packets.

Paxson et al [10]define an architecture that ignores keyword matching as they argue that the level of sophistication of attacks have gone beyond the keyword matching. The same idea is supported by many researches [70-72]. However, not only that, there is still a lot of research trying to improve keyword matching [73-76] Snort remains by far the most popular IDS due to its ease of use and modification. It is commonly agreed that only keyword matching would not be good enough to prevent against the latest attacks(Barman, et al., 2009). Having said that keyword matching remains a great tool for detecting attacks [77]. The architecture presented in this research will be based on keyword matching with additional level of packet inspections. A correlation between different keywords would be done before alerting as the author believes that a stateless keyword matching is not efficient for recent polymorphic attacks. In their Model as shown in Figure 2-14, they define three stages.

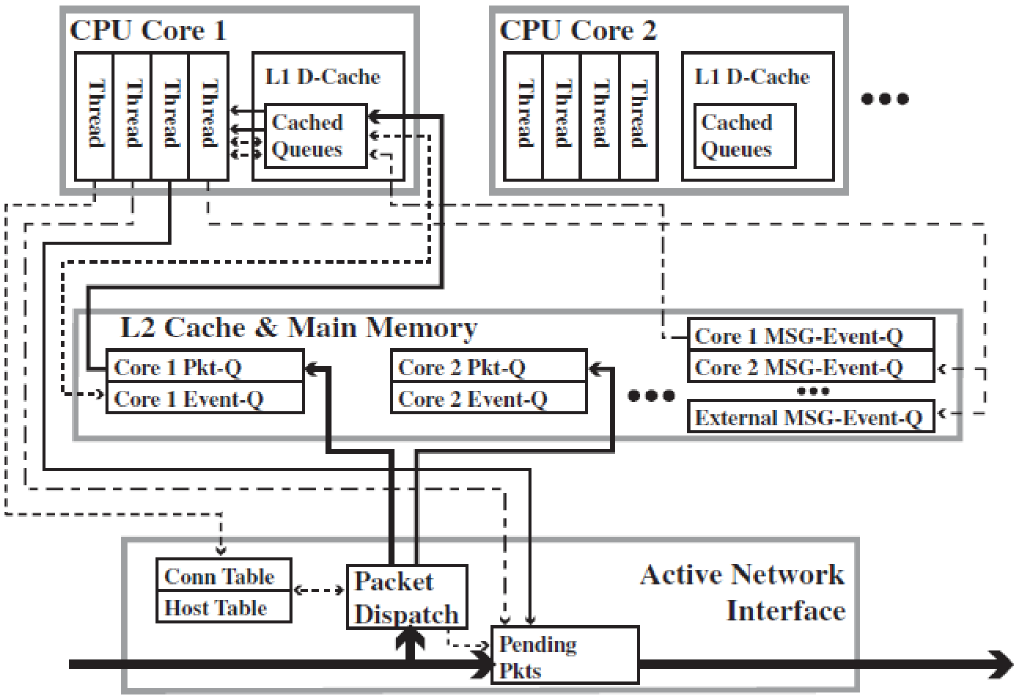


Figure 2-14: Parallel Execution of Network Analysis [78]

At the first stage, they perform all packets reassembly before proceeding to any analysis. The author argues that this could be a major inconvenience for the whole structure. An attacker could send millions of fragmented packets and that will cause the analysis to be slow. Also, they argue that “Ideally, the front-end ANI would retain each packet until all events to which the packet contributes in any way” have fully processed. This has some inconveniences in packets processing. There will certainly be a delay in communication and this may require more buffers to handle big numbers of packets. The architecture the author proposes will correct important missing features for a first stage parallel architecture. A good number of security features can be implemented in the first stage or layer of protection of IDS. Spoofing is generally used when an attacker is trying to remain anonymous. For strong first layer of protection, the architecture proposed in this research will eliminate all unnecessary traffic present in the wire. The IANA reserved address should not appear in any routing table. Hence at the first level of security, the author cleans any traffic that should not be in the routing table.

At their second stage, they define a series of parallel processors based on events. This approach is similar to what Wheeler and Fulp [14] describe in their model but with more details of what is been processed in parallel.

Even though Paxson et al [10]aim at building a network IDS that will be used by general-purpose commodity hardware, their work has been based on a particular hardware the ANI device. This does not guarantee that other hardware will support the architecture they implement. No specific limitation was made as per the type of hardware supported.

In a white paper, Intel Corporation [15] claimed to have improved Snort performance by a multiplication factor of 6 in the best case. They have adopted 3 approaches in improving Snort. The first approach runs five functional Snort process in a single core. The five functional processes of Snort are known as packet capture, packet decoder, preprocessors, detection engine and output plug-ing [28]. In the second approach, Intel ran the five functional components on each of the cores. This approach is referred to as the node parallelisation level in other research [14]In the last approach used by Intel in these experiments, the capture was executed in one core and all the other cores were running in parallel. There has been a great achievement as claimed by Intel. However, there has not been any security improvement over Snort architecture. The current implementation of Intel certainly improves the speed of Snort but does not provide any relation between flows enabling multi-stage attack detection such as attacks identified by [79-84]. There are important limitations on the accelerated implementation performed by Intel Corporation in regards to recent attacks. For static attacks that are all contained in separate flow without any relation to other flow, the implementation discussed here would be a very good improvement of Snort. In addition, Intel does not give details on the modification that they have made on Snort hence the difficulty to repeat their experiments.

Even though Snort aims at offering an overall security, they are other valuable research works that have been accomplished looking at application layer security especially web services [85] [86] [87].

2.10 Conclusion

A great deal of work has been done in advancing the effectiveness of security systems. Before parallel IDS were discussed, attacks that are split into different stages have always been very difficult to analyse, detect and mitigate. With the shift toward parallel IDS, multistage attacks would be even more difficult to detect. The difficulty resides in the fact that there is no correlation between the different cores that perform the analysis. The IDS will certainly improve in terms of speed i.e. the number of packets processed per second and at the same time, but when attacks are split into different flow, most current systems do not correlate flows. In this research, the author adds that dimension to the existing system. However, this would cause the IDS to be redesigned and that’s what this research is all about.

3 The problem

3.1 Introduction

Snort is a Network Intrusion Detection (NIDS) that was considered to be a lightweight Intrusion Detection System IDS [88]. However, technology has evolved and Snort has been considerably improved [89]. Snort remains an open source network intrusion prevention and detection. It is based on a language rule-driving used in combination with signatures; signatures and protocol anomaly based inspection methods [90]. Despite the big improvement over the years, Snort stills struggle to keep up with the fast growing network industry and attacks [91].

Many researches [91] report the inability of Snort to cope with current attacks. This triggered the author to test Snort in order to see its limitation and propose applicable solutions. In this chapter, the author will

- Review the trend of latest threats and attacks on the Internet
- Test Snort accordingly to these threats and attacks to study its ability to resist current and future threats and attacks
  - HTTP Complex multistage attacks.
    - Obfuscated JavaScript
    - Obfuscated HTML
  - Flood attacks (ICMP, UDP, HTTP)
  - Other not so well classified attacks
- Propose solutions to the problems identified

Solutions proposed in this chapter will be integrated in the design of our new IDS.

3.2 Security trends: threats and attacks

From analysis performed by Arbor Network Inc, the size of the attacks has grown almost double from 2007 to 2008. In the course of last year, 2009, the size of the attacks has continued to growth by over 22% [92] [93].

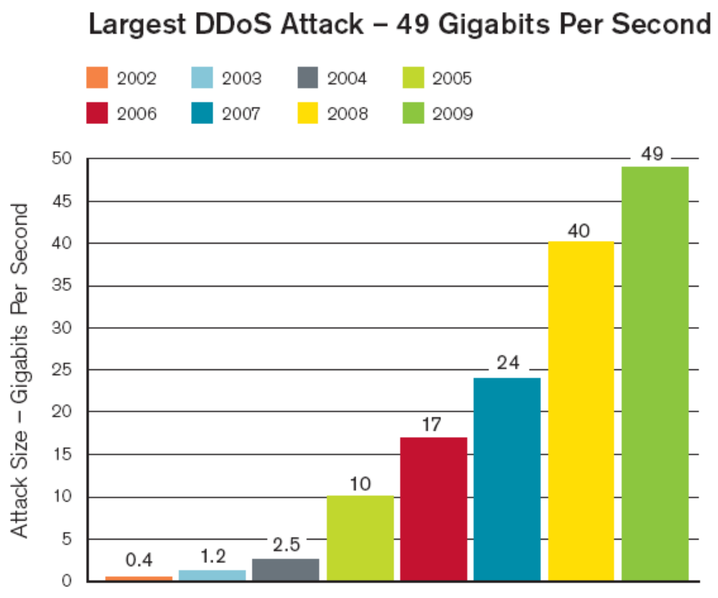


Figure 3-1: Attack size, ([92])

On another note, Arbor Network Inc, anticipated that the Link, Host or Services DDOS as the single biggest attack on the Internet for 2009 as shown in Figure 3-2

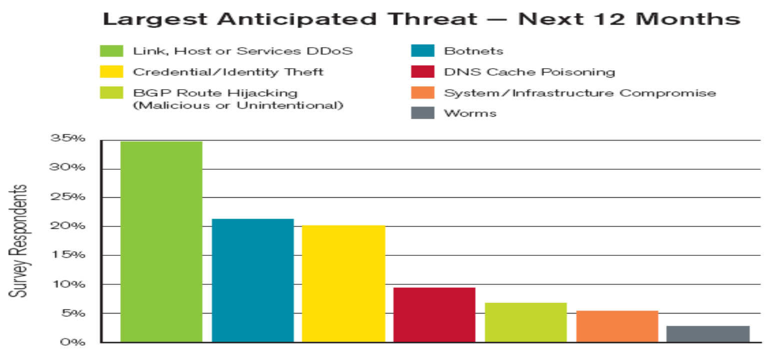


Figure 3-2: threats prediction for 2010 ([92])

The above attack’s classification does not clarify what is really included in the Link, Host or Services DDOS as these days hackers use every single opportunity to make any computer a zombie. McAfee reports of threat prediction for 2010 will go into more details by naming social networking sites as one of the biggest threats for Internet security stability. The reason behind social networking being the biggest threat is that more and more people have joined social networks. For instance Facebook network is as big as 350 million users with more than 350,000 applications as claimed by McAfee in their report. As social networking is still relatively new, most people are inclined to be curious. This attitude is not necessarily in favour of security as most people tend to click on any link. In addition the Internet has seen the so called “tinyurl.com” use to shorten Internet links. However, when using tinyurl.com, the user does not know the real link and is more likely to click on the link [94].

Botnet activities are relentlessly increasing either by using malware or Trojan through emails or by taking advantage of social networks both for mobile devices as well as for PCs [95]. Botnets have been used mostly for SPAM but recently a move toward cyberwarfare has become popular [96][97].

Search engines poisoning is yet another attack that has made victims of millions [98]. Internet users, especially social community users, are tricked into thinking that they are using a genuine search engine but yet they are redirected to results (i.e. links) that, when opened, install malware and Trojans [99] [100].

In the light of these recent trends, the author chose to test Snort for its resistance against

- Flood attacks (i.e. ICMP, UPD, HTTP/TCP),

- DDOS attacks
- High speed networks
- Malware intrusion
- Recognition of botnet traffic.

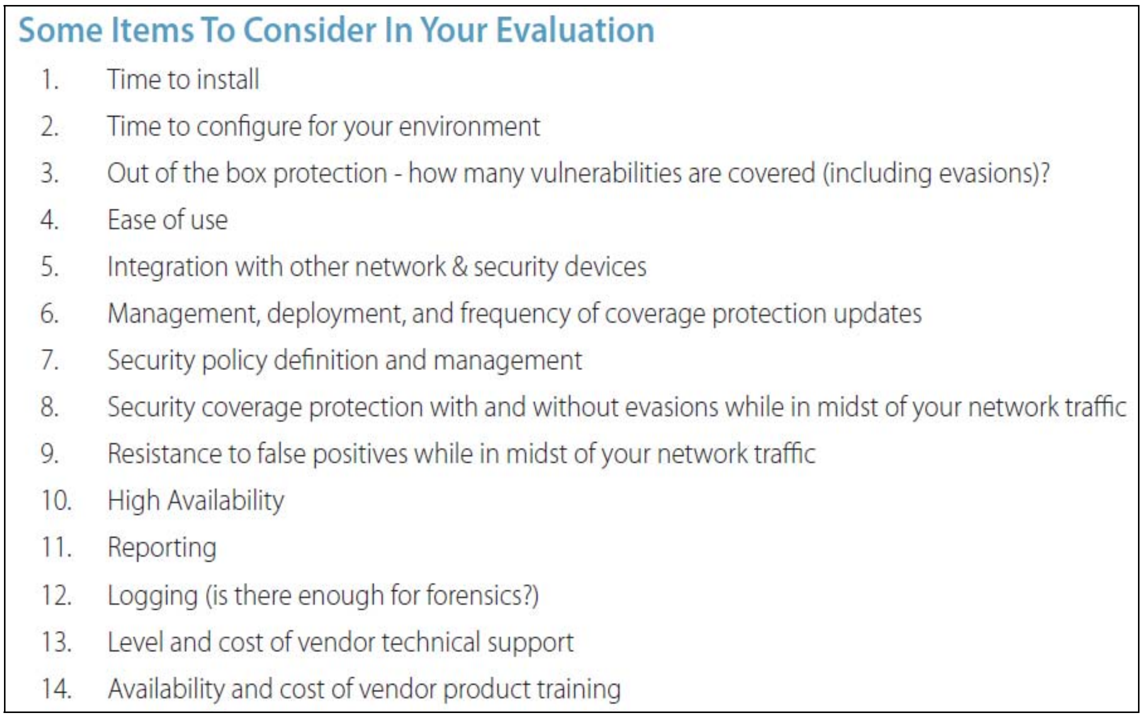
The above list of tests may cause confusion between DDOS attacks and flood based attacks. DDOS attacks are typically any distributed effort to cause the system to stop offering services it is intended for. However, flood based attacks are a type of DOS or DDOS attacks are they generally cause the system to stop responding and offering services. DDOS attacks can be performed either by generating millions of small packets against a victim of generate only few jumbo packets that will cause the same effect. In the latter case, there has not been any flooding.

The objectives of the author here was to test Snort against attacks that are current and attacks that are likely to affect system in the future. Traditional attacks for which many solutions exist already or attacks that are very specific to a system have not been given priority. These include attacks such as buffer overflow, SQL injections.

3.3 Choosing the IDS

Even though snort is the most popular IDS, Bro is nevertheless one of the most interesting IDS used for research purposes [43]. Various comparison have been made comparing Snort to Bro. Most comparisons aim at guiding a customer who is trying to purchase an IDS product. For instance, [101] suggested a checklist of features that need to be met before purchasing any IPS. These features are found Figure 3-3.

Looking at the features identified by the ICSA, there is no concern as to how the system is built. What is important in this case is the performance of the IPS based on the identified criteria. For research purposes, it would be more interesting to look at the different architecture, the programming languages, the structures used when designing the IPS, the level of customisation possible, and the support available for further research.


Some Items To Consider In Your Evaluation

1. Time to install
2. Time to configure for your environment
3. Out of the box protection - how many vulnerabilities are covered (including evasions)?
4. Ease of use
5. Integration with other network & security devices
6. Management, deployment, and frequency of coverage protection updates
7. Security policy definition and management
8. Security coverage protection with and without evasions while in midst of your network traffic
9. Resistance to false positives while in midst of your network traffic
10. High Availability
11. Reporting
12. Logging (is there enough for forensics?)
13. Level and cost of vendor technical support
14. Availability and cost of vendor product training

Figure 3-3: ICSA IPS comparison features

Very little comparison in search for the best tool for research purposes have been made between Snort and Bro. Bro is a Network Intrusion Detection System (NIDS) which is highly customisable. The first purpose of Bro has always been defined as a research tool that can be used to advance detection technique against Intruders [102]. . Rather than being an "out of the box" solution, Bro was geared at UNIX expert

Bro is designed for use by UNIX experts who place a premium on the ability to extend an intrusion detection system with new functionality as needed, which can greatly aid with tracking evolving attacker techniques as well as inevitable changes to a site's environment and security policy requirements. [103]

One of the major drawbacks that we found for the research carried in this thesis was that Bro does not provide any default security feature. Bro has to be tailored to the network in which he is installed [43]. Correlation of event is important to ensure a good overall protection of the system. Bro provides a better correlation of events that Snort in the fact that Bro uses syslog output as an input to create a better picture of what is happening in the system. This feature is nonexistent in Snort and will be considered as an addition if the work carried in this research is based on Snort.

The author looked at the different communities related to Snort and Bro to ensure that help could be provided when needed. Snort

Snort community represented in Figure 3-4 is estimated at 300,000 registered users.

With nearly 4 million downloads and approximately 300,000 registered users with more than 4 million download, with hundreds of universities actively using Snort for research purposes or simply for tutorial [104]. As opposed to Bro, there is no clear indication how many people are involved in the community. [43] recognises that involving the community has not been a prime objective.

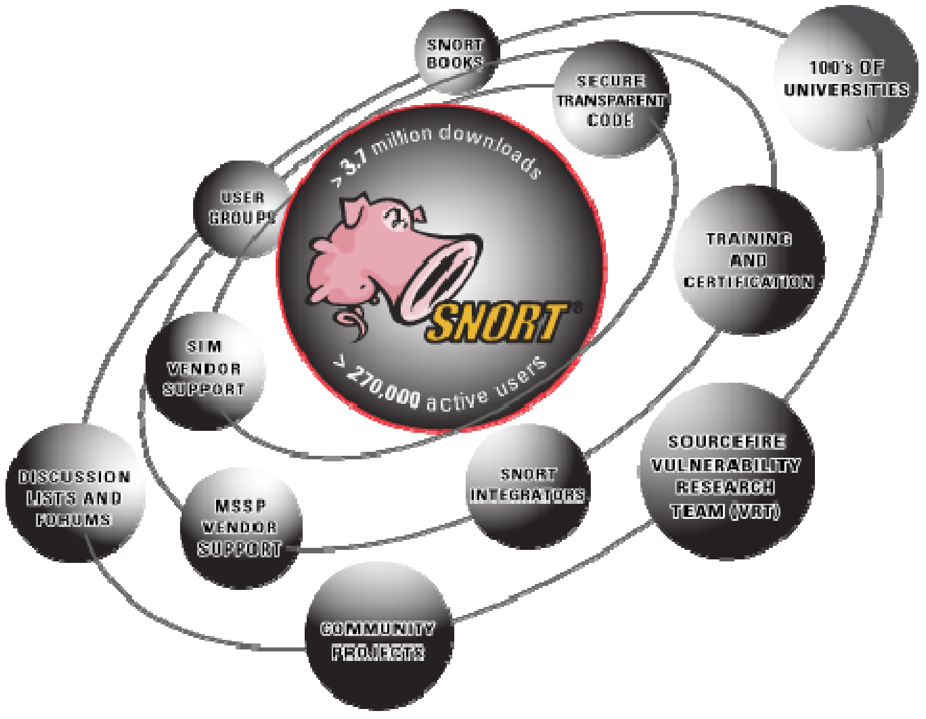


Figure 3-4: Snort community

The author also looked at the number of tutorials available for both Bro and Snort. A Google search returned 23,800 for “Bro IDS tutorial” and 438000 for Snort. This means that Bro provides an equivalent of 5,434% support as compared to Snort. At the time the author started this research, his knowledge of UNIX system was very minimal.

Looking at the research platforms, Snort is supported for virtually any Operating System (OS) yet Bro is only on UNIX like OS. In addition, the level required to use Bro is of a UNIX expert.

Without a shadow of a doubt, the level of analysis that Bro provides is far superior to the one that Snort provides. There are many limitations to keywords based IDS which mainly rely on the fact that a previous attack was successfully analysed and represented as rules. These rules are then used to detect future identical attacks.

Snort is mainly based on C programming language, yet Bro is based on various languages with specialised scripting language i.e. Bro custom language.

The criteria rating technique [105] was applied to make a decision on whether to perform the test using BRO or Snort. Each of the factors identified were given a weighting factor in the overall comparison based on literature review. The following table was produced with a scoring mechanism ranging from 0 – 100; the highest score representing the most favourable option.

3.4 Snort overview

The basic structure is represented in Figure 3-5. When a packet arrives at the network, Snort listens and captures packets. The packet is then parsed and sent to the appropriate preprocessor for more analysis such as the “http_decode” responsible of normalizing HTTP traffic.; The minfrag preprocessor is another example of pre-processor and it deals with mini (tiny) fragments. Any tiny fragment found on the network is then sent to the minifrag preprocessor for more analysis.

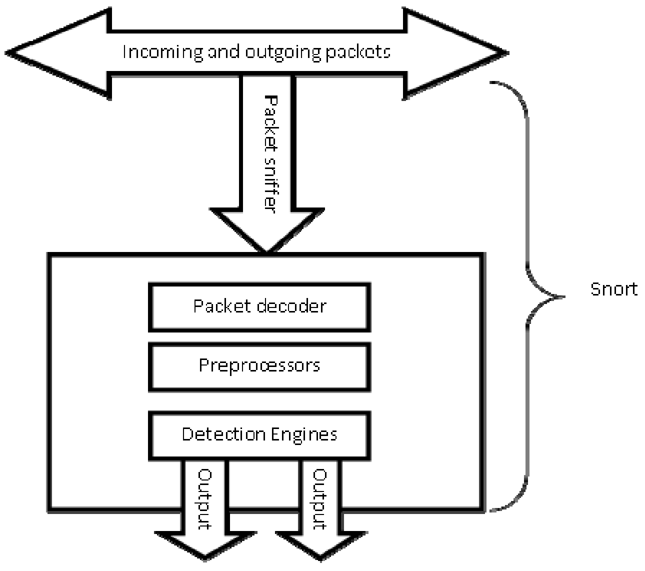


Figure 3-5: Basic Snort Architecture

The preprocessors are also referred to as plugins. There are currently three types of plugins in Snort which are preprocessors plugins, detection plugins and output plugins. Once the preprocessors job done, the packets are passed to the detection engine that will cause Snort to either fire an alert, or log an alert in the case of IDS or drop the packet in the case of IPS.

3.5 Testbed

The main testbed used for this experiment is represented in Figure 3-6

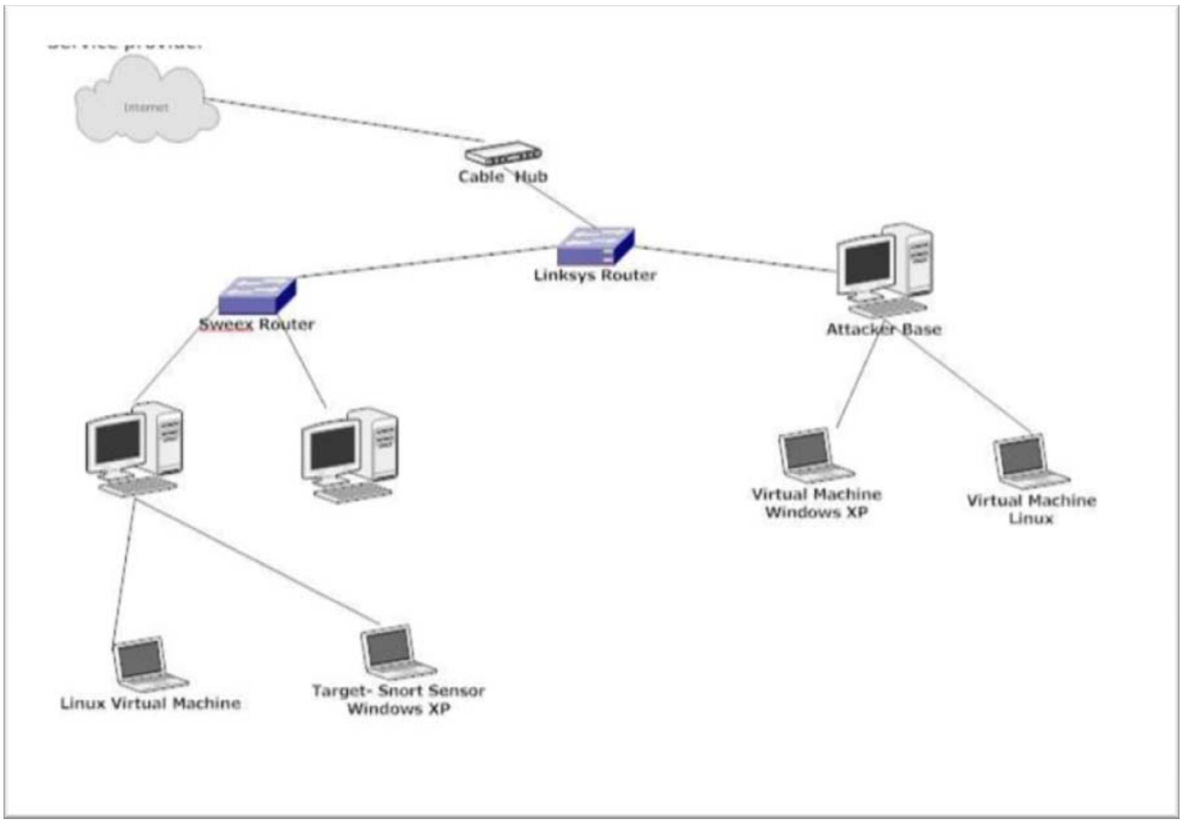


Figure 3-6: testbed

3.6 Test under high speeds networks

Under Fedora, a Linux distribution, the author ran over 5 consecutive tests to analyse the Snort performance using the number of packets received, the number of packets analysed, the number of packets dropped, the number of alerts and the number of logs as our parameters. For each of the tests, the speed at which the packets were sent was increased. The tests started by running [snort –r]. The author then used Tcpreplay to vary the speed at which Snort received the packets. As a result, the author observed that Snort analysed every single packets that reached the wire. The number of alerts produced was optimal as Snort was controlling the speed at which each packet was sent. The results are presented in Figure 3-7.

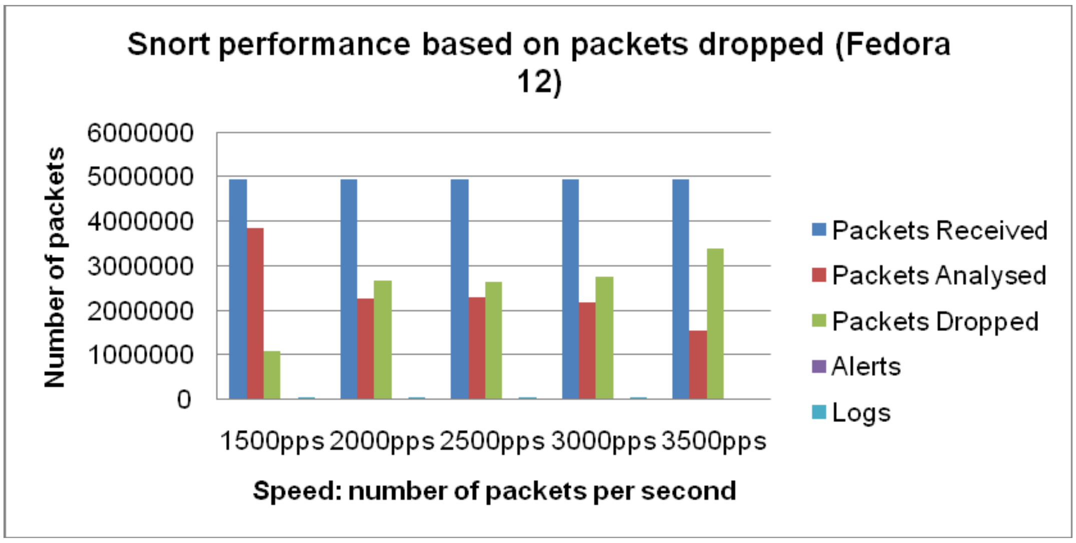


Figure 3-7: Snort performance under controlled speeds

Figure 3-7 shows that the number of packets received remained constant while the number of packets analysed changed considerably. The number of packets received was predefined to allow fair comparison between the different data rate speeds. The author first observed a sudden drop in the number of packets analysed which then remained constant over a certain speed then continued to drop. As the speed increased, the number of packets dropped increased as well. Similarly, Snort logging capabilities were reduced as the speed increased as shown in Figure 3-8. Not only did the number of packets logged decreased, the number of alerts also decreased.

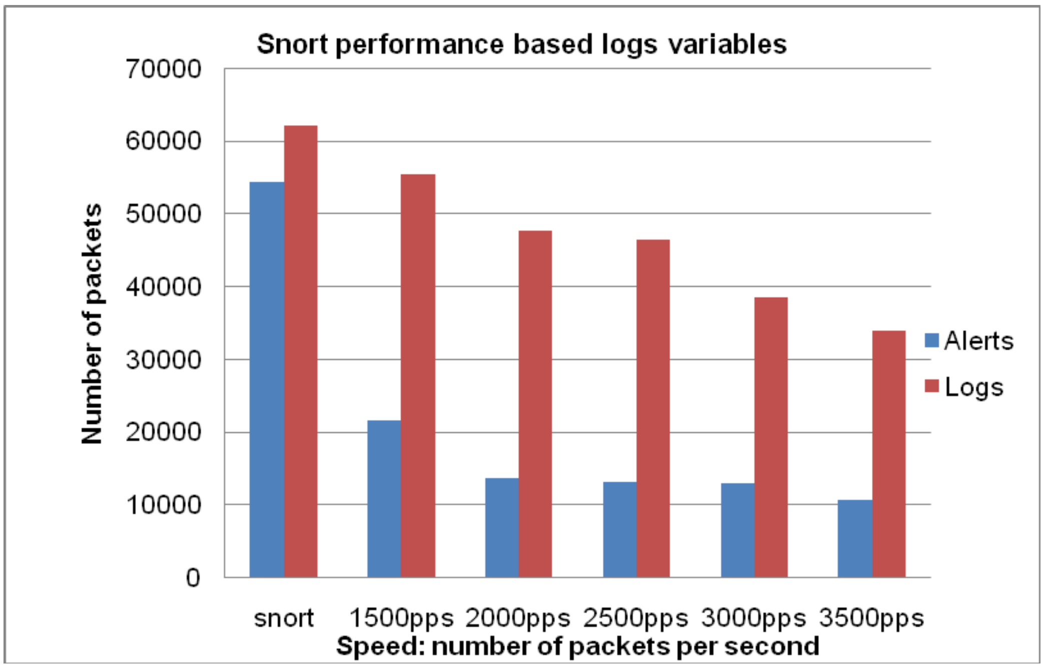


Figure 3-8: Snort performance based on logs variable

It is important to note that under different circumstances i.e. different computer systems and network environments, Fedora could have performed differently, whether better or worse. The results presented here are a representation of the performance of Fedora under our systems.

More interestingly, the author has noticed that the number of IPs that Snort was able to see decreased as the speed increased as shown in Figure 3-9 .

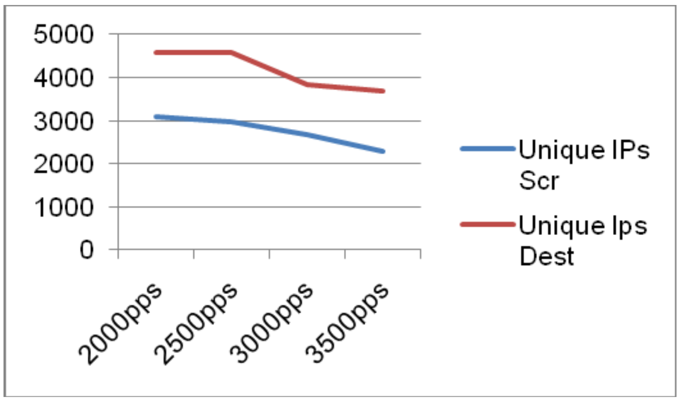


Figure 3-9: Snort performance based on the number if IPs

The latter graph shows that an attacker can take advantage of Snort weaknesses by generating a lot of noise around the attack, using a tool like “bonesi” [106] - a tool that generates up to 50k IPs addresses with up to 150,000packets per second. Our experiments show that Snort was only able to see up to 26% of the IPs when the speed was increased. In this experiment the author has used Tcpreplay to replay the traffic at various speeds. At 2000 packets per second, the number of unique source IP and unique destinations IPs were recorded. The traffic speed was increased by 500 packets per second three times. The third time, when packets were passing the network at 3500 packets per second, Snort was not able to analyse all the traffic. This resulted in a drop of 74% of IP that were passing the network. This suggests that whatever attacks these 74% of IPs were carrying did not get analyse. From 9088 IPs, Snort did only analyse traffic for 2303 IPs. 6785 IPs traffic went undetected.

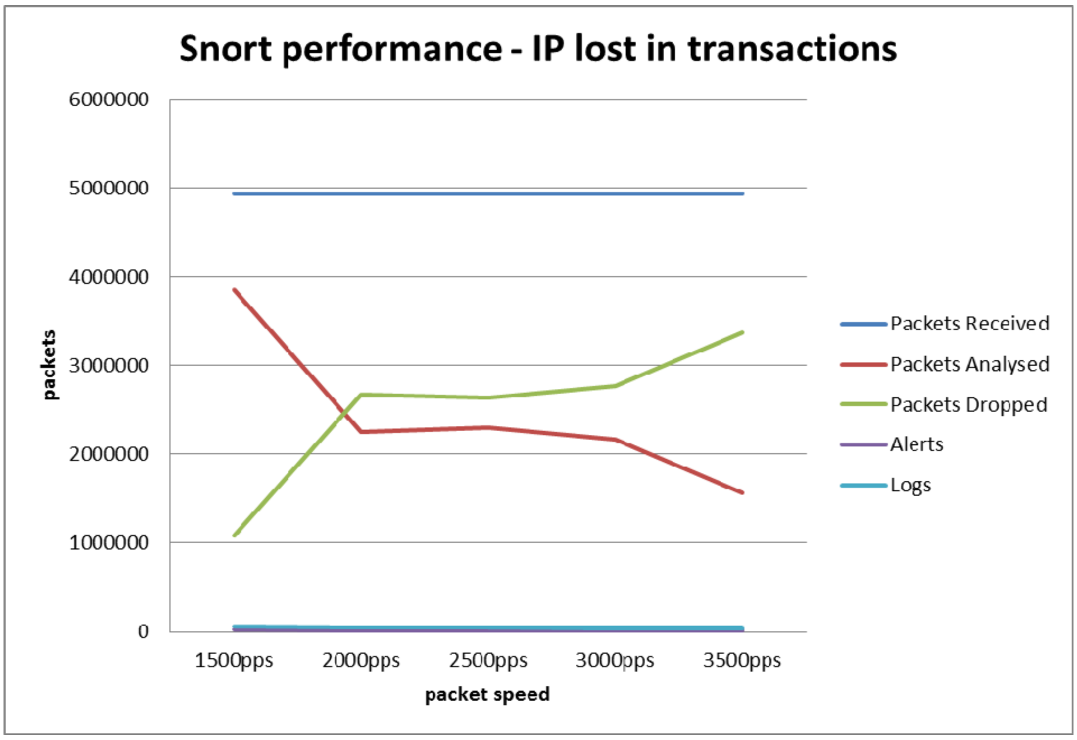


Figure 3-10: IP lost in transaction performance study

Figure 3-10 show that the number of packets received was kept to the same value for a fair analysis between the different speeds. The later graph show a quick increase in packet drop which matches a quick decrease of the number of packet analysed. Table 1 show that from 4933926 packets Snort only analysed 1560217 packets at the end. Hence the big loss noticed earlier.

Speeds	snort	1500pps	2000pps	2500pps	3000pps	3500pps
Packets Received	4933926	4933926	4933926	4933926	4933926	4933926
Packets Analysed	4933926	3848739	2256913	2294284	2166329	1560217
Packets Dropped	0	1085187	2677013	2639642	2767597	3373709
Alerts	54289	21565	13726	13209	12940	10643
Logs	62147	55426	47695	46498	38585	33896
Unique IPs Scr	9088	4722	3084	2964	2685	2303
Unique Ips Dest	10999	6522	4578	4589	3826	3697

Table 1: Statistics IP lost in transaction

There are serious implications to packet dropping. Many attacks simply go undetected and the systems that should be protected become unprotected and open to attacks. Preventing a system to alert has proved to be fatal. In 2008, the computer system that was responsible of alerting fault in the plane during a routine check before taking off, failed to report [107]. Believing that there was not fault, the plane was allowed to take off. Later, the plane crashed causing 154 dead with 14 survivors. Dropping packets without prior analysis could have similar fatal consequences depending on the environment that is being protected.

In the quest of finding what could be the reasons behind that lack of good performance, the research looked at how the rules are analysed by Snort, and how they were performed. After running the "rules performance monitor", a tool that comes with Snort, the author observed that some of rules go through the detection process many more times than others and yet, they did not generate alerts.

SID	**GID**	**Rev**	**Checks**	**Matches**	**Alerts**	**Microsecs**	**Note: rules for …**
11966	1	1	324257	0	0	1136496	Internet Explorer
11965	1	2	117045	0	0	1085734	HTTP SERVERS
3154	1	5	101123	0	0	595723	DNS traffic
11671	1	2	117045	0	0	533102	HTTP SERVERS
2660	1	8	117045	0	0	532240	HTTP SERVERS
16291	1	2	95534	0	0	438676	Mozilla Firefox
2329	1	10	70514	0	0	295143	SQL SERVERS
477	1	3	83173	0	0	219236	ICMP
473	1	5	83173	0	0	215193	ICMP

1838	1	10	37845	0	0	209482	Microsoft Windows
485	1	5	83173	1032	1032	166204	ICMP
13948	1	3	101217	0	0	158021	DNS, Windows
3059	1	5	8039	0	0	128357	HTTP SERVERS
1388	1	15	13225	0	0	127934	Microsoft Windows
2584	1	6	16091	0	0	117668	eMule, Windows

Table 2: Snort rules performance snapshot

In Table 2, for the purposed of presentation the top 15 results were selected from the "rules performance monitor" ordered by the time Snort had spent checking the rules. From the top 15 results, one could observed that only one of the rules "sid:=485" returned an alert. All the other 14 rules were checked and did not returned any alerts. Also, Figure 3-11 shows that the time spent to check the top 15 rules was 39% of the full timing.

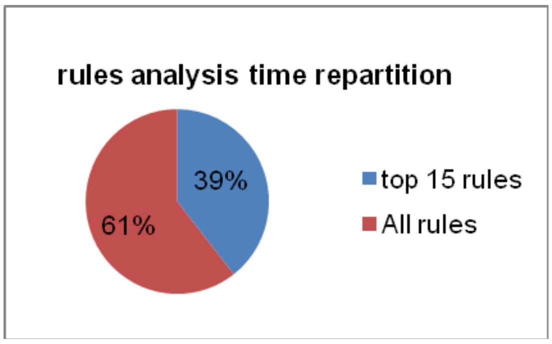


Figure 3-11: Time repartition for rules analysis

Looking further in Table 2 and having identified the category for each rule, the author observed that the same rules are checked whether the work carry was under Linux or Under Windows environment. Despite many rules related to HTTP attacks being checked, the targeted server did not have an HTTP server. Not only was extra time spent performing irrelevant tasks, but any match of HTTP rules would be a false positive. This led us to conclude that Snort performance is affected by performing unnecessary tasks such as checking rules that are not relevant to the system being protected. By reducing the tasks that Snort performs the time that Snort spends checking rules would certainly be reduced.

3.7 Snort reaction to ICMP flood

In this section the research the focus will be on studying the behaviour of Snort against ICMP flood. The objective here is to find out how well Snort performs against flood attacks. Also, the victim system performance is also studied. One of the objectives of the study is to establish at what particular time an alert should be raised in order for Snort to go into “attack state”. Later in this work, similar studies will be carried out studying Snort behaviour against UDP and HTTP floods. This section of the study will help to finding common problems for various situations in order to suggest a solution that will consider most problems.

3.7.1 Experiment 1

Tools:

- Bonesi, a Bot net and DDOS attack simulator – Bonesi was used to generate packet at an average rate of 500 packets per second, each packet carrying 1024bytes , from up to 50K IPs addresses.
- Snort, an Intrusion detection System
- Ifstat , a tool to collect network statistics
- Sar, a tool to collect CPU utilisation

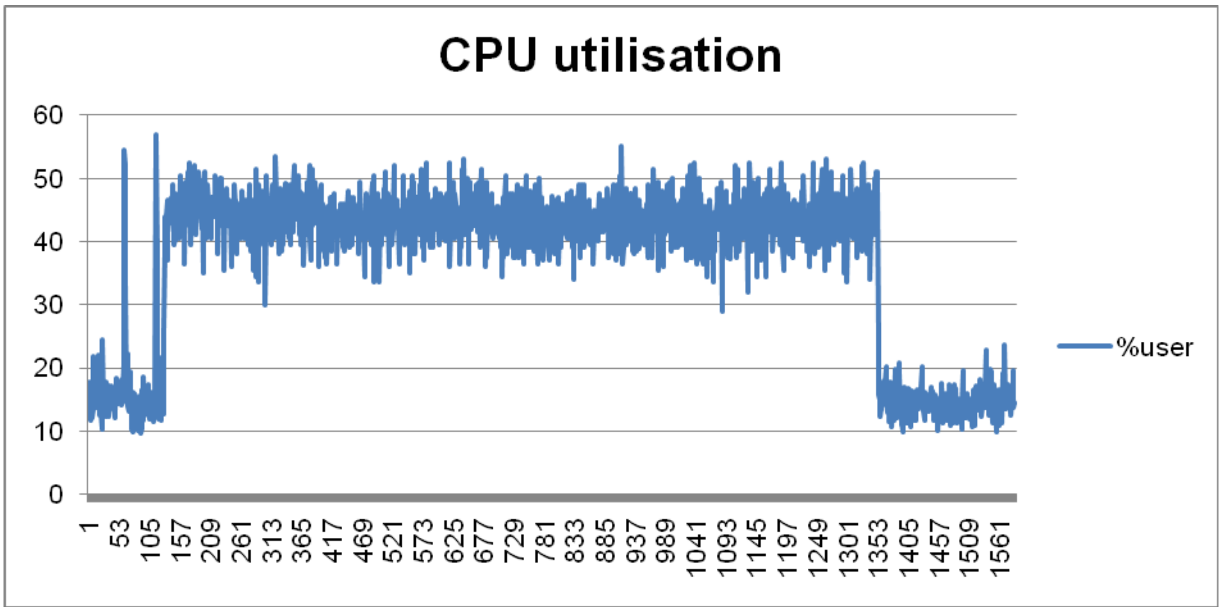


Figure 3-12: experiment 3.1

Based on Figure 3-12, the CPU utilisation has increased from an average of 20 to an average of 48. This study aims at looking at CPU variations when the system is under attack. DDoS attacks, utilize all resources available in order to prevent the Computer System to serve legitimate users. More studies have been done to set the level at which the internal agent will notify the IDS of the attack level hence changing the state of the IDS. Also, a fault can occur and cause the CPU to become very busy. Even though this might not be caused by an external attack, it is important to make sure that Computer Systems continue to provide services for which they are intended. The IDS should be notified that the correct system is not able to handle a lot of traffic. Actions should then be taken to avoid Denial of Service.

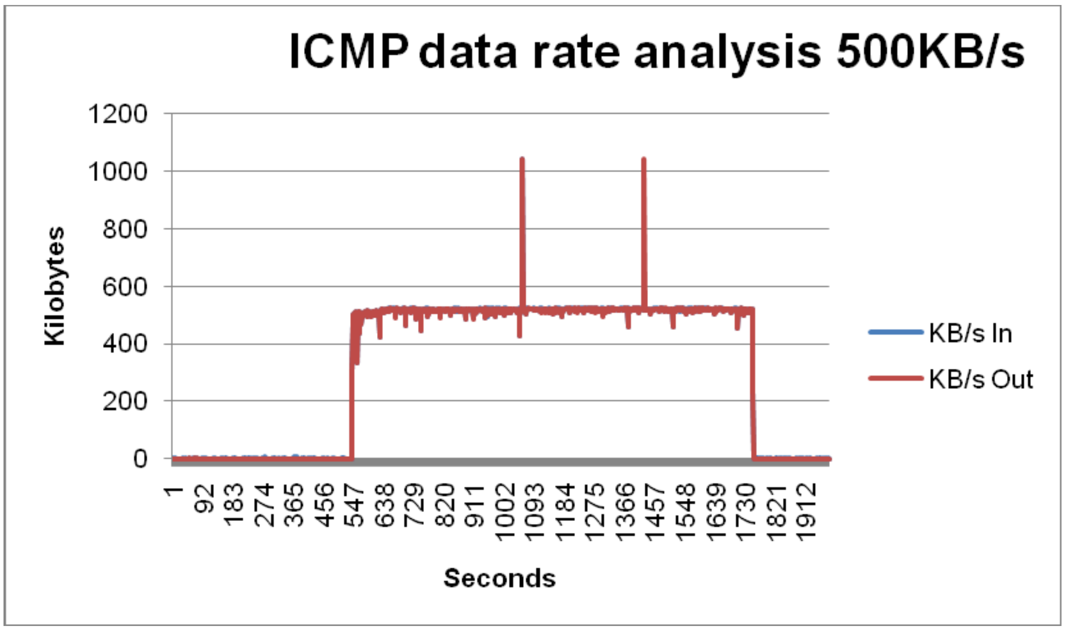


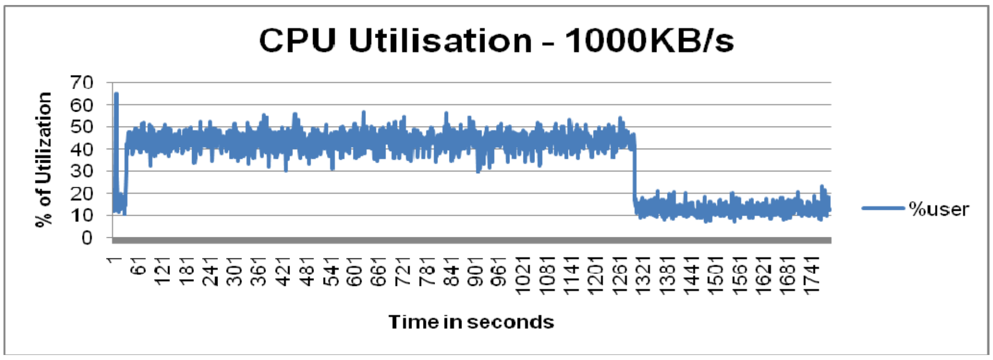
Figure 3-13: ICMP data analysis 500KB/s

In this graph one will see that when the system was not under attack the incoming traffic as well as to outgoing traffic was very low. However, the traffic has increased by 500KBps.

3.7.2 Experiment 2

Traffic speed was set to an average of 100KB/s. When compared to the behaviour in experiment one, the CPU utilization has not changed. However, data speed rate has doubled. Again, one could notice that there are two peak values when monitoring traffic. The same scenario was observed in the second experiment Figure 3-14

Figure 3-14: CPU Utilisation - 100KB/s



This behaviour could be an indication that the IDS were dropping more and more packets. In this experiment only 3% of packets were analysed.

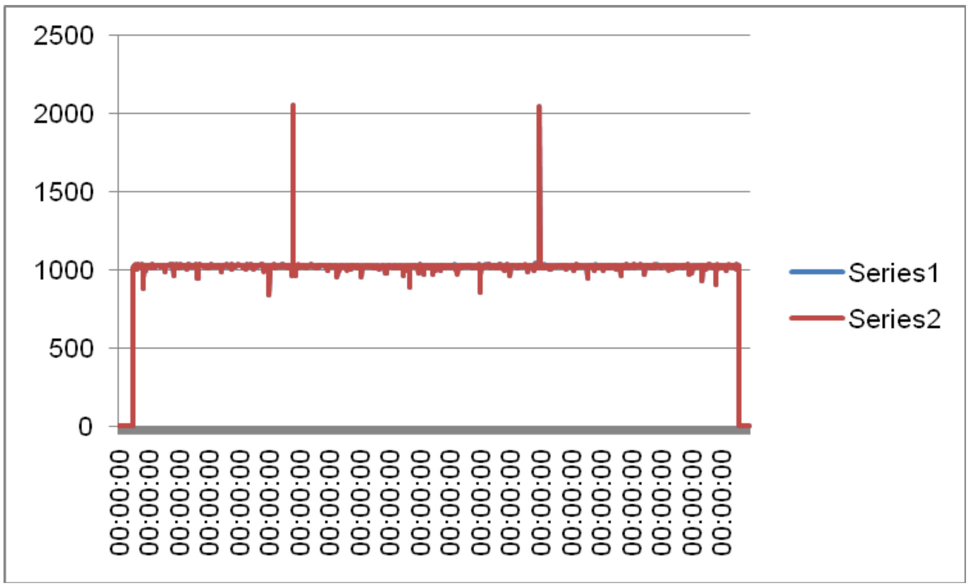


Figure 3-15: ICMP data analysis 1000KB/s

The number of packets dropped was considerably higher when testing Snort against ICMP flood

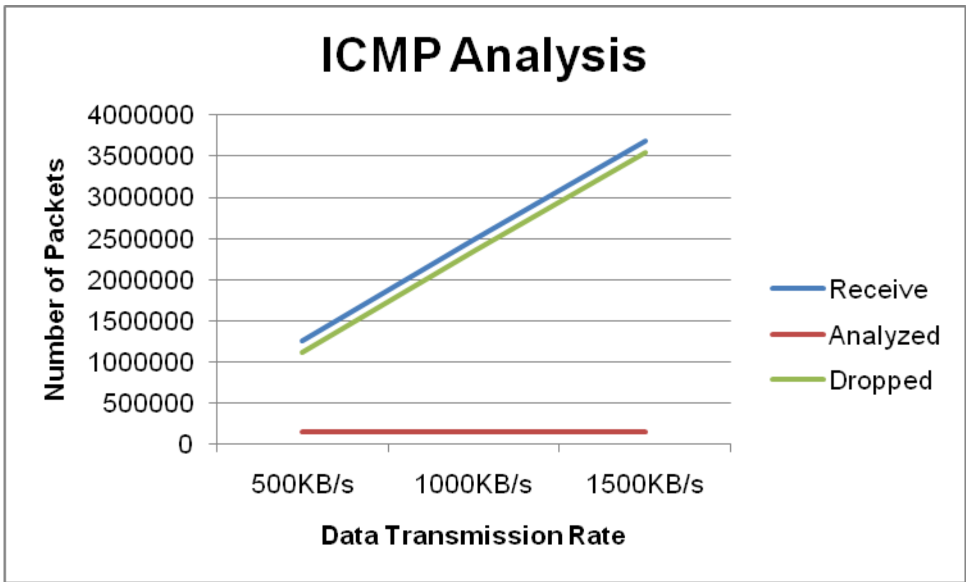


Figure 3-16: Analysis of packet drop against ICMP flood

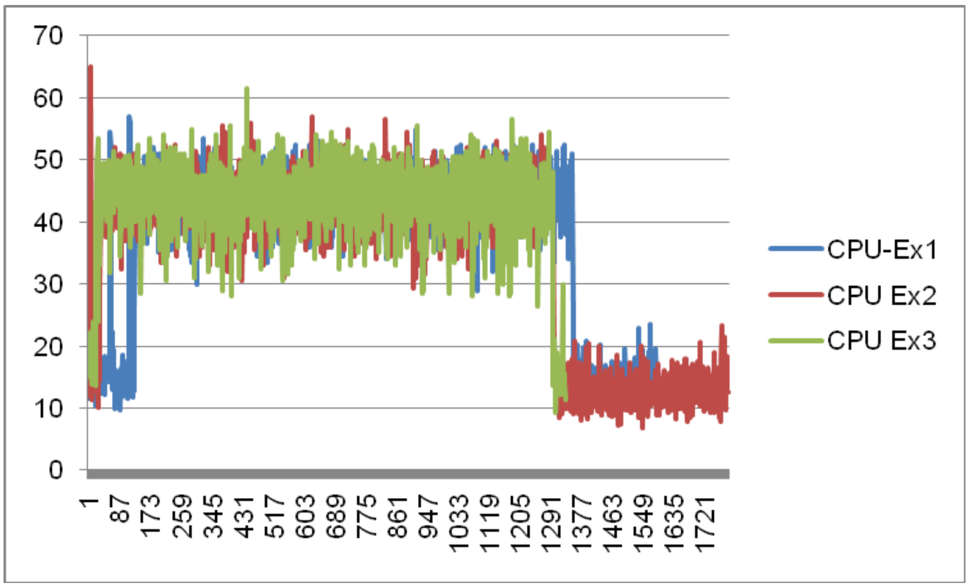


Figure 3-17: Analysis of CPU Utilisation against ICMP Flood

When putting all ICMP results together, it appears that the CPU utilization has not changed much even though the data rate has considerably changed over the time, from 500Kps to 1500KBps. This implies that it could be a while before an attack is detected. Therefore it is important to monitor the traffic and CPU variations concurrently. Further studies are on the way, to identify the specific characteristic of system changes when an attacker is happening. An algorithm is to be defined taking into consideration the CPU utilization, the number of packets sent and received, and the consistency of the changes.

In recent experiments, it has been interesting to note is that the number of packets sent and a number of packets received where almost equal. When ICMP packets are sent, the receiving system will reply whether the system is alive or not. Even when the system is locked by administrator configuration, there is always a reply; this will double the number of packets on the network. ICMP messages can be blocked but this does not reduce the number of packets in a system.

3.8 Snort reaction to UDP Flood

In this experiment, Snort was tested against UDP flood. The objective here is to find out if Snort has the same behaviour or ideas with ICMP flood. Snort was tested against UDP flood using the same conditions as in previous experiments. A common prevention mechanism would apply to both.

3.8.1 Experiment 1

Tools:

- Bonesi, a Bot net and DDOS attack simulator – Bonesi was used to generate packet at an average rate of 500 packets per second, each packet carrying 1024bytes , from up to 50K IPs addresses.
- Snort, an Intrusion detection System
- Ifstat , a tool to collect network statistics
- Sar, a tool to collect CPU utilisation

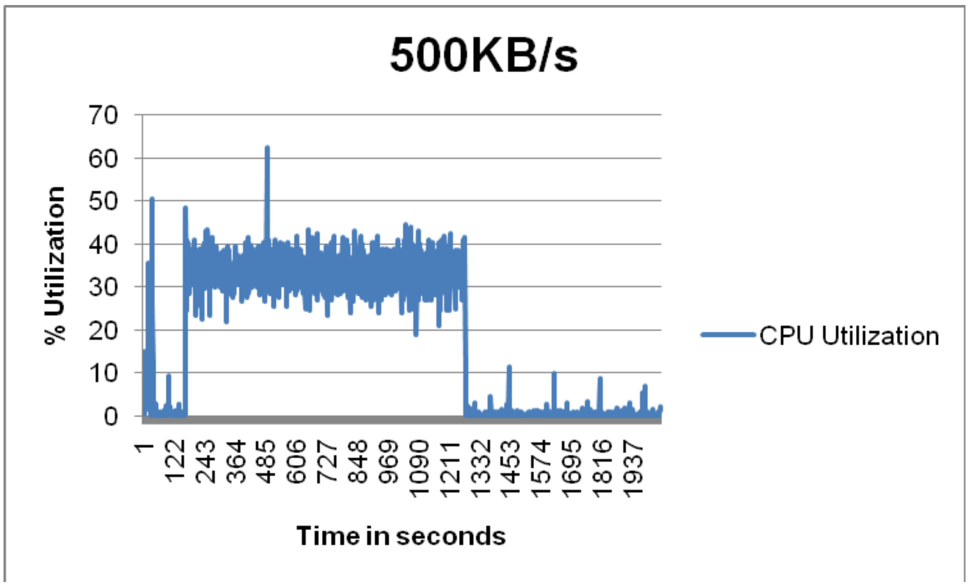


Figure 3-18: CPU utilization when sending 500KB/s UDP packets

In this experiment, the CPU utilization jumped to an average just below 40%. In comparison to Snort behaviour when tested against ICMP traffic under the same conditions; the CPU concentration was above 40%. The number of packets dropped when under ICMP flood was 11% yet with UDP flood, it was 13%. The main difference between the two was the number of outstanding packets waiting to be analysed. When under ICMP flood, the number of outstanding packet was jump 1 yet under UDP flood, the number of packet was 1339. There could be various reasons to justify this behaviour. UDP is more complex than a simple ICMP packet; the number of checks Snort performs for ICMP packets is much less than the number of checked performed by UDP. Over all, the system was in a better state when under UDP flood than he was when under ICMP flood.

Looking at Figure 3-13, throughout the course of the experiment, there was hardly any difference between the number of packets sent and the number of packets received. One would then note a very high number of packets send and received. Yet, looking at Figure 3-19, there is a clear difference between the number of packets sent and the number of packets received. Each ICMP packet sent generates a response whereas UDP packets do not need a response. Each response will cause more traffic hence a higher CPU utilization and a slower performance. This explains further why Snort would handle UDP flood better than ICMP flood.

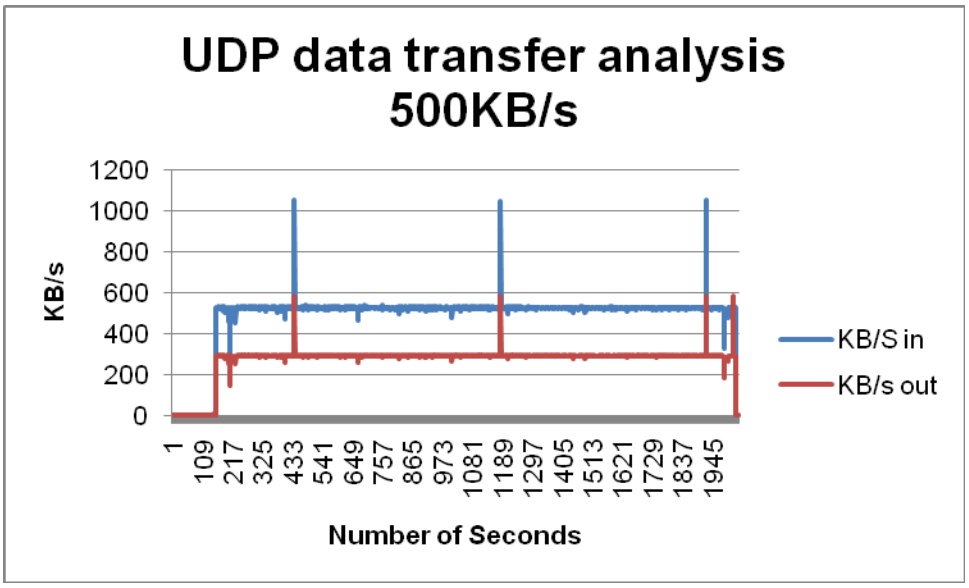


Figure 3-19: UDP data rate transfer analysis 500KB/s

Further analyses were performed comparing Snort behaviour as data rate was increased.

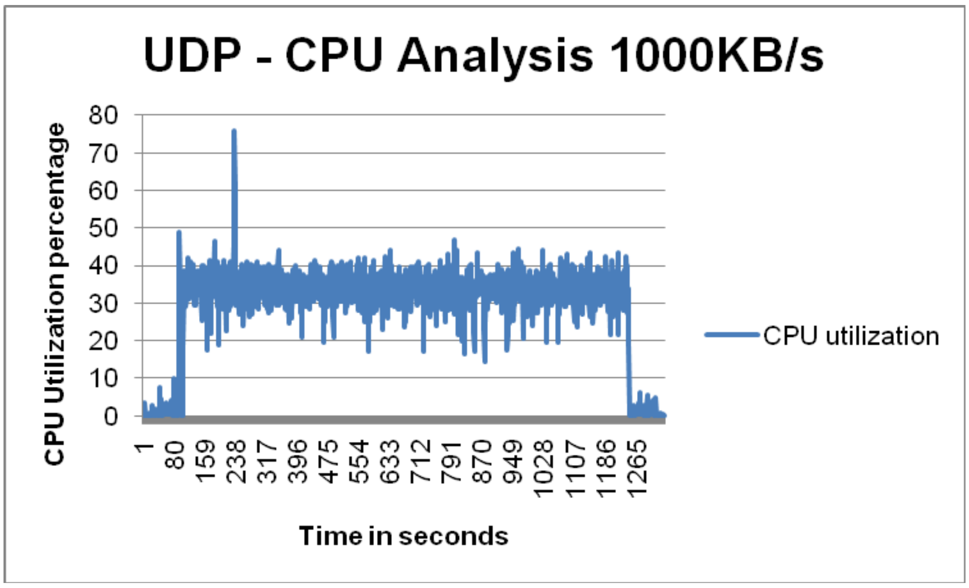


Figure 3-20: CPU utilization 1000KB/s 30bots

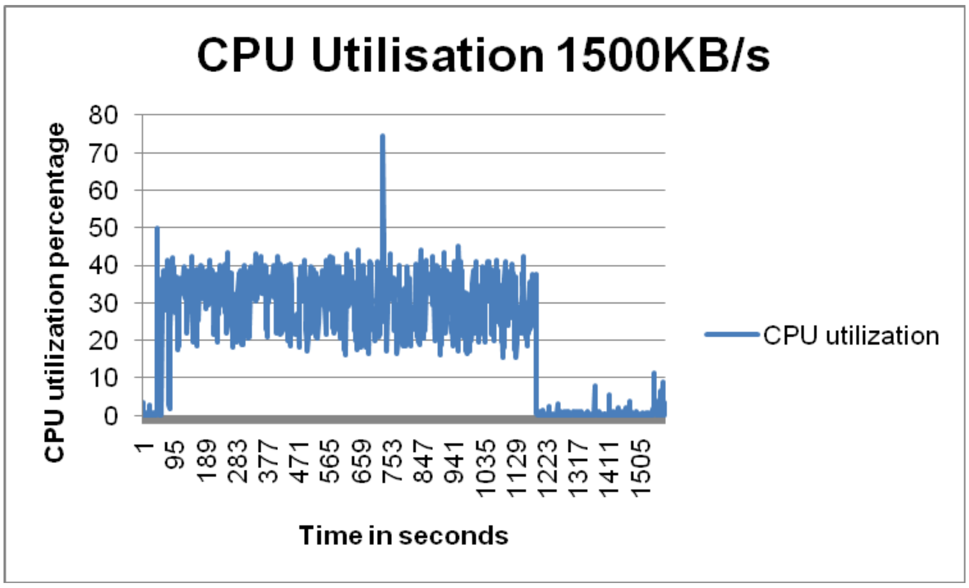


Figure 3-21: CPU Utilization for 1500KB/s

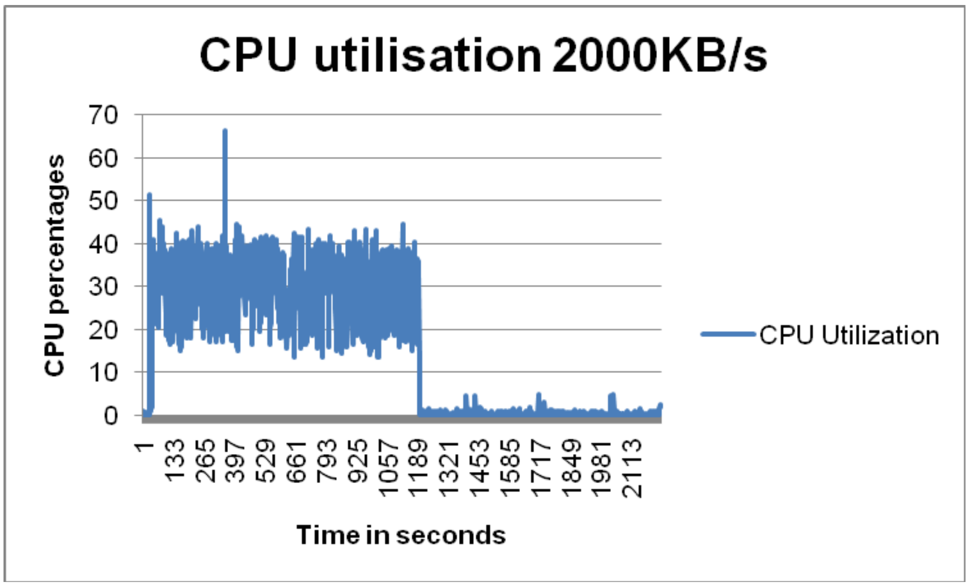


Figure 3-22: CPU utilization data rate = 2000KB/s

Looking at all the results from UDP flood attack, Figure 3-20, Figure 3-21, and Figure 3-22, the CPU utilisation remained very constant. It is important to note that the CPU utilisation remained high for many cycles. Further studies done at a later stage in this research will design and implement a method for detecting flooding back based on the network data rate variation, the CPU utilisation, and the data rate

As the speed of data was increased, the number of packets dropped also increased (Figure 3-23).

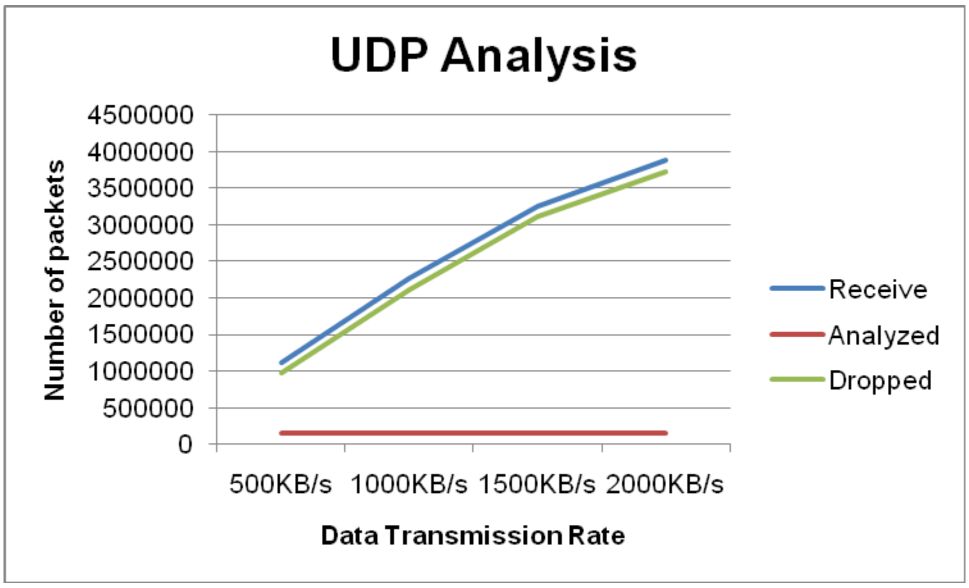


Figure 3-23: UDP - Snort performance analysis

To conclude this experiment, one could look at the variations of the CPU in combination with the number of packets loss and the data rate. DDOS attacks occur when the system under attack is not able to provide services any more to the legitimate users. DDOS can be caused by either a then external element to the system concerned by launching a specific attack; DDOS can also occur by a fault in the system causing the system resources to be too low to provide any service. Either way, an IDS should be able to detect that system resources are low enough and either slow the packets down or take other appropriate action. The architecture proposed later by this search will address this issue.

3.9 Snort reaction to HTTP Flood

Tools

- Bonesi, a Bot net and DDOS attack simulator – Bonesi was used to generate packet at an average rate of 500 packets per second, each packet carrying 1024bytes , from up to 50K IPs addresses.
- Snort, an Intrusion detection System
- Ifstat , a tool to collect network statistics
- Sar, a tool to collect CPU utilisation
- Apache with Joomla installed

The behaviour of Snort is once again analysed when the network is subject to HTTP based DDOS attacks. Looking at Figure 3-24, Snort performance has not been seriously affected by the number of botnets used. As seen on the latter graph, the number of packets analysed was predefined for fair analysis and comparison. Snort managed to analyse over 14% of traffic. Snort performed better in handling HTTP traffic than handling UDP and ICMP traffic. At this stage, there is no clear indicator that the system is under attack. After looking at the internal performances of Snort, the CPU utilisation and the network bandwidth rate were analysed.

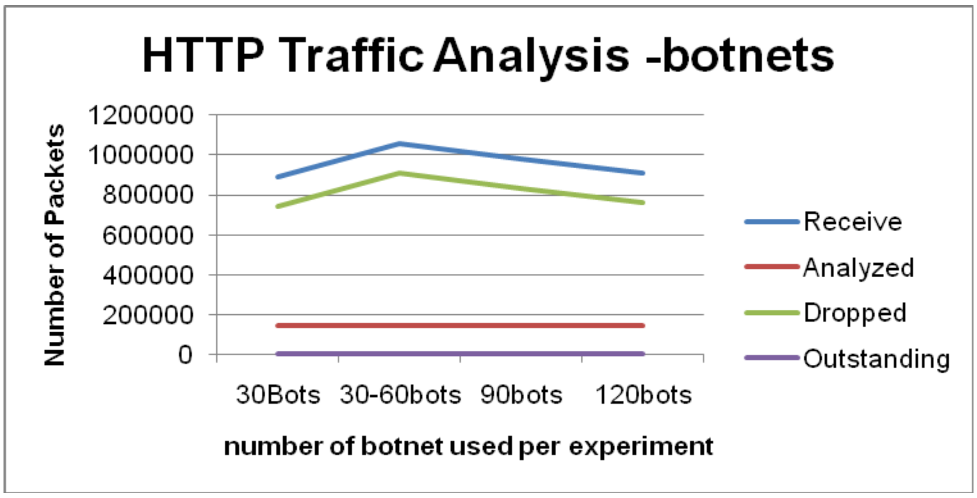


Figure 3-24: HHTP based DDOS attack view by Snort

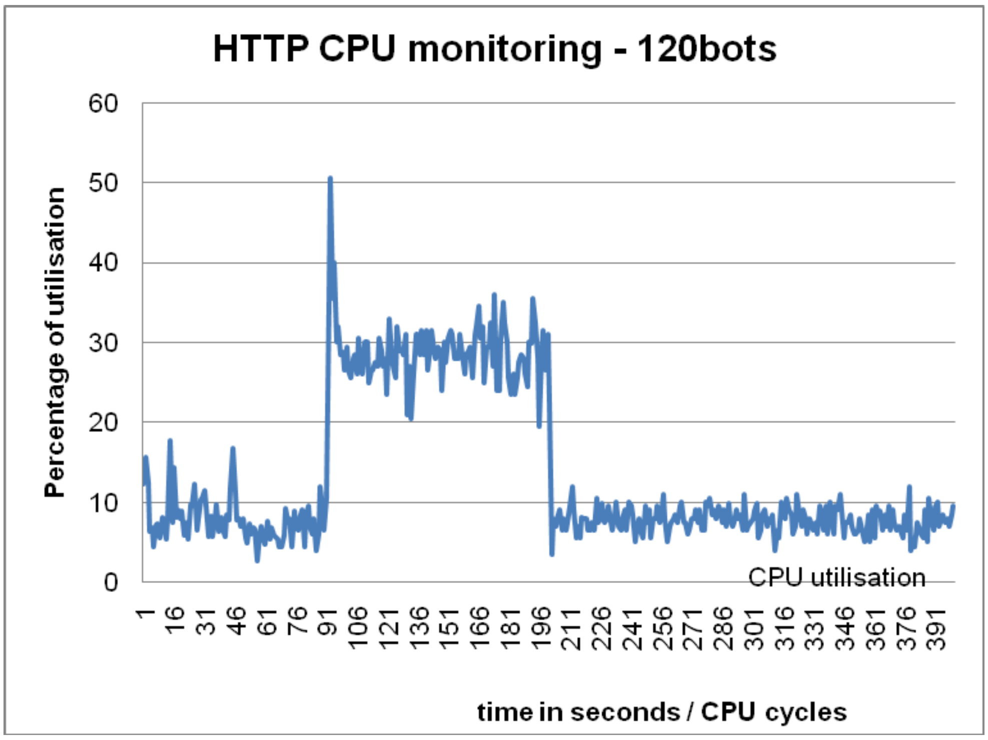


Figure 3-25: CPU Utilisation 120bots

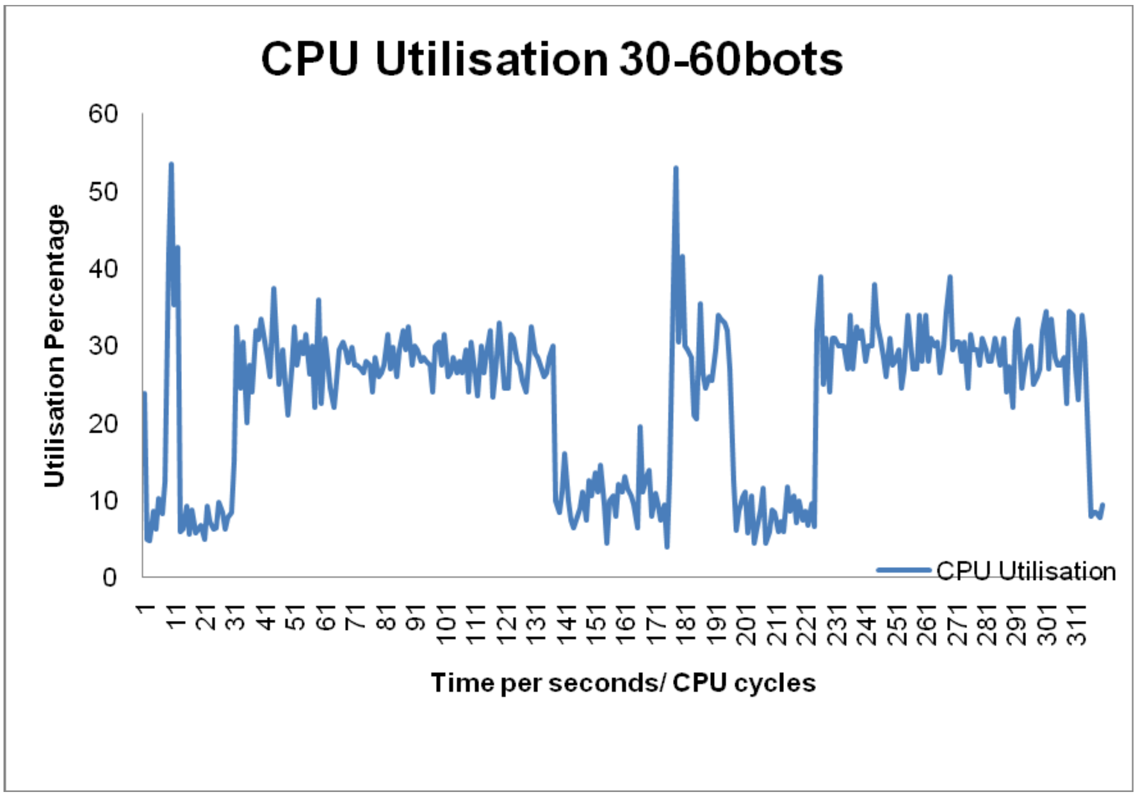


Figure 3-26: CPU utilisation 30-60bots

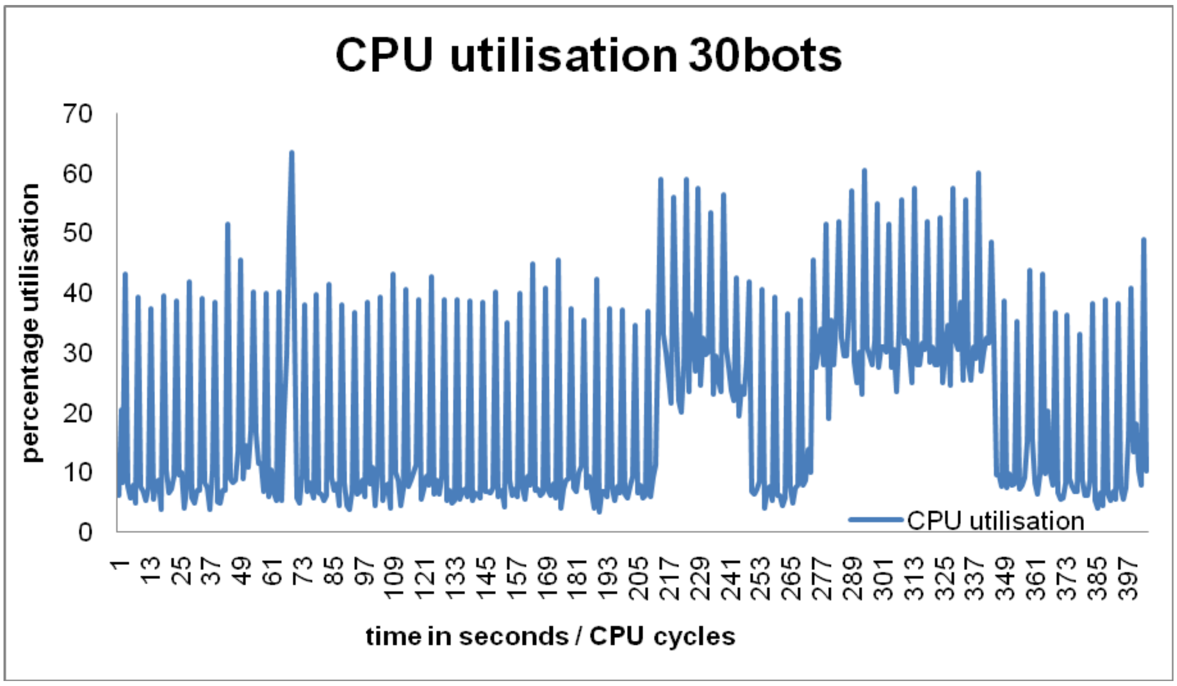


Figure 3-27: CPU Monitoring 30bots

CPU utilisation in the case of HTTP presents different characteristics than those observed when studying Snort under ICMP and UDP flood. In the case of ICMP and UDP, the CPU utilisation raised and remained constant throughout the attack. In this case, there are many variations. For a better view of the CPU utilisation, a zoom on Figure 3-27 was realised.

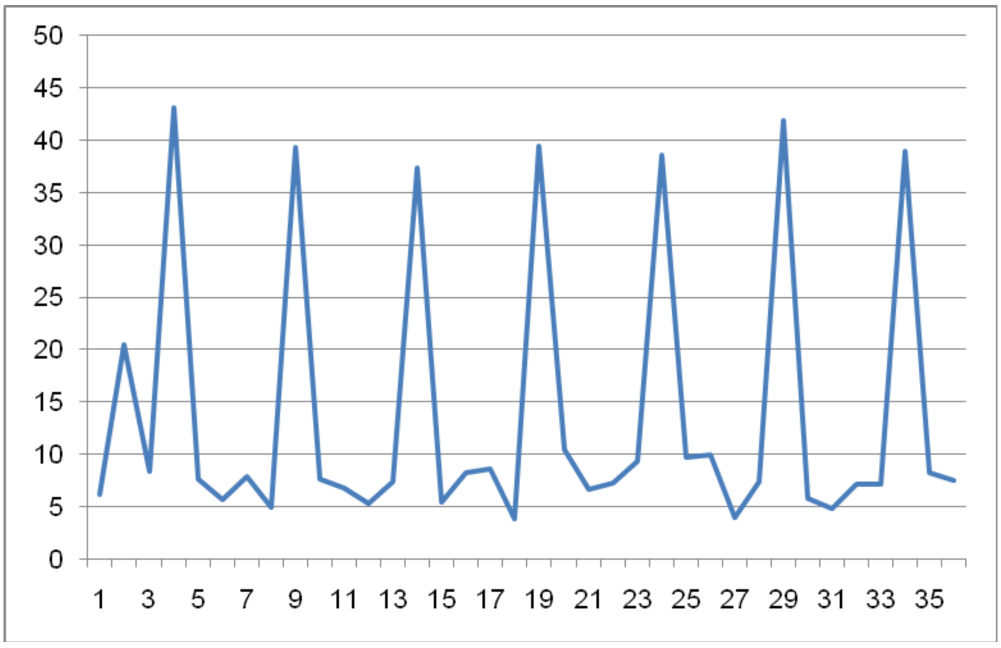


Figure 3-28: Zoom on Figure 3-27

A close look at the zoomed in figure gives the impression of a mathematical sinus function. One could easily think that a pattern is repeating. This could be true and will be subject to mathematical calculation later in this research. At this stage it is very difficult to determine what would signal of an attack. However, the repetition of high peak of the CPU usage over a period of time could be a very good indicator that an attack is taking place. Determining the accuracy of the repetition will be subject to more tests and mathematical design.

When the variation 30-60bots were used, the CPU level remains constant for sometimes before dropping and goes back to the previous high level. A possible indicator of attack here would be the constancy of the CPU level when the system is under attack.

The CPU pattern recorded when under 120bots is similar to the pattern recorded for 30-60 in that when the CPU hit a peak, the value remains constant for a moment before going down

3.10 Snort reaction to multistage attacks

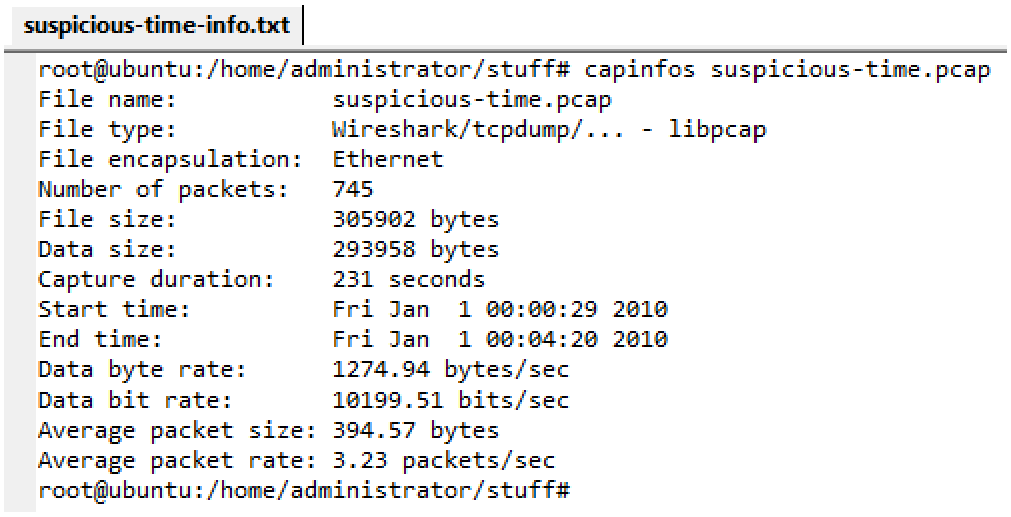
This section will focus on analysing a modern HTTP attack reflecting the type of attacks that are current nowadays.

The analysis in this section will be organised as follow:

- Summary of file information
- Percentage of participants IPs
- Summary of conversations
- Summary of protocols
- in depth analysis

The file used for this analysis was used the challenge 2 of the forensic challenge 2010 – Browser under attack [108].

3.10.1 Summary of file information


suspicious-time-info.txt

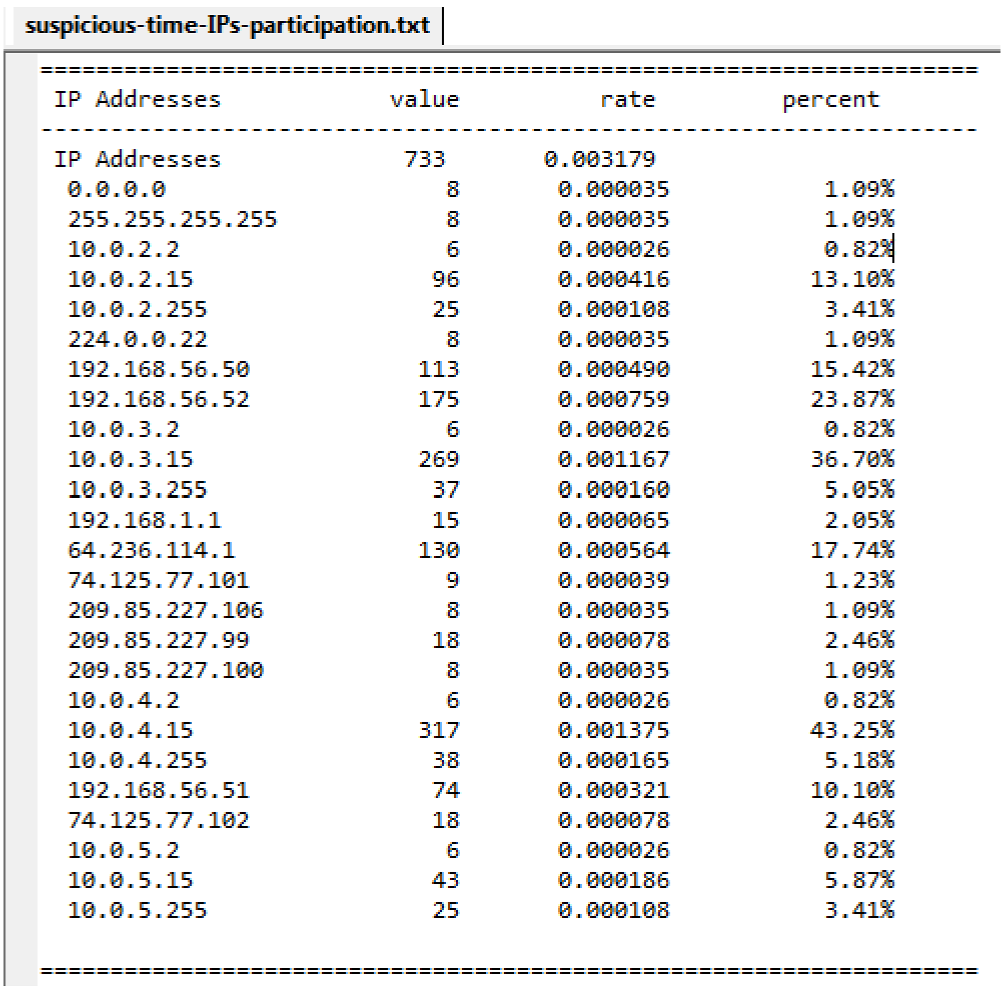
```
root@ubuntu:/home/administrator/stuff# capinfos suspicious-time.pcap
File name:           suspicious-time.pcap
File type:           Wireshark/tcpdump/... - libpcap
File encapsulation:  Ethernet
Number of packets:   745
File size:           305902 bytes
Data size:           293958 bytes
Capture duration:    231 seconds
Start time:          Fri Jan  1 00:00:29 2010
End time:            Fri Jan  1 00:04:20 2010
Data byte rate:      1274.94 bytes/sec
Data bit rate:       10199.51 bits/sec
Average packet size: 394.57 bytes
Average packet rate: 3.23 packets/sec
root@ubuntu:/home/administrator/stuff#
```

Figure 3-29: Suspicious-time file information

This section is purely informative and does not carry any attack hint. However, information such as the type file, the file size the data bit and data byte rate, can give an indication as per what to expect in the file i.e. slow traffic, flood attack, etc.

3.10.2 Percentage of participants IPs

This section gives good indication on the traffic behaviour and the number of IP participant. For instance, a presence of closely related IPs could indicate a scan. In this case, few IPs are above 10% of the overall traffic. It is important to note at this stage that the traffic has been synthetized and foreign IPs have been replace by 192.168.x.x. In this scenario, IPs 192.168.56.52 and 192.168.56.50 are the two external IPs with the most presence in the communication.


suspicious-time-IPs-participation.txt

```
===================================================================
 IP Addresses              value          rate          percent
-------------------------------------------------------------------
 IP Addresses               733       0.003179
  0.0.0.0                     8        0.000035            1.09%
  255.255.255.255             8        0.000035            1.09%
  10.0.2.2                    6        0.000026            0.82%
  10.0.2.15                  96        0.000416           13.10%
  10.0.2.255                 25        0.000108            3.41%
  224.0.0.22                  8        0.000035            1.09%
  192.168.56.50             113        0.000490           15.42%
  192.168.56.52             175        0.000759           23.87%
  10.0.3.2                    6        0.000026            0.82%
  10.0.3.15                 269        0.001167           36.70%
  10.0.3.255                 37        0.000160            5.05%
  192.168.1.1                15        0.000065            2.05%
  64.236.114.1              130        0.000564           17.74%
  74.125.77.101               9        0.000039            1.23%
  209.85.227.106              8        0.000035            1.09%
  209.85.227.99              18        0.000078            2.46%
  209.85.227.100              8        0.000035            1.09%
  10.0.4.2                    6        0.000026            0.82%
  10.0.4.15                 317        0.001375           43.25%
  10.0.4.255                 38        0.000165            5.18%
  192.168.56.51              74        0.000321           10.10%
  74.125.77.102              18        0.000078            2.46%
  10.0.5.2                    6        0.000026            0.82%
  10.0.5.15                  43        0.000186            5.87%
  10.0.5.255                 25        0.000108            3.41%

===================================================================
```

Figure 3-30: IP participant

3.10.3 Summary of conversation

After analysing a trace file, 16 Ethernet conversations were found, 29 IPv4 conversations, 25 TCP conversations, and 15 UDP conversations. Looking further into the conversations, it appears that four different systems in the communication had the same netbios name. However, they appear to be in different subnets. This is a typical setting for virtual machine environment.

```
root@ubuntu:/home/administrator/stuff# tshark -r suspicious-time.pcap | grep
'NB.*20\>' | sed -e 's/<[^>]*>//g' | awk '{print $3,$4,$9}' | sort -u
Running as user "root" and group "root". This could be dangerous.
10.0.2.15                    ->                    8FD12EDD2DC1462
10.0.3.15                    ->                    8FD12EDD2DC1462
```

```
10.0.4.15 -> 8FD12EDD2DC1462
10.0.5.15 -> 8FD12EDD2DC1462
```

The setting used by the malicious user here is simple to build yet carries technicalities that IDS systems are not able to detect. Current IDS systems are not able to build a map of the attacking system or the system being attacked.

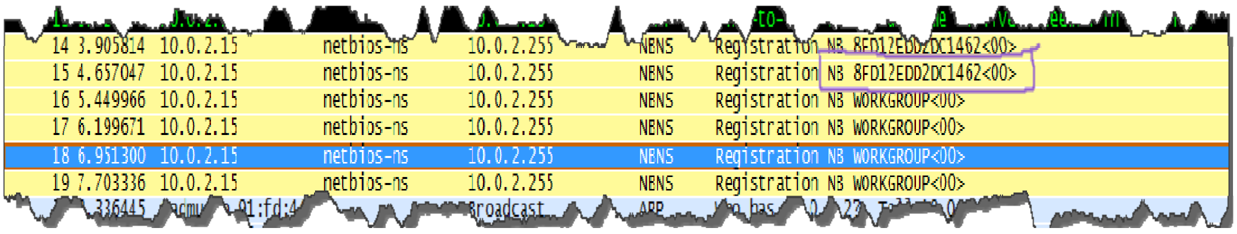
3.10.4 Snort analysis

The current file was analysed using Snort IDS 2.8.5.1 and no alert were reported. There are various reasons why Snort was unable to detect any possible threat or attack in the trace file provided for analysis. The following section, an in depth analysis, will go into details of what is actually taking place in the trace file.

3.10.5 In depth analysis

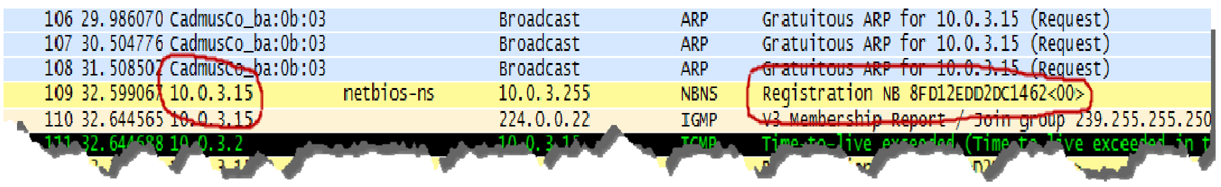
Obfuscating the attack using VMware settings

As shown in the section 3.10.3 above, the attacker makes connection to various systems by using the same computer but with a different virtual machine each time. Being in the Local Network, the IDS will view each connection as a different and separate connection. Even if the IDS was able to detect each separate occurrence of connection, there will be no connection whatsoever between the different connection yet they are all from the same attacker. This technique is more and more used as a way to obfuscate the attack. IDS should be able to detect this as the clear indication was given by the fact that four different IPs had the same name. Figure 3-31 and Figure 3-32 show IP == 10.0.2.15 and IP == 10.0.3.15 registering the same Netbios name.



```
 14 3.905814  10.0.2.15        netbios-ns   10.0.2.255    NBNS   Registration NB 8FD12EDD2DC1462<00>
 15 4.657047  10.0.2.15        netbios-ns   10.0.2.255    NBNS   Registration NB 8FD12EDD2DC1462<00>
 16 5.449966  10.0.2.15        netbios-ns   10.0.2.255    NBNS   Registration NB WORKGROUP<00>
 17 6.199671  10.0.2.15        netbios-ns   10.0.2.255    NBNS   Registration NB WORKGROUP<00>
 18 6.951300  10.0.2.15        netbios-ns   10.0.2.255    NBNS   Registration NB WORKGROUP<00>
 19 7.703336  10.0.2.15        netbios-ns   10.0.2.255    NBNS   Registration NB WORKGROUP<00>
```

Figure 3-31: IP 10.0.2.15 registration



```
106 29.986070 CadmusCo_ba:0b:03               Broadcast     ARP    Gratuitous ARP for 10.0.3.15 (Request)
107 30.504776 CadmusCo_ba:0b:03               Broadcast     ARP    Gratuitous ARP for 10.0.3.15 (Request)
108 31.508502 CadmusCo_ba:0b:03               Broadcast     ARP    Gratuitous ARP for 10.0.3.15 (Request)
109 32.599067 10.0.3.15          netbios-ns   10.0.3.255    NBNS   Registration NB 8FD12EDD2DC1462<00>
110 32.644565 10.0.3.15                       224.0.0.22    IGMP   V3 Membership Report / Join group 239.255.255.250
```

Figure 3-32: IP 10.0.3.15 Registration

Attack scenario 1

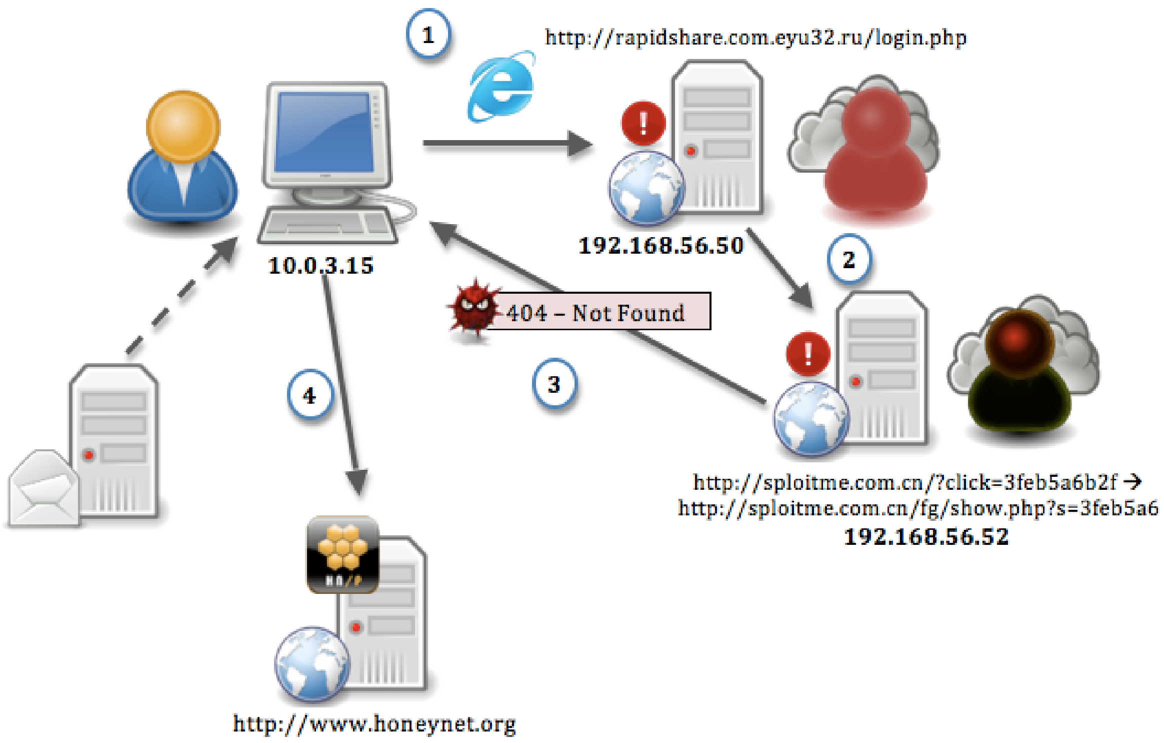

1
http://rapidshare.com.eyu32.ru/login.php
192.168.56.50
10.0.3.15
2
404 – Not Found
3
4
http://sploitme.com.cn/?click=3feb5a6b2f →
http://sploitme.com.cn/fg/show.php?s=3feb5a6
192.168.56.52
http://www.honeynet.org

Figure 3-33: Attack Scenario 1

➢ In this scenario, the attacker use one of the virtual machine to connect to rapidshare.eyu32.ru/login.php using Firefox. Snort did not complain as there is nothing visible or apparent that appear illegitimate. The only way for Snort to catch this action was to have the URL specified as a string to be searched.

```
GET /login.php HTTP/1.1
Host: rapidshare.com.eyu32.ru
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.3) Gecko/20090824 Firefox/3.5.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

- The page is then returned successfully, however, with a non-readable text. Snort does not provide any mechanism to read non-trivial text. Any encrypted text is generally ignored by Snort hence considered as safe. This is yet a popular way to hide attacks.

```
...........Vmo.6..._.j.l!..[......l....5)......-...RT.7...Q.c.C.
~...{.Ew.....:.......ys.......!........n...5e88D7..%S..8..._,d..w..<..j.:HTb...P.]..ed.J...[....?.r.
'.l#m.aJq.w:..B.TF..X..:.....;..Y..t..R.b.z
...Tad.]..b..5....ro6.Z.4..R.NyF....-.....u...&..2H.+VH.,.J..h.R.,....&ei.T.Ed../i..e.......[...a.!..._..O..<.7..>a........>.^-..^..p..
......j......)...[...UA....>....O......9q.%U..U.O.....&......;.s..........k%!......B6.......9!...j....a...g..@Rp......o..s.NY.......t>F.....f=]E....[.
.0~|.8.3}K.......0%..a.C......x4..k.....D........^......%..J...~..^....>..;..=....=..o?..}d....O..>.X.{....3.....3...'..oW         }..C{a.......c....
.....:..C.E..9........%P'.[....9Xc.....r.....+/.=..g............s...?............z4..W:...H.C...s...P./t.J..up....O....N...L.%STW.-
..T...R...H...Gz<..X..}..;..g....2....`.....5du..[...ZK,........(d....D..k.R.._..'.4t.D.d.!....Q".H..J.  ...`|.v.8.m.{V...4.0T$..!.<Zx.e...b..r..
.OV|%i...}.Z7|...&...W4..q@..Q.5'../Y.g.......X...5.U.E.^l$                                            .k....@.F..?sV....o............
:..p..,.DqY.|.m..%..?.
..<.......X..ux.hw......S......i...ix+.q...1.................47...~-...MT..m"....)Z..\@.V-
.u%...i.9..]._6.......D..v....Y.!.`Vh...f..N....oO.zg.$.
....C.0.KG{..r.........%s.lW..?5-.R.........q.Y...sY.B..b...W..Q/Tg.p}.E.~..TX0=...+....WxnF.P..@..|..u1..     ......R..9.......&j........
|.c.%.0..=.N.eV.._F...[.............K..Dn..IS.[.e5....z..^...N..V...+....P.o)..
....U.N.S...'S*..&.D.f..1OH.h.j....H.(d.@JJ*..6 7.............|...........k.R..e.nA..n..A1..65.....<>.h.'..?........
```

Using a popular and free tool widely available [109], the obfuscated code was made clear, readable and returned

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>
35?String.fromCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--
)r[e(c)]=k[c]||e(c);k=[function(e){return
r[e]}];e=function(){return'\\w+'};c=1};while(c--)if(k[c])p=p.replace(new
RegExp('\\b'+e(c)+'\\b','g'),k[c]);return            p}('..........y.6...x.j.l!..[......I....5)......-
...A.7...Q.c.C.~...{.G....:.......M.......!........n...10..%S..8...x,d..w..<..j.:11...P.]..12.J.
..[....?.r.\'.I#m.13.w:..B.14..X..:.....;..Y..t..R.b.z\n...15.]..b..5....16.Z.4..R.17....-
.....u...&..18.+19.,.J..h.R.,....&1a.T.1b../i..e.......[...a.!...x..O..<.7..>a........>.^-
..^..p..\n......j......)...[...1c....>....O......1d.%U..U.O.....&......;.s..........k%!......1e.......
9!...j....a...g..@1f......o..s.1g.......t>F.....f=]E....[..0~|.8.3}K.......0%..a.C......1h..k.....
D........^......%..J...~..^....>..;..=....=..o?..}d....O..>.X.{....3.....3...\'..1i}..C{a.......c......
...:..C.E..9........%P\'.[....1j.....r.....+/.=..g...........s...?...........1k..W:...H.C...s...P./t.
J..1l....O....N...L.%1m.-
..T...R...H...1n<..X..}..;..g....2....`.....1o..[...1p,........(d....D..k.R..x..\'.1q.D.d.!....Q".
H..J.
...`|.v.8.m.{V...4.1r$..!.<1s.e...b..r..\n.1t|%i...}.1u|...&...1v..q@..Q.5\'../Y.g.......X...
5.U.E.^I$
.k....@.F..?1w....o............\n:..p..,.1x.|.m..%..?.\n..<.......X..1y.1z......S......i...1A+.q
...1................1B...~-...1C..m"....)Z..\\@.V-
.u%...i.9..].1D.......D..v....Y.!.`1E...f..N....1F.1G.$.....C.0.1H{..r.........%s.1I..?5-
.R.........q.Y...1J.B..b...W..Q/1K.p}.E.~..1L=...+....1M.P..@..|..1N........R..9.......&j..
......|.c.%.0..=.N.e\\/..1O...[............K..1P..1Q.[.1R....z..^...N..V...+....P.o)......U.N.
S...\'S*..&.D.f..1S.h.j....H.(d.@1T*..6
7............|..........k.R..e.1U..n..1V..1W.....<>.h.\'..?........',62,121,'|||||||||||||||||||||||||||
|||||_|Vmo||RT||||||Ew||||||ys|||||||||||||5e88D7|HTb|ed|aJq|TF|Tad|ro6|NyF|2H|VH|e
i|Ed|UA|9q|B6|Rp|NY|x4|oW|9Xc|z4|up|STW|Gz|5du|ZK|4t|0T|Zx|OV|Z7|W4|sV|
DqY|ux|hw|ix|47|MT|_6|Vh|oO|zg|KG|lW|sY|Tg|TX0|WxnF|u1|_F|Dn|IS|e5|1OH|
JJ|nA|A1|65'.split('|'),0,{}))
```

The latter even though not completely clear, suggests that the malicious user was trying to hide some code that could have been detected by the IDS.

A complete DE obfuscation of the code reveals that the attacker was using iframe to hide another link with more malicious code

```
<iframe      src="http://sploitme.com.cn/?click=3feb5a6b2f"width=1      height=1
style="visibility: hidden"></iframe>
```

- The successful page then redirects the malicious user to another page [HTTP code 304]

  Redirecting a page to another page is a normal behaviour in TCP/IP communication. However, in this case the intention was malicious. Snort does not analyse traffic with much depth to actually see the intention behind actions. Snort would have been able to look at this behaviour if Snort supported attack tree and if the attack tree was defined. Obfuscating HTML code becomes more and more attractive as this would bypass most security system. It is therefore important to have a system that is capable of analysing obfuscated portions of code.

```
HTTP/1.1 304 Not Modified
Date: Tue, 02 Feb 2010 19:05:12 GMT
Server: Apache/2.2.9 (Ubuntu) PHP/5.2.6-2ubuntu4.6 with Suhosin-Patch
Connection: Keep-Alive
Keep-Alive: timeout=15, max=99
ETag: "5e472-fef-47ea19070f940"
```

As a result of the redirection, the server returns the code [HTTP 404] which normally would mean that the page was not found. However, the page even thought a normal looking error page a further analysis will look at the irregular non-readable section of the page.

```
..........MP.j.0...+.9..h].
-.A.;$.&...=*...........P.e`fgv..w.n.|.%...
....,a6G...
.h..$#)b....*.:2.$..x....i.[.aeB/(....d.{#.c....D...5J..?A:/.......ugz....A.C.1......'.YZBq....\.+.co....
.d....}.}x.z].s...,LRN.p.^.WP..~^s.E6.....A.....3'"..)#6@m.......Xr....ol~..J..Q...
```

When deofuscated by the free online tool identified earlier the script becomes

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?Stri
ng.fromCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--
)r[e(c)]=k[c]||e(c);k=[function(e){return        r[e]}];e=function(){return'\\w+'};c=1};while(c--
)if(k[c])p=p.replace(new  RegExp('\\b'+e(c)+'\\b','g'),k[c]);return  p}('..........4.j.0...+.9..h].-
.A.;$.&...=*...........5.e`6..w.n.|.%.......,7....h..$#)b....*.:2.$..x....i.[.8/(....d.{#.c....a...f..?A:/....
...g....A.C.1......\'.k....\\.+.l.....d....}.}x.z].s...,o.p.^.q..~^s.r.....A.....3\'"..)t@m.......u....v~..y..
B...',39,39,'||||MP|P|fgv|a6G|aeB||D|||||5J|ugz||||YZBq|co|||LRN||WP|E6||undefined|Xr|ol|
||J|||Q|'.split('|'),0,{}))
```

Scenario 2

In this scenario, start another virtual machine as shown in Figure 3-32. Given that Netbios name are unique per network. This suggests that the attacker turned off the first virtual machine to start a second. The objective of such actions is to deceive security systems in a way that even if individual attacks are traced, it will be very difficult to link these different attacks as coming from the source.

Similar actions to scenario 1 are repeated.

The attacker connected to rapidshare.com.eyu32.ru

```
GET /login.php HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: rapidshare.com.eyu32.ru
Connection: Keep-Alive
```

The attacker gets a response with a redirection to another page. Just as in scenario 1, the script behind the pages was obfuscated.

Under the page login downloaded as seen above, the following script was embedded

```
...........Vmo.6..._.j.l!..[......l....5)......-...RT.7...Q.c.C.
~...{.Ew.....:.......ys.......!........n...5e88D7..%S..8..._,d..w..<..j.:HTb...P.]..ed.J...[....?.r.
'.l#m.aJq.w:..B.TF..X..:.....;..Y..t..R.b.z
...Tad.]..b..5....ro6.Z.4..R.NyF....-
.....u...&..2H.+VH.,.J..h.R.,....&ei.T.Ed../i..e.......[...a.!..._..O..<.7..>a........>.^-..^..p..
......j......)...[...UA....>....O......9q.%U..U.O.....&......;.s..........k%!......B6.......9!...j....a...g..@
Rp......o..s.NY.......t>F.....f=]E....[..0~|.8.3}K.......0%..a.C......x4..k.....D........^......%..J...~..
^....>..;..=....=..o?..}d....O..>.X.{....3.....3...'..oW                                  }..C{a.......c....
.....:..C.E..9........%P'.[....9Xc.....r.....+/.=..g...........s...?...........z4..W:...H.C...s...P./t.J..up.
...O....N...L.%STW.-
..T...R...H...Gz<..X..}..;..g....2....`.....5du..[...ZK,........(d....D..k.R.._..'.4t.D.d.!....Q".H..J.
...`|.v.8.m.{V...4.0T$..!.<Zx.e...b..r..
.OV|%i...}.Z7|...&...W4..q@..Q.5'../Y.g.......X...5.U.E.^l$ .k....@.F..?sV....o...........
:..p..,.DqY.|.m..%..?.
..<.......X..ux.hw......S......i...ix+.q...1................47...~-...MT..m"....)Z..\@.V-
.u%...i.9..]._6.......D..v....Y.!.`Vh...f..N....oO.zg.$.
....C.0.KG{..r.........%s.lW..?5-
.R.........q.Y...sY.B..b...W..Q/Tg.p}.E.~..TX0=...+....WxnF.P..@..|..u1..
......R..9.......&j........
|.c.%.0..=.N.e\/.._F...[............K..Dn..lS.[.e5....z..^...N..V...+....P.o)..
```

```
....U.N.S...'S*..&.D.f..1OH.h.j....H.(d.@JJ*..6
7............|...........k.R..e.nA..n..A1..65.....<>.h.'..?........
```

Using the same online tool, the obfuscated code was made more readable.

On successful download of the login page, the user was redirected to hidden page using iframes.

3.11 Analysis:

In this attack commonly known as "the browser under attack", the attacker uses four almost identical scenarios almost identical to deceive computer security systems. The attacker used four different techniques to bypass Snort security checks.

Javascript obfuscation: currently Snort does not provide any analysis method to prevent against attacked embedded into obfuscated Javascript. Introducing a deobfuscation features for IDS would be very possible as they are already existing codes that are commonly used to obfuscate and deobfuscate Javascript.

A pretended error page – 404 error code: Pages usually display 404 when the page is not found. But in this case, the 404 page is the page intended as it contains hidden malicious code. Some security systems check for error code 404 to trigger certain alerts based on a predefined threshold. In this case, the page will not generate any alert as the return code number is 200 which indicates that the intended page has been successfully downloaded.

Iframe: more and more, iframes are becoming a serious security concern as attackers use them to hide malicious code

Content compression: content compression is becoming more and more of a security challenge. The great difficulty in this is that most files sent over the internet are compressed in order to limit the amount data sent and increase the speed at which the date is sent. However, malicious users take advantage of file compression to upload and download their script which can then be interpreted by the received end.

At the end of this study, is it safe to conclude that Snort is not adapted to detect the latest attacks. This is an indication that strictly string matching for security systems nowadays is very limited. Later in this research work, an alternative design is suggest for a current and revolutionary IDS.

Reconfiguration vs. new plug-ins

In order to change Snort behaviour, a common practice is to reconfigure Snort. However, Snort reconfiguration is limited to assigning values to existing variables [110]. They are few areas where Snort would accept new variables. When defining group of IPs, Snort is flexible enough to accept new variable such as New_IP_Group = <list of IPs>.

However, Snort would not recognise values that are not predefined in most cases. If Snort has not been compiled with an option, any variable related to that option will generate an error. For instance, if Snort is not configured with the option to support database any attempt to connect to database will fail and generate an error that will prevent Snort from running. Moreover, even if database access was compiled, specific database need to be specified. For instance the configuration line specifies MySql database access.

```
output database: log, mysql, user=snort password=56y7@po#90 dbname=snort
host=localhost
```

Any change of behaviour of Snort that cannot be done by modifying existing parameters need to be compiled before it can be used. This implies that the source code to support the addition needs to be provided. Whenever the source code is provided for a new option that Snort did not support, a new plug-in is then written. To date, many plug-in have been written to extend Snort functionalities [111]. For instance, SnortSam is a Snort plug-in that enables the communication between Snort and different firewalls such as Checkpoint Firewall, Cisco PIX firewalls, Cisco Routers (using ACL's or Null-Routes), Former Netscreen, now Juniper firewalls, IP Filter (ipf) [112]. When an IP has been flagged as attacker, Snortsam will then send the IP to the firewall with instructions to block the given IP.

The solution proposed to the various problems identified during our investigations will require to write different plug-in. One plug-in for rules optimisation, one plug-in for DDOS detection, one plug-in for DDOS mitigation, one plug-in for detecting multistage attacks, one plug-in for mitigating multistage attacks, one plug-in for source code analysis, etc. Instead, the author decided to produce a new IDS architecture for a better integration of these new components.

Conclusion

After various experiments, Snort does not handle well traffic any flood situation. Up to 90% of traffic can be dropped whilst the CPU becomes very high and remains so for many cycles. At the same time, the data rate increased. There is a clear link between the variations of the CPU, the variations of data rate and the increased number of packets drop. This link will be used to build an attack indicator raising the IDS state to attack in progress. In addition, there were clear indications that an attack was in progress but Snort failed to notice them. For example, Snort received the same payload from over a thousand IPs; the system generated over a thousand ICMP response based on the same port number. The problems that will be addressed later when building the new IDS architecture are:

- To detect increased change in data rates
- To detect Increased CPU utilisation
- To detect increased packets drop
- To detect regular pattern such as repeated payload from multiple host
- To provide a way to analyse obfuscated Javascript
- To provide a way to analyse obfuscated HTML
- To provide multiple encoding system

4 Modelling Multistage attacks for Intrusion Detection System

"DDoS is a threat that must be included in all risk mitigation plans for any company with critical online services and applications". By Richard Stiennon

Introduction

Mitigating today's attacks has become a very serious challenge for Internet based businesses and services. Recently, hackers have developed systems that allow them to compromise and infect computers and then put the later computers into an army of computer ready to obey commands from a master computer. These armies are referred as botnet. Botnets are generally used to send SPAMS or to launch Denial of Service Attacks. Also, when computers have been compromised they are subject to various attacks as they are controlled remotely. The malicious user having control over the computers could decide to perform various actions listed as but not limited to installing key loggers, installing worms, viruses, destroying data, copying data.

In this chapter, presents our understanding of multistage attacks based on real live traffic capture. The author will then use Snort to perform an offline analysis of our trace file and discuss the ability of Snort to detecting multistage attacks. The author presents a new model that will help detecting and mitigating multistage attacks. The model is very extensible and attacker tree can be used to extend the model.

4.1 Multistage attacks

Over the years, various solutions have been proposed to resolve cyber-attacks. However, it is becoming increasingly difficult to distinguish legitimate traffic from illegal traffic. Often, some network communications that are not considered to be problematic are actually crucial to the attackers. These steps are generally ignored by IDS as they do not violate any rules. In this section, the focus will be on analysing captured traffic from live network and honeypot. One of the highlight is the steps that are generally ignored by IDS and most security systems.

4.2 Analysis strategy

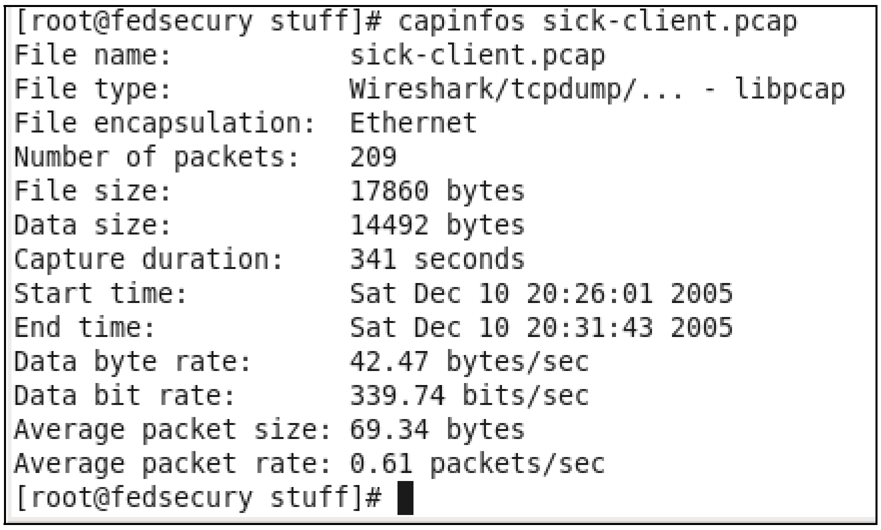
As a general strategy, important statistics of each trace file analysed are obtained to have a quick general overview of what could have been going on during the capture. The steps taken would be, whenever applicable:

- General file statistics: trace file statistics
- List of IPs participants and their percentage of participation
- Operating system involved
- Summary of TCP transactions
- Summary of conversation
- Extract any file present in the trace file
- In-depth analysis

4.3 Scenario Alpha

This scenario is about a capture that was made as the attacker was trying to register computers to its bot army.

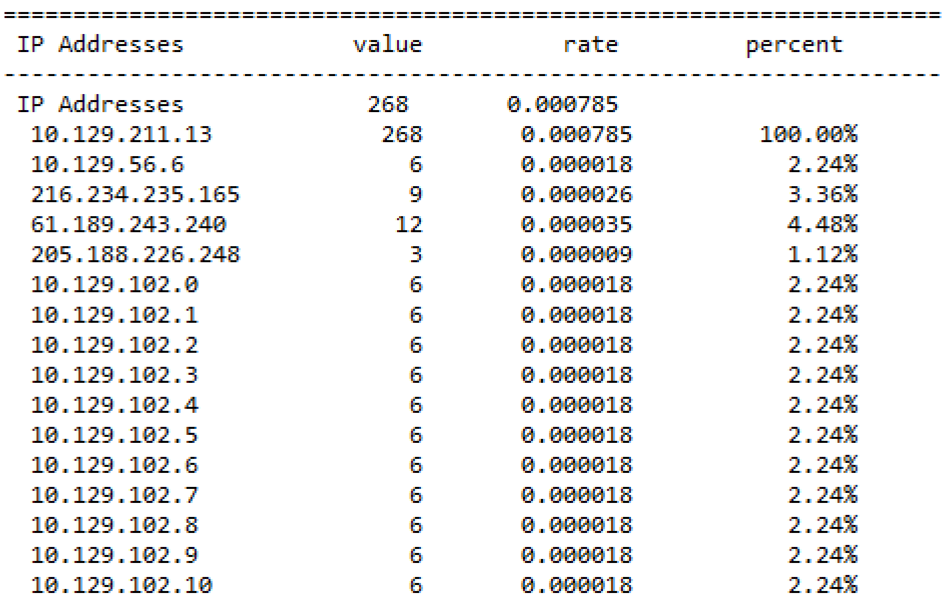
Trace file statistics summary



```
[root@fedsecury stuff]# capinfos sick-client.pcap
File name:             sick-client.pcap
File type:             Wireshark/tcpdump/... - libpcap
File encapsulation:    Ethernet
Number of packets:     209
File size:             17860 bytes
Data size:             14492 bytes
Capture duration:      341 seconds
Start time:            Sat Dec 10 20:26:01 2005
End time:              Sat Dec 10 20:31:43 2005
Data byte rate:        42.47 bytes/sec
Data bit rate:         339.74 bits/sec
Average packet size: 69.34 bytes
Average packet rate: 0.61 packets/sec
[root@fedsecury stuff]# █
```

Figure 4-1: Sick-client.pcap file information

4.3.1 Trace file analysis

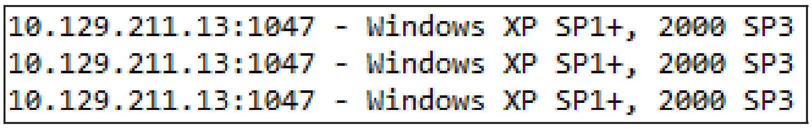


```
==================================================================
 IP Addresses              value          rate          percent
------------------------------------------------------------------
 IP Addresses               268       0.000785
  10.129.211.13             268        0.000785          100.00%
  10.129.56.6                 6        0.000018            2.24%
  216.234.235.165             9        0.000026            3.36%
  61.189.243.240             12        0.000035            4.48%
  205.188.226.248             3        0.000009            1.12%
  10.129.102.0                6        0.000018            2.24%
  10.129.102.1                6        0.000018            2.24%
  10.129.102.2                6        0.000018            2.24%
  10.129.102.3                6        0.000018            2.24%
  10.129.102.4                6        0.000018            2.24%
  10.129.102.5                6        0.000018            2.24%
  10.129.102.6                6        0.000018            2.24%
  10.129.102.7                6        0.000018            2.24%
  10.129.102.8                6        0.000018            2.24%
  10.129.102.9                6        0.000018            2.24%
  10.129.102.10               6        0.000018            2.24%
```

Table 3: List of IP participants

A quick look at the Table 3 shows a succession of IP address. This could indicate a scan. Also, one of the IPs 10.129.211.13 is involved in every single conversation. This can be seen by its participation rate of 100%.

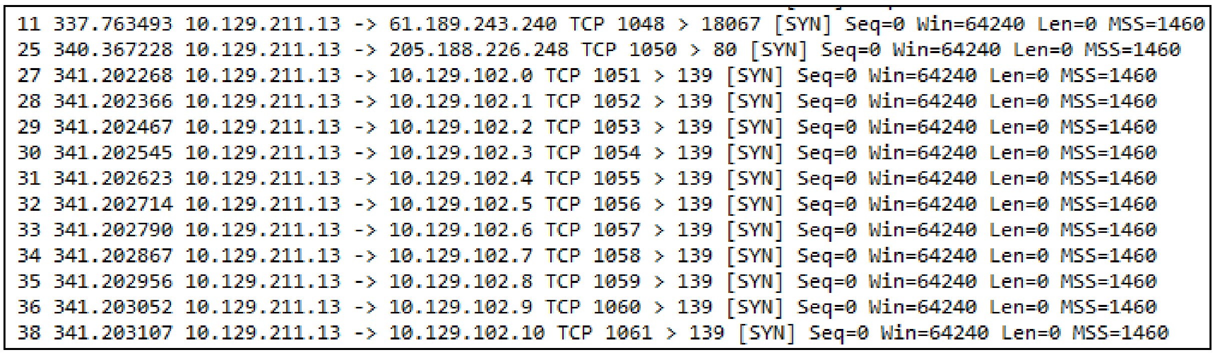
Looking at the OS, it appears that the attacker is a Windows machine



```
10.129.211.13:1047 - Windows XP SP1+, 2000 SP3
10.129.211.13:1047 - Windows XP SP1+, 2000 SP3
10.129.211.13:1047 - Windows XP SP1+, 2000 SP3
```

Figure 4-2: List of Operating System

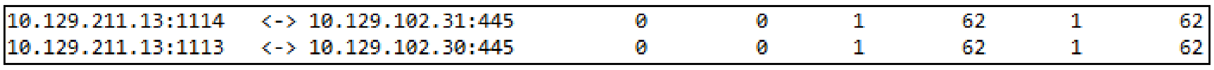
Looking at more indications of attacks, there are many connection initiation as shown Figure 4-3[Seq = 0, Len = 0]. This could indicate a session flooding or a TCP scan. Further analysis will give us more details on the exact nature of the activity.



```
11 337.763493 10.129.211.13 -> 61.189.243.240 TCP 1048 > 18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
25 340.367228 10.129.211.13 -> 205.188.226.248 TCP 1050 > 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
27 341.202268 10.129.211.13 -> 10.129.102.0 TCP 1051 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
28 341.202366 10.129.211.13 -> 10.129.102.1 TCP 1052 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
29 341.202467 10.129.211.13 -> 10.129.102.2 TCP 1053 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
30 341.202545 10.129.211.13 -> 10.129.102.3 TCP 1054 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
31 341.202623 10.129.211.13 -> 10.129.102.4 TCP 1055 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
32 341.202714 10.129.211.13 -> 10.129.102.5 TCP 1056 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
33 341.202790 10.129.211.13 -> 10.129.102.6 TCP 1057 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
34 341.202867 10.129.211.13 -> 10.129.102.7 TCP 1058 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
35 341.202956 10.129.211.13 -> 10.129.102.8 TCP 1059 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
36 341.203052 10.129.211.13 -> 10.129.102.9 TCP 1060 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
38 341.203107 10.129.211.13 -> 10.129.102.10 TCP 1061 > 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
```

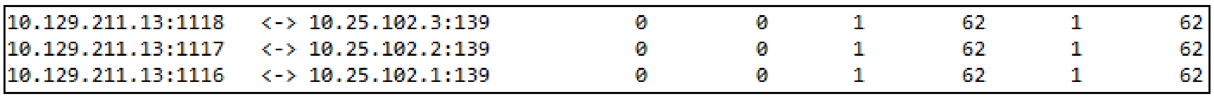
Figure 4-3: Open connections

Another quick command revealed that there have been 130 conversations in total. These conversation were observed around port 445 (Figure 4-4), port 139 (Figure 4-5)



```
10.129.211.13:1114   <-> 10.129.102.31:445        0        0       1        62       1        62
10.129.211.13:1113   <-> 10.129.102.30:445        0        0       1        62       1        62
```

Figure 4-4: port 445 usage



```
10.129.211.13:1118   <-> 10.25.102.3:139          0        0       1        62       1        62
10.129.211.13:1117   <-> 10.25.102.2:139          0        0       1        62       1        62
10.129.211.13:1116   <-> 10.25.102.1:139          0        0       1        62       1        62
```

Figure 4-5: port 139 usage

Given the number of conversation on port 445 and port 139, there is a strong indication that the host was vulnerable.

Further analysis of the trace file indicates an apparent problem as there are many repetitions of the same message. This could be fine if it was a usual TCP or UDP connection but many ICMP messages would indicate something unusual. Table 4 shows a good number of repeated ICMP messages. As shown on the table, one could observe the same ICMP message originating from different IPs and directed at a single IP. This generally indicates the sign of a Scan.

Packet No	Timestamp	Source IP	Destination IP	Protocol	Other info
175	341.221903	10.129.102.20	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
176	341.222633	10.129.102.21	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
177	341.223361	10.129.102.22	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
178	341.223848	10.129.102.23	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
179	341.224578	10.129.102.24	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
180	341.225064	10.129.102.25	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
181	341.225797	10.129.102.26	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
182	341.22628	10.129.102.27	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
183	341.22701	10.129.102.28	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
184	341.227739	10.129.102.29	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
185	341.228225	10.129.102.30	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
186	341.228955	10.129.102.31	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
187	341.229442	10.129.102.0	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
188	341.230171	10.129.102.1	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
189	341.230657	10.129.102.2	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
190	341.231387	10.129.102.3	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
191	341.232116	10.129.102.4	10.129.211.13	ICMP	Destination unreachable (Port unreachable)
192	341.232603	10.129.102.5	10.129.211.13	ICMP	Destination unreachable (Port unreachable)

Table 4: Repeated ICMP Messages

At the beginning of the trace file, starting from packet 1 of the trace file, the IP that was identified at the compromised IP sent a DNS query to a domain name. This exchange of information between two parties is absolutely normal and does not indicate a problem. This step will be identified as step 1 in the attack process of our analysis. Even though the communication seems normal, a later analysis will show that the DNS server being queried is found amongst the DNS blacklist. The DNS query is shown as:

```
10.129.211.13 10.129.56.6 DNS Standard query A
bbjj.househot.com
```

In the following packets (packet No2 from our capture), the compromised IP gets a response back from the DNS query made earlier. This will be referred as step 2.

```
2 0.237997 10.129.56.6 10.129.211.13 DNS Standard query
response CNAME ypgw.wallloan.com A 216.234.235.165 A 151.198.6.55 A
216.234.247.191 A 68.112.229.228 A 61.189.243.240 A 218.12.94.58 A
61.145.119.63 A 202.98.223.87 A 218.249.83.118 A 68.186.110.158 A
221.208.154.214
```

Step 2 show some signs of unusual behaviour. A DNS response will generally have 5 IPs or less. In this case the answer came back with 11 hosts, 11 IPs.

Step 3, the compromised host try to establish connection with the first IP that appeared in the DNS query.

```
3 0.239858 10.129.211.13 216.234.235.165 TCP neod1 >
18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
```

Right after the attempt to establish connection to a host, an ICMP message was received indicating that the host is not live or not accepting connection on the port number that was used.

The compromised host did not have any success establishing connection with hosts (IP) from the first DNS query. From the trace, the author notices that the compromised host will start a second DNS query aiming at the canonical name (CNAME) that was in the DNS response on step 2. This will be referred as step 4.

```
9       337.528083  10.129.211.13       10.129.56.6  DNS  Standard query A
ypgw.wallloan.com
```

From the latest DNS query, stage 4, the DNS response will give another set of IPs.

```
10      337.757036  10.129.56.6  10.129.211.13       DNS  Standard     query
response A 61.189.243.240 A 61.145.119.63 A 151.198.6.55 A 202.98.223.87
A 218.249.83.118 A 68.186.110.158 A 68.112.229.228 A 218.12.94.58 A
216.234.235.165 A 216.234.247.191 A 221.208.154.214
```

Step 5 of the attack process:

Again, the DNS query has returned 11 IPs, which is also highly unusual. From the next few packets, one could notice that the compromised host will attempt another connection with the first host from the DNS response. On this occasion, the connection was successful. In Table 5 , packet 11 shows that the malicious IP 10.126.211.13 tries to establish the connection with other IPs. On packet 13, the three hand shake process is completed. From packet 14, the malicious IP start sending packets using the PUSH flag (this will be referred to as step 6).

The PUSH flag indicates that no delay should be observed, whether the receiving system is ready to accept the packet or not.

Packet No	Timestamp	Source IP	Destination IP	Protocol	Info
11	337.763493	10.129.211.13	61.189.243.240	TCP	neod2 > 18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
12	338.160099	61.189.243.240	10.129.211.13	TCP	18067 > neod2 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
13	338.160284	10.129.211.13	61.189.243.240	TCP	neod2 > 18067 [ACK] Seq=1 Ack=1 Win=64240 Len=0
14	338.160379	10.129.211.13	61.189.243.240	TCP	neod2 > 18067 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=13
15	338.719557	61.189.243.240	10.129.211.13	TCP	18067 > neod2 [ACK] Seq=1 Ack=14 Win=65522 Len=0
16	338.719607	10.129.211.13	61.189.243.240	TCP	neod2 > 18067 [PSH, ACK] Seq=14 Ack=1 Win=64240 Len=17
17	339.122268	61.189.243.240	10.129.211.13	TCP	18067 > neod2 [PSH, ACK] Seq=1 Ack=31 Win=65505 Len=23

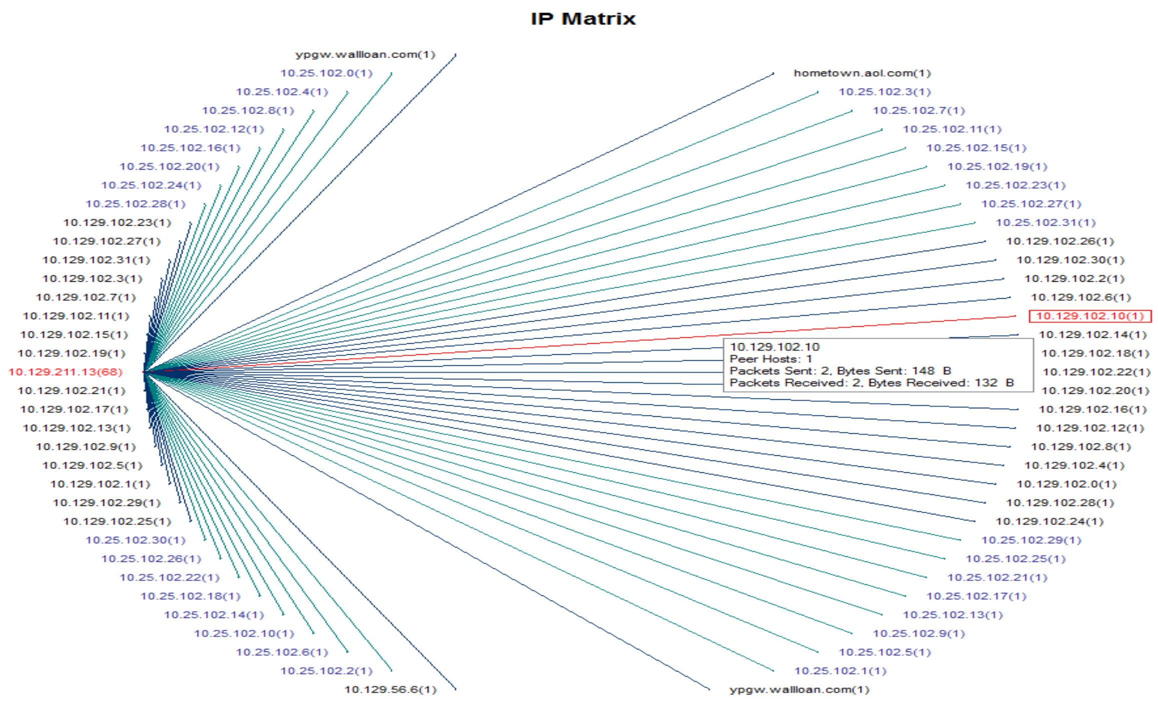
Table 5: malicious IP establishing connection

Tracking down the conversation between the infected host 10.129.211.13 and the target host 61.189.243.240 the following payload was recorded

```
USeR l l l l
NiCK p8-00196671
:a7 001 p8-00196671 :

USeRHOST p8-00196671
:a7 302 p8-00196671 :p8-00196671=+l@010.129.211.13

JOiN #p8 ihodc9hi
:a7 332 p8-00196671 #p8 :!Q
gfcagihehehadkcpcpgigpgngfhegphhgocogbgpgmcogdgpgncphihihigmgpgmhh
hegggjgigbhihihihicphdgpgdglhddjgbcogkhagh

:a7 333 p8-00196671 #p8 a 1134159047

:a7 366 p8-00196671 #p8 :
```

From the recorded payload, one could identify commands that botnet use. The author argues that at least three of the different steps described above could have triggered an alert indicating some sort irregularities. A further analysis on the DNS servers revealed commands that are commonly used by a the IRC-MocBot virus [1], which communicate through a port 18067 to a bot master and awaits command such as scan, DDOS or execute other malicious programs [2].

Going a few steps back into our analysis, the malicious user had tried to establish communication with all IPs that were under the CNAME of the DNS server. Figure 4-6 shows the matrix of communication between the malicious users and all the targeted computers. Having one IP communicating with multiple IPs is not a problem neither does it necessarily indicate something unusual. However, the nature of the communication between that one IP and all the other IPs will help us to understand and identify any sort of irregularities.



```
10.129.102.10
Peer Hosts: 1
Packets Sent: 2, Bytes Sent: 148  B
Packets Received: 2, Bytes Received: 132  B
```

Figure 4-6: Matrix of communication between attacker and victims PCs

From the Matrix, it is clear that all communications are centred on one IP. Again this is very unusual and should be flagged by any sensitive IDS. The case is made worse by the amount of data exchanged between the different IPs. Only one packet was exchanged in most cased between the attackers and the targeted IPs. Not being able to establish communication with all the hosts there were a good number of ICMP generated afterward. Again, a big number of ICMP messages from different IP belonging to the network should have been an indication that something was not right. Table 6 shows that there have been 59 ICMP Port Unreachable messages and from different, consecutive sources.

Name	Count
All Diagnosis Events	133
Transport Layer	7
TCP Invalid Checksum	5
TCP Retransmissions	2
Network Layer	126
ICMP Port Unreachable	59
IP Invalid Header Checksum	67

Table 6: strange behaviour IP

4.3.2 Snort analysis of the attack trace file

The trace file was passed into Snort for analysis and the following result was obtained.

```
Snort exiting
Run time for packet processing was 0.8000 seconds
===============================================================
===================
Snort processed 209 packets.
===============================================================
===================
Breakdown by protocol (includes rebuilt packets):
     ETH: 209       (100.000%)
  ETHdisc: 0         (0.000%)
    VLAN: 0          (0.000%)
    IPV6: 0          (0.000%)
  IP6 EXT: 0         (0.000%)
  IP6opts: 0         (0.000%)
  IP6disc: 0          (0.000%)
      IP4: 209        (100.000%)
  IP4disc: 0          (0.000%)
    TCP 6: 0          (0.000%)
    UDP 6: 0          (0.000%)
    ICMP6: 0          (0.000%)
  ICMP-IP: 0          (0.000%)
      TCP: 144        (68.900%)
      UDP: 6          (2.871%)
     ICMP: 59         (28.230%)
  TCPdisc: 0          (0.000%)
  UDPdisc: 0          (0.000%)
  ICMPdis: 0          (0.000%)
     FRAG: 0          (0.000%)
   FRAG 6: 0          (0.000%)
      ARP: 0          (0.000%)
    EAPOL: 0          (0.000%)
  ETHLOOP: 0          (0.000%)
      IPX: 0          (0.000%)
IPv4/IPv4: 0          (0.000%)
IPv4/IPv6: 0          (0.000%)
IPv6/IPv4: 0          (0.000%)
IPv6/IPv6: 0          (0.000%)
      GRE: 0          (0.000%)
  GRE ETH: 0          (0.000%)
 GRE VLAN: 0          (0.000%)
 GRE IPv4: 0          (0.000%)
 GRE IPv6: 0          (0.000%)
GRE IP6 E: 0          (0.000%)
 GRE PPTP: 0          (0.000%)
  GRE ARP: 0          (0.000%)
  GRE IPX: 0          (0.000%)
 GRE LOOP: 0          (0.000%)
     MPLS: 0          (0.000%)
    OTHER: 0          (0.000%)
  DISCARD: 0          (0.000%)
InvChkSum: 209        (100.000%)
   S5 G 1: 0          (0.000%)
   S5 G 2: 0          (0.000%)
    Total: 209
==============================================================
===================
Action Stats:
ALERTS: 0
LOGGED: 0
PASSED: 0
```

As shown by the result above, Snort did not detect any of the different attack steps what were identified as part of this research. Referring to the different attack steps, Snort does not provide:

- A way to detect known bad DNS server
- A way to detect irregularities within DNS response
- A way to detect botnet communications
- Snort does not track connections
- Snort does not correlate different alerts to have a wider view of the attack that is taking place.

A modern Intrusion Detection System should be able to cover the points mentioned above. Hence the need to design a new approach of tackling the latest attacks.

Scenario interpretation

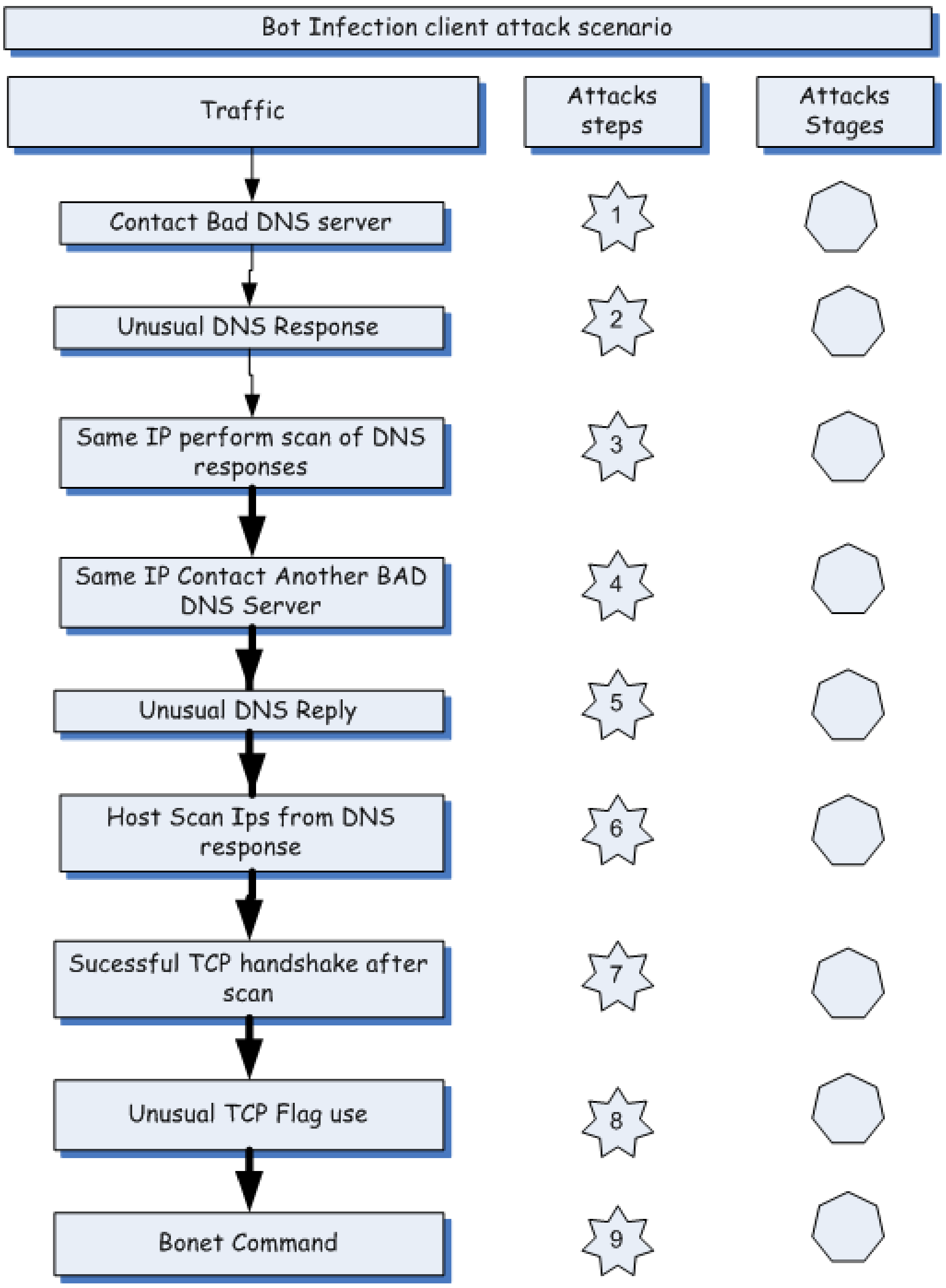

Bot Infection client attack scenario
Traffic
Attacks steps
Attacks Stages
Contact Bad DNS server
Unusual DNS Response
Same IP perform scan of DNS responses
Same IP Contact Another BAD DNS Server
Unusual DNS Reply
Host Scan Ips from DNS response
Sucessful TCP handshake after scan
Unusual TCP Flag use
Bonet Command
1
2
3
4
5
6
7
8
9

Figure 4-7: Attacks stages: bot infected computer

Scenario interpretation

In the light of events that took place in this scenario, it is difficult to identify each of the steps as a successful attack if considered separately. In step1, the attacker contacted a DNS server which is completely legal and does not violate any law. However, there was an indication that the intention behind this activity was not good as the DNS server contacted is known as a bad DNS server [113]. Step2 in this scenario is a normal DNS response. Yes, the responses contained more entry that usual, but the response was quite legitimate. The activity of the attacker could have been stopped when the scans were performed. However, unless the scans are of a type that will create a DDOS attack, most systems would consider them as noise. The only step that could have been flagged as a medium step is the last step. Again, this very step is a normal activity of IRC chat servers. Here are nine steps that could be interpreted as very legal when taken individually but yet, put together, they form a very powerful attack. From this scenario, many attack trees can be deducted.

As shown in Figure 4-8, the system would identify a successful attack if the malicious user starts by scanning one of many computers, with some possible failure in the scans, then move in to sending botnet commands or IRC commands.

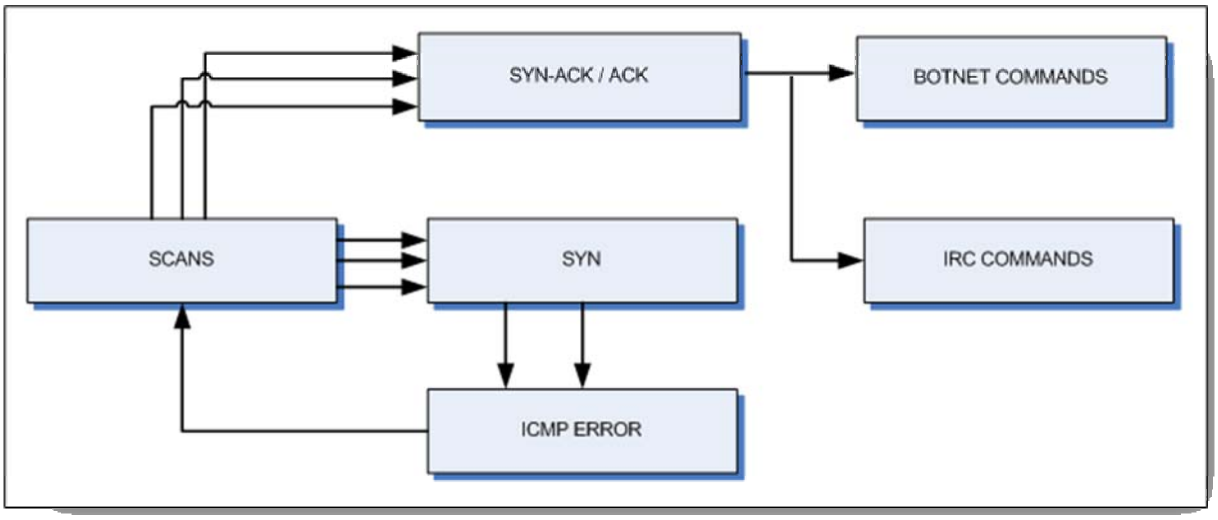


Figure 4-8: Attack Tree 1 - bot infected

The attack tree in Figure 4-8 could be altered by using proxy server. A malicious user will proxy server to perform SCANs. Also, proxy can be used to push data to victim computer. The new attack tree would be as shown in Figure 4-9

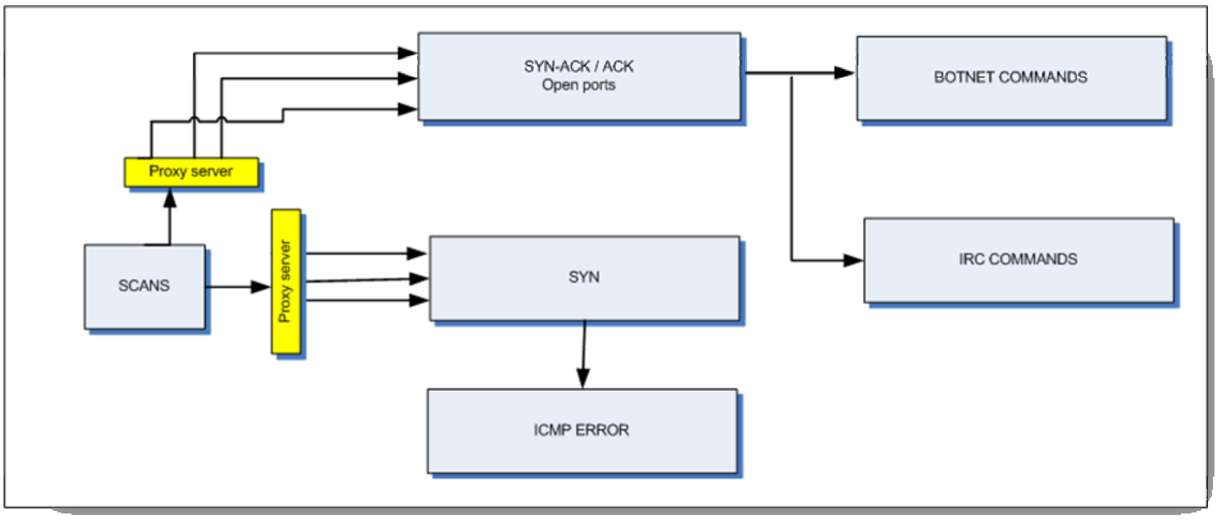


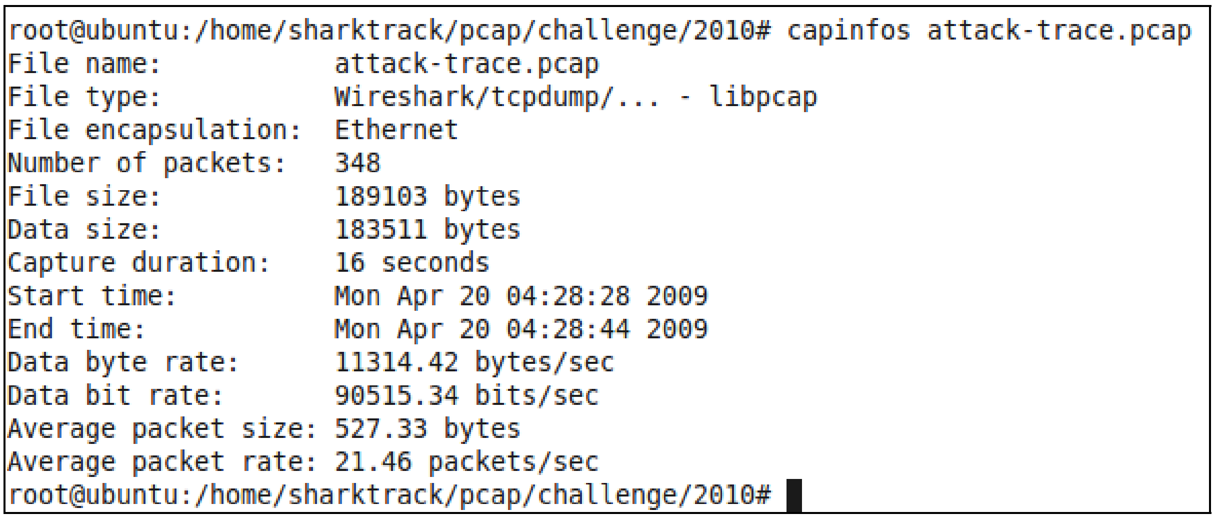
Figure 4-9: Attack tree bot infected with proxy

4.4 Scenario Beta

In this scenario, the author used a trace file provided by the Honeynet project, a live capture as an attacker takes advantages of Windows XP SP1 vulnerability. Using an automated malware, the attacker takes advantage of one of the vulnerabilities disclosed in the Microsoft Security Bulletin MS09-059 [114] i.e. vulnerability in the Local Security Authority that could lead to a DDOS attack.

The objective of this analysis is to show that the attackers could have been identified if attacks indicators were set correctly. At the end of this analysis, the research will suggest some attacks indicators and the diagram representing the different stages of the attack will be drawn.

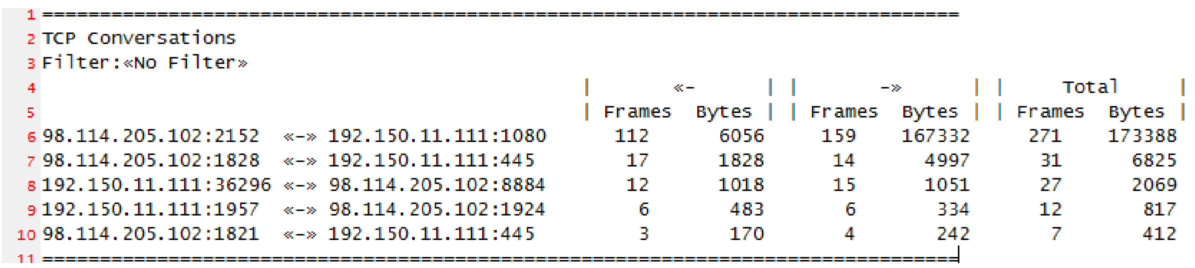
Trace file statistics summary



```
root@ubuntu:/home/sharktrack/pcap/challenge/2010# capinfos attack-trace.pcap
File name:           attack-trace.pcap
File type:           Wireshark/tcpdump/... - libpcap
File encapsulation:  Ethernet
Number of packets:   348
File size:           189103 bytes
Data size:           183511 bytes
Capture duration:    16 seconds
Start time:          Mon Apr 20 04:28:28 2009
End time:            Mon Apr 20 04:28:44 2009
Data byte rate:      11314.42 bytes/sec
Data bit rate:       90515.34 bits/sec
Average packet size: 527.33 bytes
Average packet rate: 21.46 packets/sec
root@ubuntu:/home/sharktrack/pcap/challenge/2010#
```

Figure 4-10: Attack-trace.pcap file summary

In order to discover the actions of the attacker, the author ran a command that gave a summary of all the conversations between the attacker and the victim PC. As shown in **Figure 4-11** , they had been five conversations.



```
================================================================================
TCP Conversations
Filter:«No Filter»
                                               |       «-      | |       -»      | |     Total     |
                                               | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |
98.114.205.102:2152  «-» 192.150.11.111:1080      112     6056     159   167332     271   173388
98.114.205.102:1828  «-» 192.150.11.111:445        17     1828      14     4997      31     6825
192.150.11.111:36296 «-» 98.114.205.102:8884       12     1018      15     1051      27     2069
192.150.11.111:1957  «-» 98.114.205.102:1924        6      483       6      334      12      817
98.114.205.102:1821  «-» 192.150.11.111:445         3      170       4      242       7      412
================================================================================
```

Figure 4-11: conversation between the attacker and the victim PC

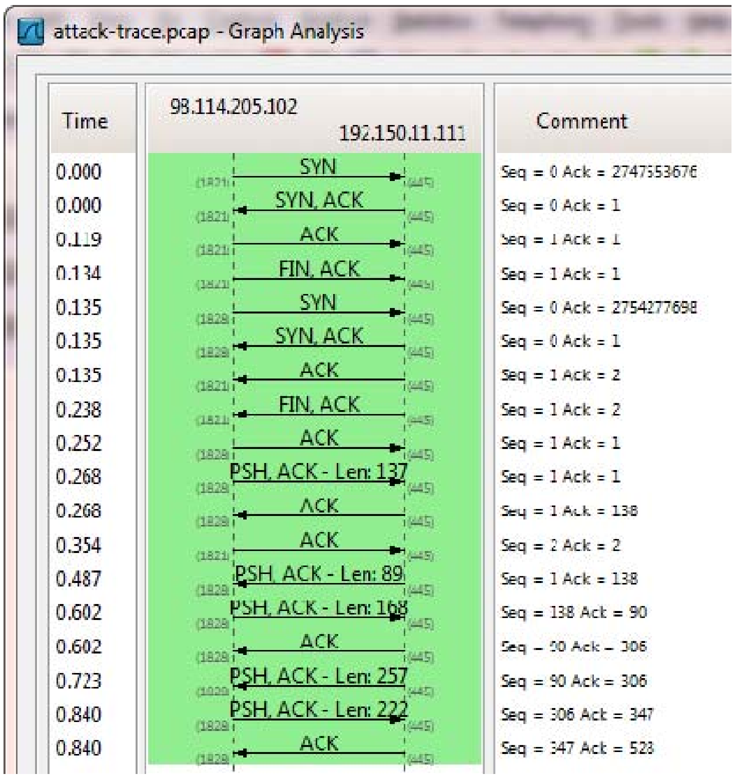
Taking a closer look at conversation 1, the attacker was trying to establish whether the targeted PC was live: the reconnaissance phase. On this occasion, as shown in packet 7 & 8, the [FIN, ACK] & [ACK] were received. The first conversation could be considered as the first step of the attack.

No.	Time	Source	Destination	Protocol	Info
1	0	98.114.20	192.150.11.11	TCP	donnyworld > microsoft-ds [SYN] Seq=0 Win=64240 Len=0

		5.102	1		MSS=1460 SACK_PERM=1
2	0.000464	192.150.11.111	98.114.205.102	TCP	microsoft-ds > donnyworld [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
3	0.119058	98.114.205.102	192.150.11.111	TCP	donnyworld > microsoft-ds [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.134175	98.114.205.102	192.150.11.111	TCP	donnyworld > microsoft-ds [FIN, ACK] Seq=1 Ack=1 Win=64240 Len=0
7	0.135193	192.150.11.111	98.114.205.102	TCP	microsoft-ds > donnyworld [ACK] Seq=1 Ack=2 Win=5840 Len=0
8	0.238169	192.150.11.111	98.114.205.102	TCP	microsoft-ds > donnyworld [FIN, ACK] Seq=1 Ack=2 Win=5840 Len=0
12	0.354302	98.114.205.102	192.150.11.111	TCP	donnyworld > microsoft-ds [ACK] Seq=2 Ack=2 Win=64240 Len=0

Table 7: reconnaissance phase

From a closer look at the reconnaissance, the attacker contacted the victim PC via port 445 as shown in **Figure 4-12**. Port 445 was used for file sharing service [115] and allowed both inbound and outbound traffic. Most security settings would recommend blocking that port number [116]. In the instance of having port 445 open, the remote system accessing the local resource should be known in advanced and a list should be built to keep out any other intruders. This conversation was not flagged in Snort as potentially dangerous, yet most systems fail to protect this port [117]. Even though there was nothing technically illegal, Snort should have set a flag for port 445. Also Snort could have set a variable for systems allowed to access the local shared resource externally – like **$EXTERNAL_SHARE**. One of the problems is that Snort

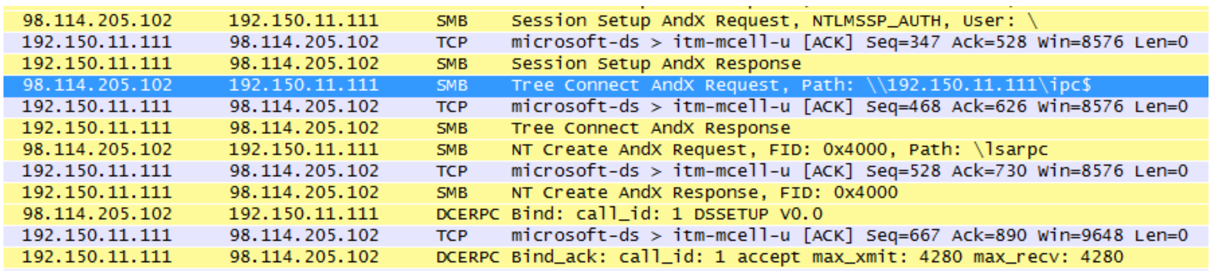


attack-trace.pcap - Graph Analysis

Time	98.114.205.102 → 192.150.11.111	Comment
0.000	(1821) SYN → (445)	Seq = 0 Ack = 2747353676
0.000	(1821) ← SYN, ACK (445)	Seq = 0 Ack = 1
0.119	(1821) ACK → (445)	Seq = 1 Ack = 1
0.134	(1821) FIN, ACK → (445)	Seq = 1 Ack = 1
0.135	(1828) SYN → (445)	Seq = 0 Ack = 2754277698
0.135	(1828) ← SYN, ACK (445)	Seq = 0 Ack = 1
0.135	(1821) ← ACK (445)	Seq = 1 Ack = 2
0.238	(1821) ← FIN, ACK (445)	Seq = 1 Ack = 2
0.252	(1828) ACK → (445)	Seq = 1 Ack = 1
0.268	(1828) PSH, ACK - Len: 137 → (445)	Seq = 1 Ack = 1
0.268	(1828) ← ACK (445)	Seq = 1 Ack = 138
0.354	(1821) ACK → (445)	Seq = 2 Ack = 2
0.487	(1828) ← PSH, ACK - Len: 89 (445)	Seq = 1 Ack = 138
0.602	(1828) PSH, ACK - Len: 168 → (445)	Seq = 138 Ack = 90
0.602	(1828) ← ACK (445)	Seq = 90 Ack = 306
0.723	(1828) ← PSH, ACK - Len: 257 (445)	Seq = 90 Ack = 306
0.840	(1828) PSH, ACK - Len: 222 → (445)	Seq = 306 Ack = 347
0.840	(1828) ← ACK (445)	Seq = 347 Ack = 523

Figure 4-12: Conversation 1 Graph Analysis

In the second conversation, the attacker took advantage of the buffer overflow vulnerability and then compromised the shared folder ipc$ and invoke \LSARPC. LSARPC is generally used to gain system information in the intension to launch an attack [118].

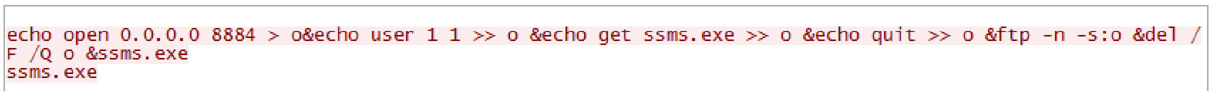


```
98.114.205.102    192.150.11.111    SMB     Session Setup AndX Request, NTLMSSP_AUTH, User: \
192.150.11.111    98.114.205.102    TCP     microsoft-ds > itm-mcell-u [ACK] Seq=347 Ack=528 Win=8576 Len=0
192.150.11.111    98.114.205.102    SMB     Session Setup AndX Response
98.114.205.102    192.150.11.111    SMB     Tree Connect AndX Request, Path: \\192.150.11.111\ipc$
192.150.11.111    98.114.205.102    TCP     microsoft-ds > itm-mcell-u [ACK] Seq=468 Ack=626 Win=8576 Len=0
192.150.11.111    98.114.205.102    SMB     Tree Connect AndX Response
98.114.205.102    192.150.11.111    SMB     NT Create AndX Request, FID: 0x4000, Path: \lsarpc
192.150.11.111    98.114.205.102    TCP     microsoft-ds > itm-mcell-u [ACK] Seq=528 Ack=730 Win=8576 Len=0
192.150.11.111    98.114.205.102    SMB     NT Create AndX Response, FID: 0x4000
98.114.205.102    192.150.11.111    DCERPC  Bind: call_id: 1 DSSETUP V0.0
192.150.11.111    98.114.205.102    TCP     microsoft-ds > itm-mcell-u [ACK] Seq=667 Ack=890 Win=9648 Len=0
192.150.11.111    98.114.205.102    DCERPC  Bind_ack: call_id: 1 accept max_xmit: 4280 max_recv: 4280
```

Figure 4-13: buffer overflow and service binding

After successfully compromising the IPC$ share, the attacker set an FTP server in the third conversation using the command. He then called

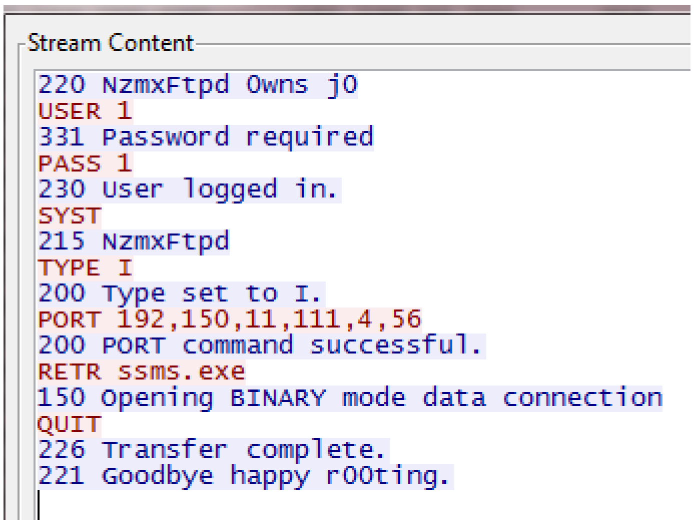
DsRoleUpgradeDownlevelServer() which was used to overflow the stack. The shell code is then executed through port 1957. After connecting to the victim's system on port 1957, the attacker then gained access to the command line, cmd.exe.



```
echo open 0.0.0.0 8884 > o&echo user 1 1 >> o &echo get ssms.exe >> o &echo quit >> o &ftp -n -s:o &del /
F /Q o &ssms.exe
ssms.exe
```

Figure 4-14: Command exploits (FTP)

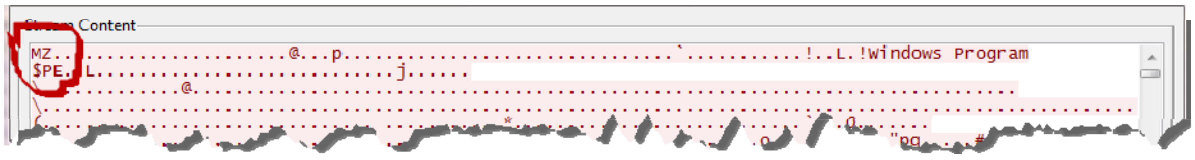
In the fourth conversation, the attacker transferred the files to the victim system using FTP



```
220 NzmxFtpd Owns j0
USER 1
331 Password required
PASS 1
230 User logged in.
SYST
215 NzmxFtpd
TYPE I
200 Type set to I.
PORT 192,150,11,111,4,56
200 PORT command successful.
RETR ssms.exe
150 Opening BINARY mode data connection
QUIT
226 Transfer complete.
221 Goodbye happy r00ting.
```

Figure 4-15: File transfer to victim system

In the last conversation, the malware is then executed as shown by the key signature of .exe files in **Figure 4-16** MZ and PE [119] [120] [121].



```
MZ.......................@...p...............................`...........!..L.!Windows Program
$PE..L...............................j......
\..........@.............................................................................
\.............................................................................................
```

Figure 4-16: Windows executable file in traffic

In summary the sequence of attack is presented in [Table 8: sequence of attack]

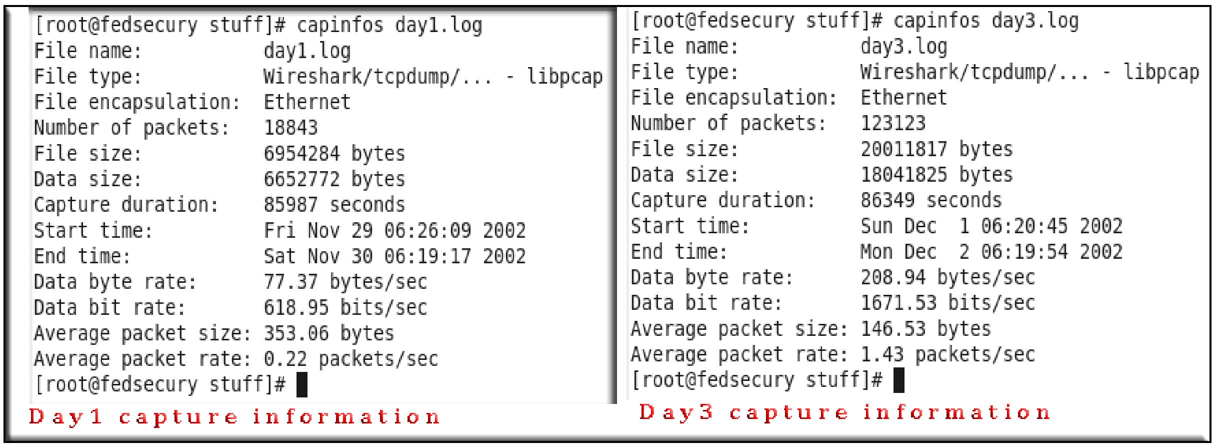
	Attack steps	*Possible security*
1	*Connection to port 445*	• *Nothing illegal in the connection* • *Flag on port 445* • *Predefined list of remote system allowed to access local shared resources*
2	• *SMS session as NULL user over port 445* • *Connection to \\192.150.11.111\ipc$*	• *Flag on NULL user* • *Flag on access local shared resources*
3	*Connection to LSARPC over SMB*	
4	*Calls DsRoleUpgradeDownlevelServer() with a long szDomainName parameter containing a shellcode of type "bind shell", which will overflow the stack (again, through the same port, 445).*	*Signature to detect buffer overflow*
5	*Execution of the shellcode* *Binds port 1957 and waits for connection*	
6	*Connection to port 1957* *Get access to shell command (cmd.exe)*	
7	*FTP session initialisation*	
8	*Sending executable to the victim system*	
9	*Malware code execution*	

Table 8: sequence of attack

4.5 Scenario Charlie

The two traces files used in this scenario are the results of the scan of the month 28 [122]. In the files provided by The Honeynet Project, the attacker use IPv6 tunnelling to realise the attack. Based on the analysis strategy defined earlier in this chapter, different statistics are retrieved from the trace file in order to have an idea of what sort of activity could be going on. This analysis is a typical example of how clever the attacks are becoming.

Trace file statistics summary



```
[root@fedsecury stuff]# capinfos day1.log
File name:              day1.log
File type:              Wireshark/tcpdump/... - libpcap
File encapsulation:     Ethernet
Number of packets:      18843
File size:              6954284 bytes
Data size:              6652772 bytes
Capture duration:       85987 seconds
Start time:             Fri Nov 29 06:26:09 2002
End time:               Sat Nov 30 06:19:17 2002
Data byte rate:         77.37 bytes/sec
Data bit rate:          618.95 bits/sec
Average packet size:    353.06 bytes
Average packet rate:    0.22 packets/sec
[root@fedsecury stuff]#
```

```
[root@fedsecury stuff]# capinfos day3.log
File name:              day3.log
File type:              Wireshark/tcpdump/... - libpcap
File encapsulation:     Ethernet
Number of packets:      123123
File size:              20011817 bytes
Data size:              18041825 bytes
Capture duration:       86349 seconds
Start time:             Sun Dec  1 06:20:45 2002
End time:               Mon Dec  2 06:19:54 2002
Data byte rate:         208.94 bytes/sec
Data bit rate:          1671.53 bits/sec
Average packet size:    146.53 bytes
Average packet rate:    1.43 packets/sec
[root@fedsecury stuff]#
```

Figure 4-17: File information Scenario Charlie

4.5.1 Trace files analysis

Tools

In order to achieve the result presented later in this section, a selection of opensource tools were considered.

List of IPs involved

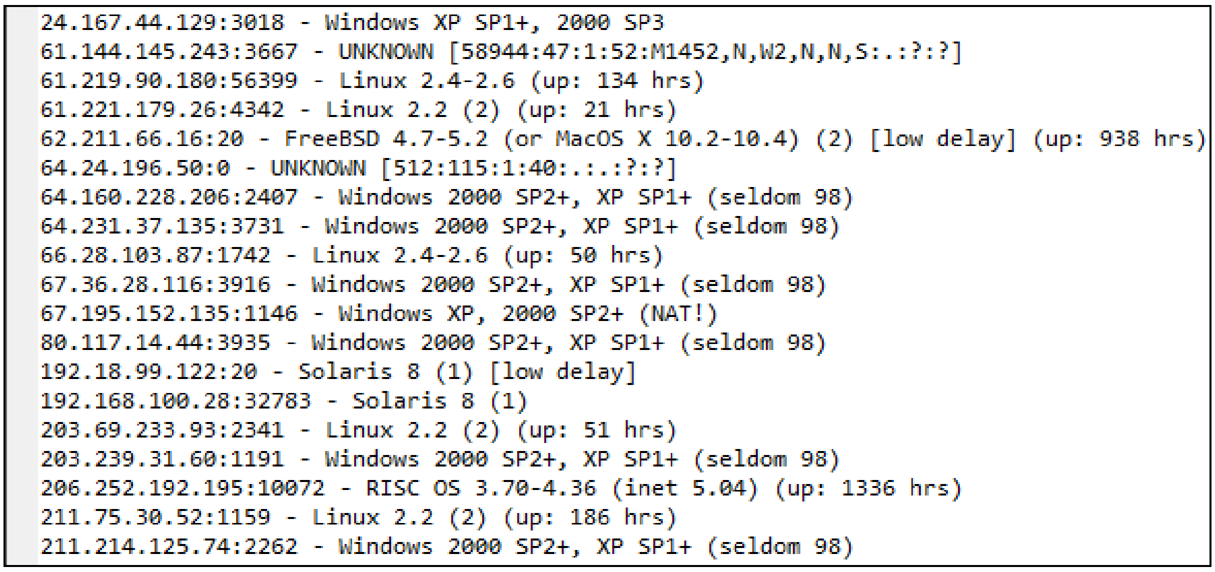
In the first trace file, 453 IPs were retrieved. Looking at Table 9 IP.Address = 192.168.100.28 appear to be at the centre of all conversations and communications. This could be an indication that it is the attacker of the target system. Further studies will reveal that that IPs was actually the IP from the Honeypot. Based on the table which is an extract of the summary of IP addresses and their activity, it appears that less than 20 IPs are at the centre of the activities recorded. However, there are many more IPs addresses that have been involved but at a low level. This technique is a typical demonstration that attacks are decentralised in order to make the detection difficult and, render scoring algorithms useless

IP	value	rate	percent
192.168.100.28	18853	0.000219	100.00%
206.252.192.195	4109	0.000048	21.79%
61.219.90.180	3732	0.000043	19.80%
62.211.66.53	2115	0.000025	11.22%
192.18.99.122	1543	0.000018	8.18%
148.244.153.91	859	0.00001	4.56%
217.116.38.10	846	0.00001	4.49%
61.134.3.11	846	0.00001	4.49%
80.117.14.44	821	0.00001	4.35%
62.211.66.16	377	0.000004	2.00%
200.33.146.213	105	0.000001	0.56%
192.12.94.30	104	0.000001	0.55%
192.31.80.30	102	0.000001	0.54%
140.135.18.25	78	0.000001	0.41%
200.33.146.217	75	0.000001	0.40%
192.5.6.30	72	0.000001	0.38%
200.33.213.66	64	0.000001	0.34%
192.35.51.30	58	0.000001	0.31%
63.250.206.138	52	0.000001	0.28%
192.168.100.196	50	0.000001	0.27%

Table 9: List of IPs day1

List of Operating System

Various OS have been detected (Figure 4-18)



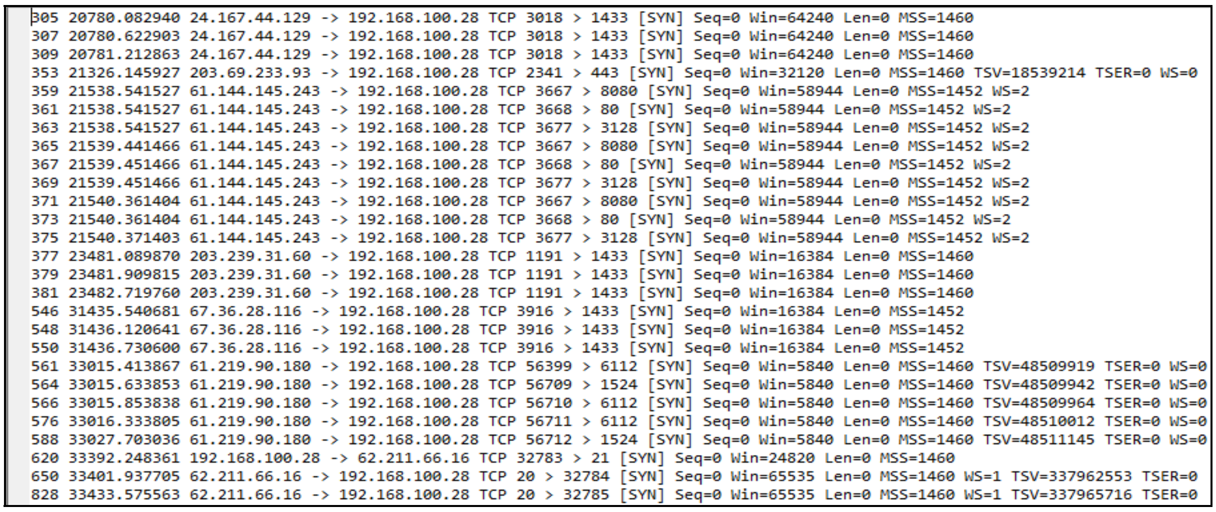
```
24.167.44.129:3018 - Windows XP SP1+, 2000 SP3
61.144.145.243:3667 - UNKNOWN [58944:47:1:52:M1452,N,W2,N,N,S:.:?:?]
61.219.90.180:56399 - Linux 2.4-2.6 (up: 134 hrs)
61.221.179.26:4342 - Linux 2.2 (2) (up: 21 hrs)
62.211.66.16:20 - FreeBSD 4.7-5.2 (or MacOS X 10.2-10.4) (2) [low delay] (up: 938 hrs)
64.24.196.50:0 - UNKNOWN [512:115:1:40:.:.:?:?]
64.160.228.206:2407 - Windows 2000 SP2+, XP SP1+ (seldom 98)
64.231.37.135:3731 - Windows 2000 SP2+, XP SP1+ (seldom 98)
66.28.103.87:1742 - Linux 2.4-2.6 (up: 50 hrs)
67.36.28.116:3916 - Windows 2000 SP2+, XP SP1+ (seldom 98)
67.195.152.135:1146 - Windows XP, 2000 SP2+ (NAT!)
80.117.14.44:3935 - Windows 2000 SP2+, XP SP1+ (seldom 98)
192.18.99.122:20 - Solaris 8 (1) [low delay]
192.168.100.28:32783 - Solaris 8 (1)
203.69.233.93:2341 - Linux 2.2 (2) (up: 51 hrs)
203.239.31.60:1191 - Windows 2000 SP2+, XP SP1+ (seldom 98)
206.252.192.195:10072 - RISC OS 3.70-4.36 (inet 5.04) (up: 1336 hrs)
211.75.30.52:1159 - Linux 2.2 (2) (up: 186 hrs)
211.214.125.74:2262 - Windows 2000 SP2+, XP SP1+ (seldom 98)
```

Figure 4-18: Operating System List - Day1

At this stage, there is no quick indication as to what could be taking place. However, one could guess that an attacker is controlling various systems or using various systems to attack the targeted system.

TCP Transactions

Looking at the TCP transactions, it appears that the many attempts were made to connect to the system. Also, there is a possibility that there had been a lot of data exchanged between the honeypot and the other participant systems (Figure 4-19).



```
305 20780.082940 24.167.44.129 -> 192.168.100.28 TCP 3018 > 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
307 20780.622903 24.167.44.129 -> 192.168.100.28 TCP 3018 > 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
309 20781.212863 24.167.44.129 -> 192.168.100.28 TCP 3018 > 1433 [SYN] Seq=0 Win=64240 Len=0 MSS=1460
353 21326.145927 203.69.233.93 -> 192.168.100.28 TCP 2341 > 443 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 TSV=18539214 TSER=0 WS=0
359 21538.541527 61.144.145.243 -> 192.168.100.28 TCP 3667 > 8080 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
361 21538.541527 61.144.145.243 -> 192.168.100.28 TCP 3668 > 80 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
363 21538.541527 61.144.145.243 -> 192.168.100.28 TCP 3677 > 3128 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
365 21539.441466 61.144.145.243 -> 192.168.100.28 TCP 3667 > 8080 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
367 21539.451466 61.144.145.243 -> 192.168.100.28 TCP 3668 > 80 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
369 21539.451466 61.144.145.243 -> 192.168.100.28 TCP 3677 > 3128 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
371 21540.361404 61.144.145.243 -> 192.168.100.28 TCP 3667 > 8080 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
373 21540.361404 61.144.145.243 -> 192.168.100.28 TCP 3668 > 80 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
375 21540.371403 61.144.145.243 -> 192.168.100.28 TCP 3677 > 3128 [SYN] Seq=0 Win=58944 Len=0 MSS=1452 WS=2
377 23481.089870 203.239.31.60 -> 192.168.100.28 TCP 1191 > 1433 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
379 23481.909815 203.239.31.60 -> 192.168.100.28 TCP 1191 > 1433 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
381 23482.719760 203.239.31.60 -> 192.168.100.28 TCP 1191 > 1433 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
546 31435.540681 67.36.28.116 -> 192.168.100.28 TCP 3916 > 1433 [SYN] Seq=0 Win=16384 Len=0 MSS=1452
548 31436.120641 67.36.28.116 -> 192.168.100.28 TCP 3916 > 1433 [SYN] Seq=0 Win=16384 Len=0 MSS=1452
550 31436.730600 67.36.28.116 -> 192.168.100.28 TCP 3916 > 1433 [SYN] Seq=0 Win=16384 Len=0 MSS=1452
561 33015.413867 61.219.90.180 -> 192.168.100.28 TCP 56399 > 6112 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=48509919 TSER=0 WS=0
564 33015.633853 61.219.90.180 -> 192.168.100.28 TCP 56709 > 1524 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=48509942 TSER=0 WS=0
566 33015.853838 61.219.90.180 -> 192.168.100.28 TCP 56710 > 6112 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=48509964 TSER=0 WS=0
576 33016.333805 61.219.90.180 -> 192.168.100.28 TCP 56711 > 6112 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=48510012 TSER=0 WS=0
588 33027.703036 61.219.90.180 -> 192.168.100.28 TCP 56712 > 1524 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=48511145 TSER=0 WS=0
620 33392.248361 192.168.100.28 -> 62.211.66.16 TCP 32783 > 21 [SYN] Seq=0 Win=24820 Len=0 MSS=1460
650 33401.937705 62.211.66.16 -> 192.168.100.28 TCP 20 > 32784 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1 TSV=337962553 TSER=0
828 33433.575563 62.211.66.16 -> 192.168.100.28 TCP 20 > 32785 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1 TSV=337965716 TSER=0
```

Figure 4-19: TCP Transactions summary

TCP Conversations

On the trace file provided for the first day, 57 TCP conversations, as well as 394 UDP and 452 IP conversations were identified

56 conversations were observed amongst which the following ports number have been recorded: 21, 80, 1524, 5555, 6112, 6667, 7000, 32784, 32785, 32786, 32788, 32792, and 32794.

It appears that FTP and HTTP traffic were recorded. The presence of FTP could indicate that the attacker had successfully connected to another system and then uploaded files.

Port Number	Description
21	File transfer protocol – Normal operation
80	Web browsing and related activities (e.g. file transfer)
1524	well-known port for Trojan activity [123][124]
5555	A well-known malware ServeMe uses this port for communication [125]
6667	Well-known port for IRC communication [126]
7000	Well-known port used by “malware Exploit” translation [125]
32784	Sometimes used as RPC in Solaris Boxes [127]
32785	Sometimes used as RPC in Solaris Boxes [127]
32786	Sometimes used as RPC in Solaris Boxes [127]
32788	Sometimes used as RPC in Solaris Boxes [127]
32792	Found in DNS poisoning [128][128][128][126][125][125][125][125][124][123][122][121][120][119][118][117][116][115][114][113][112][111][110][109][108][107][107][107][107][107][107][107][107][107], generally opened on Solaris port as listeners [129]
32794	No Particular activity found on this port

Table 10: compromised port numbers

There is a good indication of malicious activity based on the port numbers that have been used during the different conversations. This will be confirmed when looking at the details of packets. Given the large number of conversations, it was not appropriate to look into each conversation. However, based on the port numbers that are present in the different conversation, a number of intelligent filters will be applied to identify any possible malicious activity.

Scenario interpretation

They are twelve major steps that the attackers took to perform all his tasks. Each major step can be elaborated into many smaller steps. More interestingly, the attacker did not use the same IP to perform the different attacks. The IPs that were used to perform the attacks are located in different countries. Applying a multi-stage detection technique to an IP that is not tracked would not be of much help. Rather, it is important to understand the nature of the different steps (attacks) and collate them for a bigger picture to actually see what was going on.

From this scenario, it is important to learn that attacks, when viewed separately, could generate alerts that will not mean much. Few of these steps, when taken individually do not actually violate any protocol definition that will cause any firewall or IDS. For instance, in step 2, the attackers download files using FTP which is absolutely normal. Step 7 could also be interpreted as normal because performing a remote control does not technically hold a protocol violation or abuse. The same analogy will apply to step 11.

After setting up the IPv6 tunnelling in day one, the attacker came back on another day to configure and use the IPv6 tunnelling where many files there send to and from the victims systems.

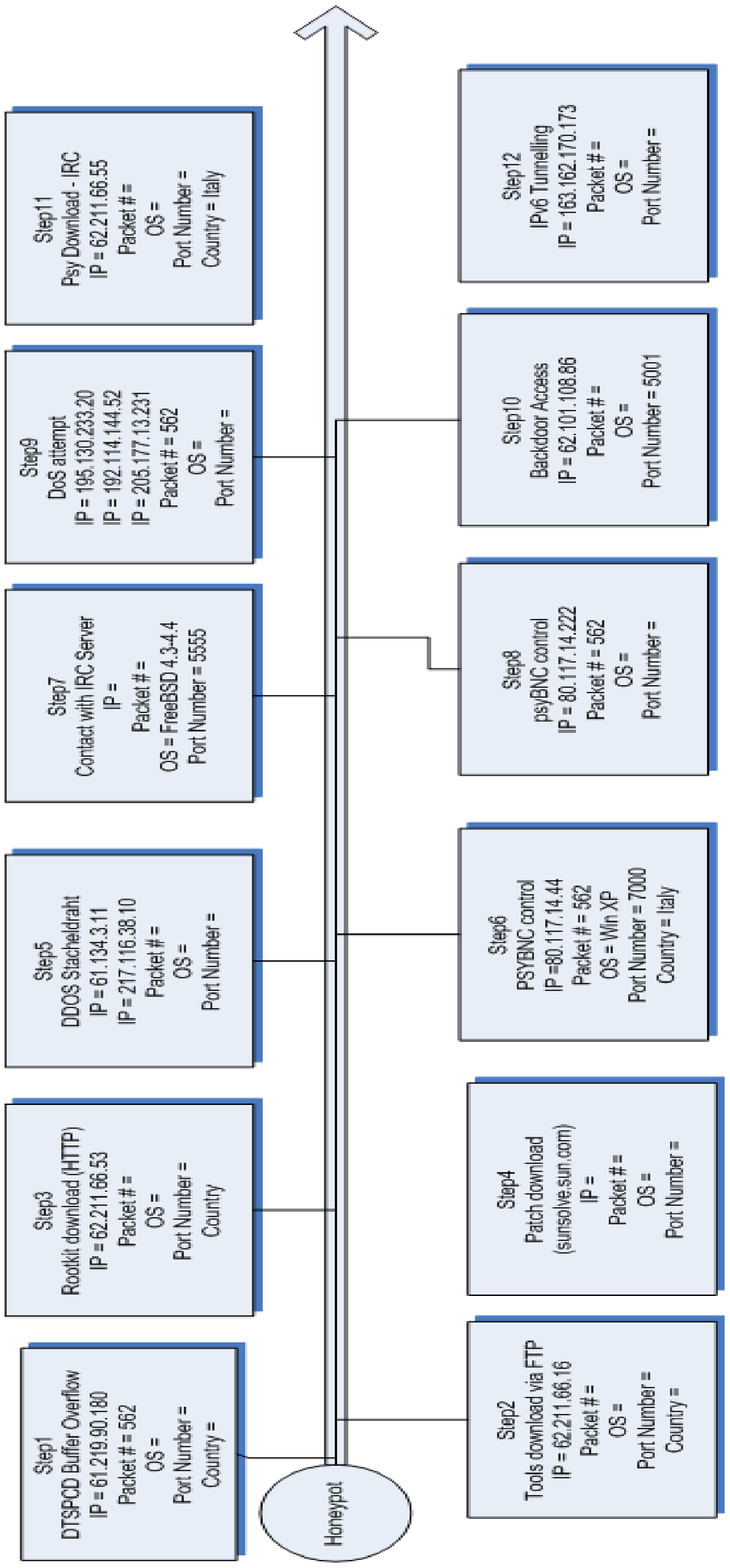

Honeypot
Step1 DTSPCD Buffer Overflow IP = 61.219.90.180 Packet # = 562 OS = Port Number = Country =
Step2 Tools download via FTP IP = 62.211.66.16 Packet # = OS = Port Number = Country =
Step3 Rootkit download (HTTP) IP = 62.211.66.53 Packet # = OS = Port Number = Country
Step4 Patch download (sunsolve.sun.com) IP = Packet # = OS = Port Number =
Step5 DDOS Stacheldraht IP = 61.134.3.11 IP = 217.116.38.10 Packet # = OS = Port Number =
Step6 PSYBNC control IP =80.117.14.44 Packet # = 562 OS = Win XP Port Number = 7000 Country = Italy
Step7 Contact with IRC Server IP = Packet # = OS = FreeBSD 4.3-4.4 Port Number = 5555
Step8 psyBNC control IP = 80.117.14.222 Packet # = 562 OS = Port Number =
Step9 DoS attempt IP = 195.130.233.20 IP = 192.114.144.52 IP = 205.177.13.231 Packet # = 562 OS = Port Number =
Step10 Backdoor Access IP = 62.101.108.86 Packet # = OS = Port Number = 5001
Step11 Psy Download - IRC IP = 62.211.66.55 Packet # = OS = Port Number = Country = Italy
Step12 IPv6 Tunnelling IP = 163.162.170.173 Packet # = OS = Port Number =

Figure 4-20: Sequence of attack scan28

Attack trees

Based on this scenario, multiple possible attacks trees can be defined.

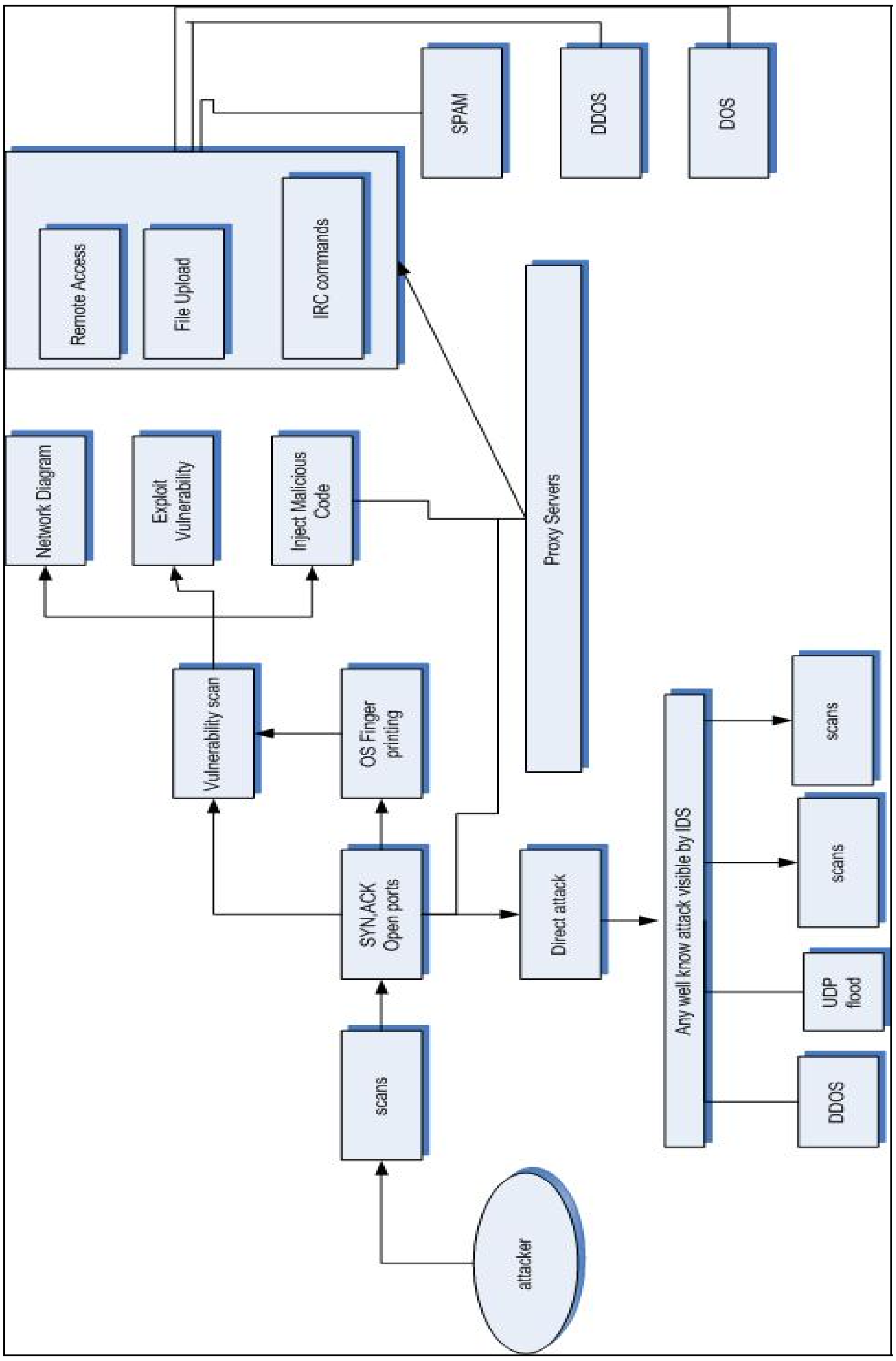


Figure 4-21: attack branches

4.6 Modelling multistage attacks

Modelling attacks can be a rather complex when working in an environment where there is no specific format or pattern used by the attackers. The level of sophistication of attacks has raised considerably and, it is becoming more difficult to distinguish normal behaviour from attack behaviour. In some cases, only the intention behind the actions performed make the difference between the legitimate user and the malicious one. The difficulty resides in the uniqueness of almost every single attack. In scenario 1, the attacker took advantage of the weakness of DNS protocol to get information about “bad DNS” and their associate IPs. After a scan, the attacker identified live hosts which were made part of a botnet. In the second scenario, the attacker took advantage of a vulnerability found in Windows XP; he went on executing a buffer overflow that allowed him to remotely take control of the victim system. In the third scenario, the attacker used either many proxies server or various compromised hosts to scan for vulnerabilities, exploit the vulnerabilities, remotely control the system, create an IPv6 tunnel over IPv4 to copy file and execute the program. At the time of the attacks, hardly any system had a good knowledge of IPv6. Not only was IPv6 not properly identified, but hardly any system was able to decode it. As a direct consequence, any attack performed using IPv6 would be successful. Even though many systems are now capable of IPv6 decoding, using IPv6 tunnelling over IPv4 remains a security challenge [130] [131] [132].

In the light of the scenarios that were used to understand multistage attacks, the model built in this chapter makes some assertions:

- very little differs from legitimate traffic to illegal traffic as shown in Scenario Alpha
- Legitimate but not innocent steps are taken in favour of the attacks. These steps are detectable by current IDS as being a problem which is in fact right.
- tracking even legitimate steps are important but will be costly [resources]
- predefined actions will be defined
- known attacks patterns are predefined into an attack tree
- An administrator should have a knowledge of the system being protected to build attack trees
- an attack tree should be built
    - For Windows based systems, all Microsoft bulletins should be transformed into attack tree enabling a multi-stage detection technique
- the attack tree should be updated regularly
- Interaction with internal event: this will be done by installing an IDS agent on local system so that they can report events (events that are generally sent to SYSLOG)

To successfully detect attacks, the framework proposed in this chapter will consider the activities as performed by the attackers and the activities as received by the victim. As demonstrated in the different scenarios, tracking attackers activities could be a tedious task. However, all activities convey to a victim. Hence, keeping track of both attacker and victim activities are important. On the attacker side, tracking the illegal activities as well as the intermediary activities would be crucial. Two major aspects will be considered at this stage: classifying known attacks into attack classes and classifying known network activities into behaviours

4.6.1 Attack classification for multistage detection

Classifying attacks is a challenging task as for an ideal classification a full knowledge of all attacks would be required. Various attacks classifications have been already published [133-143]. Each of these methods of classification has a different approach. However, the DARPA classification method [144] was considered for discussion as it was one of the first public attack classification methods. Five attacks categories were then identified:

1. Probe: the gathering of information
2. Denial of Service: Attacks that cause the system not to be available
3. Remote to Local: outside attacker targeting the local system
4. User to Root: unauthorised access
5. Data: Exfiltration of data

The above classification used with DARPA dataset was representing the attack level of period. Attack sophistication has increased and the classification that would represent such level would require to have more granularities in order to represent the finest of attacks. Another comprehensive computer attack classification was done by [145]. However, the classification suggested is exaggerated has some of the sections have nothing to do with detection. For instance, one of the sections is “attack by automation” with the different automations being automatic, semi-automatic, and manual. The author did not see any practical application of such classification. The classification that the author suggests below is geared at improving detection and mitigation.

Based on the scenarios used earlier in this chapter, it appears that the only activity that was a regular suspicious behaviour from the attacks was the scan. Most security systems unfortunately disable scan traffic as scans are generally considered pure noise without much security importance. In the classification proposed here the author makes a deliberate choice to include scans as important stages of attacks. The classification used by the author is a modification of what was proposed by [146]. Even though his classification made more sense to the author, there was a level of granularity missing for a better management of attacks.

For successful detection of malicious activities, the various attack classes have been defined and considered:

a. Reconnaissance

b. Network mapping

c. Port scanning and banner grabbing a host

d. Vulnerability identification

e. Exploitation

f. Privilege escalation

g. Rootkit installation

h. Hiding tracks

i. Monitoring

j. Using unauthorized privilege gained for benefit

k. Botnet traffic

l. Silent Response

Reconnaissance:

Reconnaissance is a well know steps in the from the ethical hacker methodology [147]. In this class of attack, the malicious users do not necessarily need to have direct access to the target system. The attacks generally comprise DNS queries, WHOIS, Ping, Finger, Traceroute, and running sniffers. Also, Google can be used for this class of attack with command such as !Host=*.* intext:enc_UserPassword=* ext:pcf to steal usernames and passwords. In a more generic way, the following types of attack will fall in this class

- tcp connect scan
- tcp syn scan
- tcp fin scan
- tcp Xmas Tree scan
- TCP Null scan

- TCP ACK scan
- TCP Windows scan
- TCP RPC scan
- UDP scan

Network Mapping:

In this class of attacks, malicious users attempt to build a picture of the network they are targeting. This is generally done by using NMAP. TCP scans can also be used for this purpose when the malicious users do not have direct access to the physical network. Also, if the network is infected by a worm or Trojan, the same objectives can be reached.

Port Scanning and banner grabbing

This class of attacks is a step that is generally looked at as not very critical. Yet, it could be the only step an attacker would perform to know what vulnerabilities exit on the victim system. The vulnerabilities database is a good source for hacker as well as for other users that want to protect their system. For example, knowing that a system use Windows XP SP1 give a good indications of the problem he can have. Banner grabbing leads to vulnerability identification.

Vulnerability identification

In this class, the malicious users use the information collected during the banner grabbing to identify vulnerabilities. Vulnerability information is widely and freely available either from the

Exploitation

Once the vulnerabilities have been identifies, they are widely and freely available tools and videos that anyone can access in order to take advantage of the problem found on the system. A common tool is Metasploit. In a more generic way, the class is subject to the following type of attacks

- shellcode-detect
- inappropriate-content
- rogue ssl certificate
- system-call-detect

Privileged escalation

In this class of attack, the malicious users will try to gain administrator/root access. This class is generally subject to the following type of attacks:

- attempted-admin
- attempted-user-login
- ftp failled login attempts

Rootkit installation

Once the access as root or administrator has been achieved, the malicious user will then install tools that will allow exfiltration of information or exploitation of the victim system. This class is subject to the following types of attacks:

- web-application-attack
- trojan-activity
- suspicious-filename-detect
- web-application-activity
- misc-attack
- malware detect
-

Hiding attacks

Skilled malicious users will erase or attempt to erase any trace of their activities. This level of attacks is not always visible from the IDS. However, these attacks can be detected by using a “radar”, a software agent, that will track the changes to system files and system parameters. When a trace file is deleted, the radar will send an alert to syslog. Syslog will be configured to send certain or all alert type to the MIDaPS, the IDS designed in this research.

Monitoring

Malicious users always ensure that the target system is still in the loop. Hence monitoring is practice generally by the use of ICMP request.

Using unauthorized privilege gained for benefit

In this class of attack, malicious users take social advantages of other users by stealing credit card information for example. This is generally done by fake email that will ask the user to submit his back details or purchase a fake antivirus. Fishing is the typical attack type of this attack class.

Botnet traffic

The author chose to put botnet traffic into a separate category as specific studies are done to identify botnet activities.

Silent Response

This attack class is based on error messages that are generally received as normal behaviour. Yet these messages are generally good indications that an attack is taking place.

Attacks type within the attack classes identified above are designed to be recognised either by signatures or by algorithms based on a deviation from a normal behaviour. However, the mitigation method found in this thesis have identified that some attacks are performed by using less illegal actions that illegal one. For instance, one attack will use two usual illegal activities whilst using 4 legal steps.

4.6.2 Behaviour classification

Alongside attack classes, key network behaviours have been defined as to trace the full attackers' activities. Network behaviours are steps that do not violate any protocol violation or exploit any vulnerability, but rather, they are steps that attackers have to go through to exploit vulnerability or to complete an attack. The classification done on malicious behaviour is based around the services found in a computer system. For instance, the FTP service would generate the following behaviour:

- TCP Connection
- ftp upload from different server
- ftp download for different server
- ftp download in action
- ftp upload in action
- PSH flag irregular used
- ftp traffic non ftp port

Services based on web traffic would be:

- file download via http
- file upload via http

- HTTP traffic non http port
- HTTP Proxy in used
- Socks Proxy Server in use

Services based on computer status would be:

- admin activity
- computer reboot
- policy-violation
- username creation
- username deletion
- new log file created
- disable antivirus

Services based on email communication would be:

- fishing email identified
- email received
- email sent
- SPAM received
- Attachment (suspicious) download

Sometimes, computer systems are abused by using non regular activities. These will fall into:

- OS Unknown
- incoming distributed port
- incoming distributed IP
- non-standard-protocol

4.6.3 Interpretation

In this scenario, an attack taking advantage of Joomla, a Content Management System (CMS), will be described whilst putting into display both attack class and behaviours.

i. intitle:"Joomla - Web Installer"

   Here, the attacker will use Google to identify the vulnerable system. This step does not hold any illegal action. However, the objective of getting this sort of information is not from a good motive.

ii. create mysql db to another server

After successfully identifying victim systems, the attacker will prepare a remote server with MySQL to which the database will be directed during the installation. This stage is somehow legal even though it contains level of Xsite Scripting. Having said that, there is an anomaly to install the file of a website on one server and the database on another sever.

iii. Install joomla

At this stage, the attacker will install Joomla as it is normally done. This stage is 100% legal.

iv. install shell component joomla

Installing a component in Joomla is absolutely legal and it is common practice as Joomla CMS is based around components.

v. install file EXTPLORER joomla component

Xplorer is one of the best and useful components of Joomla to install. It offers an excellent interface for uploading files to the remote computer without the need of any FTP information. This component is free to download and to use. The malicious user can then upload any file that he plans to use with minor restriction (i.e. 10MB in size)

vi. upload remote exploit code though joomla (like netcat)

This component allows administrator users to upload virtually any file as long as they are not restricted. In the event of a file being blocked, the administrator has full rights to modify and lift the restriction of file type that can be uploaded.

vii. using shell component, open listening port with access to command line

Using the shell component, the malicious user could open ports with program such as netcat which will wait for instructions from the remote user.

viii. remotely control the victim computer

Once the remote user executes the appropriate command, he then takes control of the remote system. This can be done via command line or even via graphical User Interface. Actions performed here could be identified by signatures

ix. install IRC client // steal information on the computer // install trojan // key logger

After taking control of the remote system, the malicious user can then install IRC client, or can copy all existing data (or a particular folder). Alternatively, the attacker can install Trojan, key logger. Most of the actions performed at this stage are malicious and could be identified by using signatures.

x. Join botnet

As a major step, the computer can be register to a botnet either for SPAM, DDOS, or any other malicious purpose. Joining an IRC server is not necessarily a malicious action. However, well designed signatures can identify the difference between a normal IRC client try to join a chat and a command used to register computer systems as zombie.

A representation of the Joomla scenario putting in perspective the attack classes and network behaviour is represented in Figure 4-22

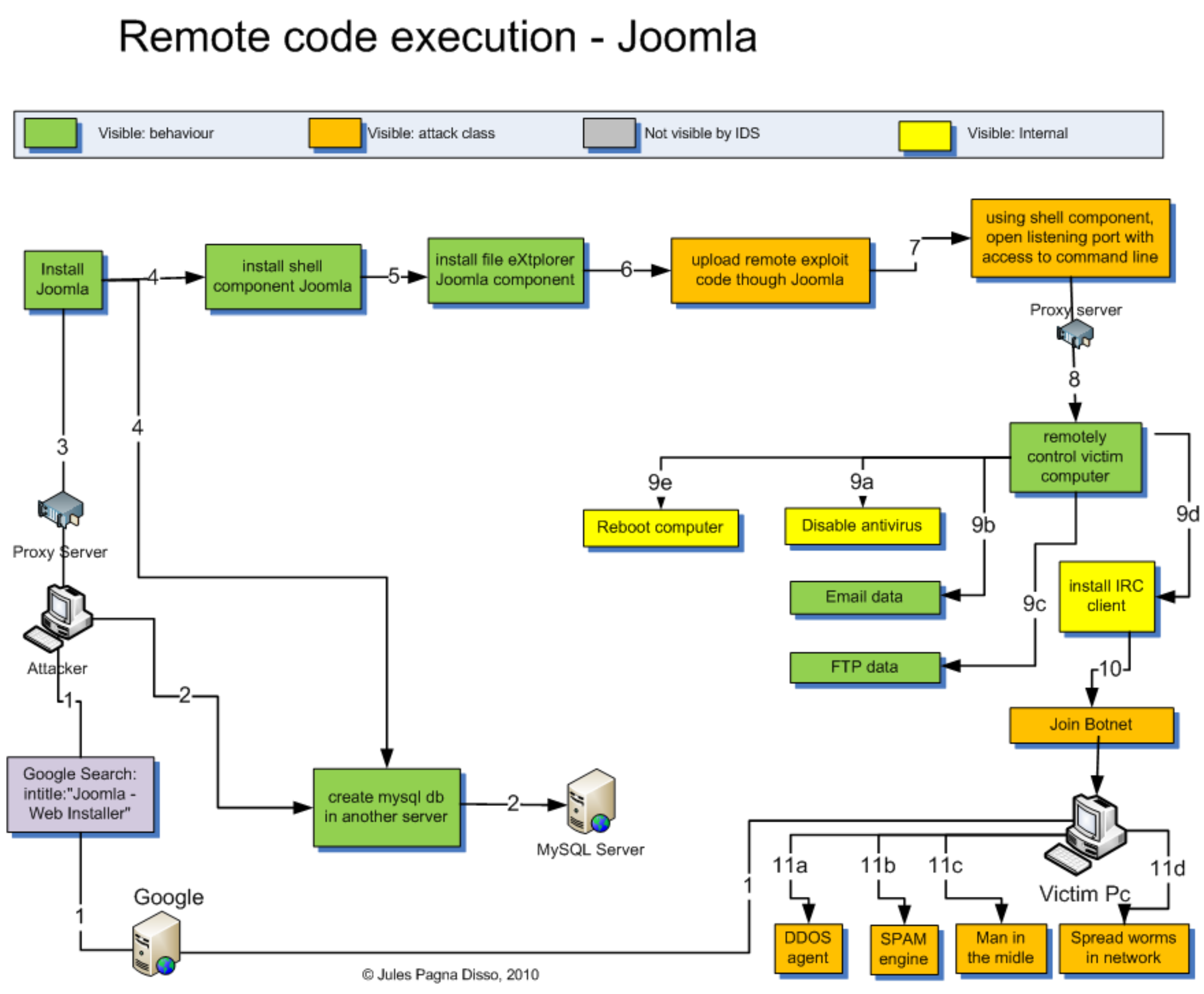

Remote code execution - Joomla
Visible: behaviour
Visible: attack class
Not visible by IDS
Visible: Internal
Install Joomla
install shell component Joomla
install file eXtplorer Joomla component
upload remote exploit code though Joomla
using shell component, open listening port with access to command line
Proxy server
remotely control victim computer
Reboot computer
Disable antivirus
Email data
FTP data
install IRC client
Join Botnet
Proxy Server
Attacker
Google Search: intitle:"Joomla - Web Installer"
create mysql db in another server
MySQL Server
Google
Victim Pc
DDOS agent
SPAM engine
Man in the midle
Spread worms in network
1
2
3
4
5
6
7
8
9a
9b
9c
9d
9e
10
11a
11b
11c
11d
© Jules Pagna Disso, 2010

Figure 4-22: Remote code execution - Joomla scenario

In the light of Figure 4-22, the victim system does not have any knowledge of its information that Google made public. Once the information about victim systems are received, there is no indication that something malicious is happening. Installing components in Joomla is a normal procedure for setting the CMS. However, there should be reason for concern if a known exploit is uploaded to any server even if the reasons are legitimate. In addition, sending shell code over the network especially over the Internet must be a concern. Even if the purpose of the shell code is unidentified, this action should be flagged a serious security threat. Remotely controlling a computer system is becoming more and more common. There should not necessarily be a concern when a computer is remotely accessed. However, if the system being remotely controlled has recently been scanned by a host or especially by a known proxy; a serious flag should be raised to stop the on-going action.

4.6.4 Attack tree

Modelling attacks using a tree structure are not new and were first introduced by Schneier[148]. In this research attacks will be used to represent possible sequencing of attacks processes. A similar structure was used by [149]. The structure defined in this research will be used as part the architecture built to defend against multistage attacks. Various elements need to be considered when building attack trees

4.6.5 Threat modelling process

[150] defines a process used to model threats for web application. However, that process is very specific to web application. In the work carried for this research, a more general threat modelling process is defined. Some of the steps are similar but their content is very different.

Identify assets: identifying assets is the first and probably one of the most important steps to achieve when thinking about security. What needs protection needs to be clearly defined in order to provide relevant security. When testing Snort performance, it was identified that Snort did not have a enough information on the system that it was protecting. This resulted in a loss of 84% of the time Snort was using to run through the rules. A solution was provided for this earlier It is important to have a full list of all servers, networks, and any specific item connecting to the network in order to provide the most efficient configuration

Create system architecture: During the analysis of multistage attacks, it was noted that the attacker was taking advantage of specific weaknesses that did not, in most cases, have a solution ready in Snort. Creating system architecture resolved into creating the list of servers with their corresponding IPs; the services used i.e. the ports number that are opened; the access time for each of the servers if this information can be known. This will help to predict any unusual behaviour and consequently provide the corresponding solution. In addition, the list of IPs that will have access to the system remotely with root privileges should be known in advance.

Map interaction between systems: A clear picture of the communication taking place between the different systems should be known in advance. Each participant IP and its associate participant ports should be identified as well as the data exchange that takes place between systems. The following questions should be answered:

- Who connects to whom?
- What type of connection is it i.e. uploads-downloads?

Identifying threats: Three types of threat can be distinguished: network, host, and applications. Security does come without effort. Securing a system demands effort and time. All participant hardware should be identified as well as their corresponding threats. A comprehensive list of hardware and their related threats should be provided in order to cater for the named threats and provide solution.

A comprehensive list of software installed on the target computer system should be produced along with their related possible problem. For example, having SQL server installed would mean potential SQL Injection attacks.

At the host level, if the IDS is meant to be protecting a network, each host should be identified and when possible their possible problems. For instance, having a Windows Operating system would mean keeping a close eye Microsoft Security Bulletins.

Create attack tree:

The structure and semantics used in this research are closed to the one used by [150]. In the light of recent attacks and analysis performed in this research, attacks are very similar to normal behaviour. However, there are sometimes strong strop that indicate an attack is happening. These steps were not identified by the work of [151]. The author introduced these steps as critical link and critical path. Also, [151] use a root node as the ultimate goal of the attack. Yet, in this work, the author defined attack tree based on the model of Aho-Corasic algorithm.

The model used to create trees is based on seven elements: the root node, active node, passive node, critical link, connectors, critical link, and critical path.

The active node is a step in the attack process that indicates a step that can standalone as an attack

The passive node is a step that leads to an attack or a step that is important in the attack process but does not represent an attack on itself

The connectors indicate whether two linked step are compulsory or optional

A critical path represents a number of steps that represent an attack. Typically, the IDS should be set to fire an attack when a critical link is completed.

A critical link is an important step to an attack.

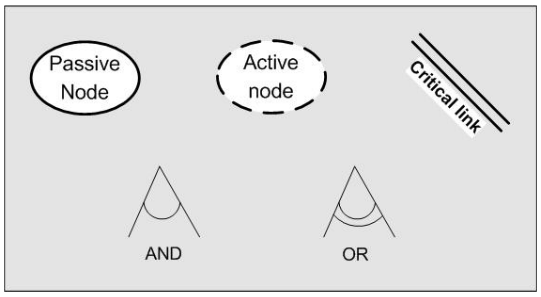


Figure 4-23: Attack Tree Objects

Attack trees can be simple or very complex depending on the nature of the attack. In the scenario that follows, a typical process of malware download is presented in Figure 4-24. The process for a malware download is one of the processes that is commonly used to bypass IDS and other security devices. A typical step by step would be:

Step 1: a user visits a compromised website. This process is not always visible to IDS yet there are public lists available that can be used to filter such as the Google safe browsing initiative. Good scan URLs that host or have host “badware” in the recent past [152].

Step2: the page requested by the “innocent” user is then redirected to another page that will be used to download malware.

Step3: obfuscated or encrypted JavaScript is then downloaded to the visitor’s computer without his knowledge. This step is not easily visible by IDS as most IDS or security software do not deal with encrypted traffic.

Step4: the code downloaded to the visitor's computer generally perform preliminary task to the attacks

Step5: the visitor's computer reports to the attacker

Step6: the attacker scans the visitor's computer for any possible vulnerability.

Step7: the appropriate malware is downloaded to the victim's computer based on the result of the scan

Step8: once the malware has been downloaded, the computer is open to any sort of attack.

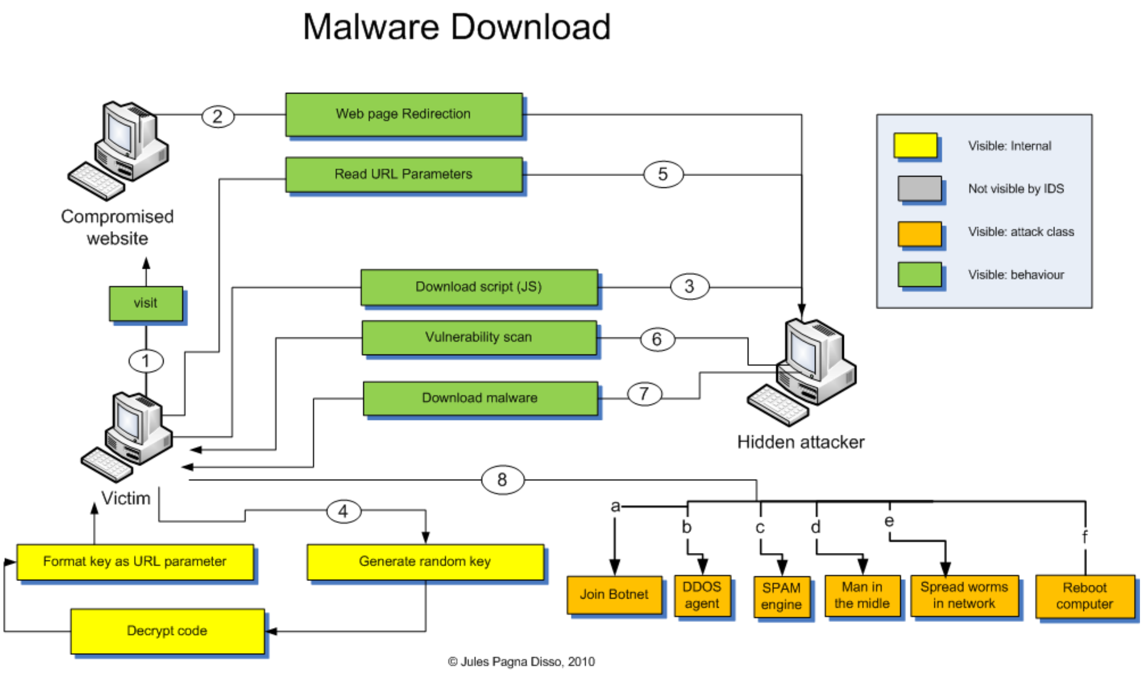


Figure 4-24: Malware download

The attack tree would then be:

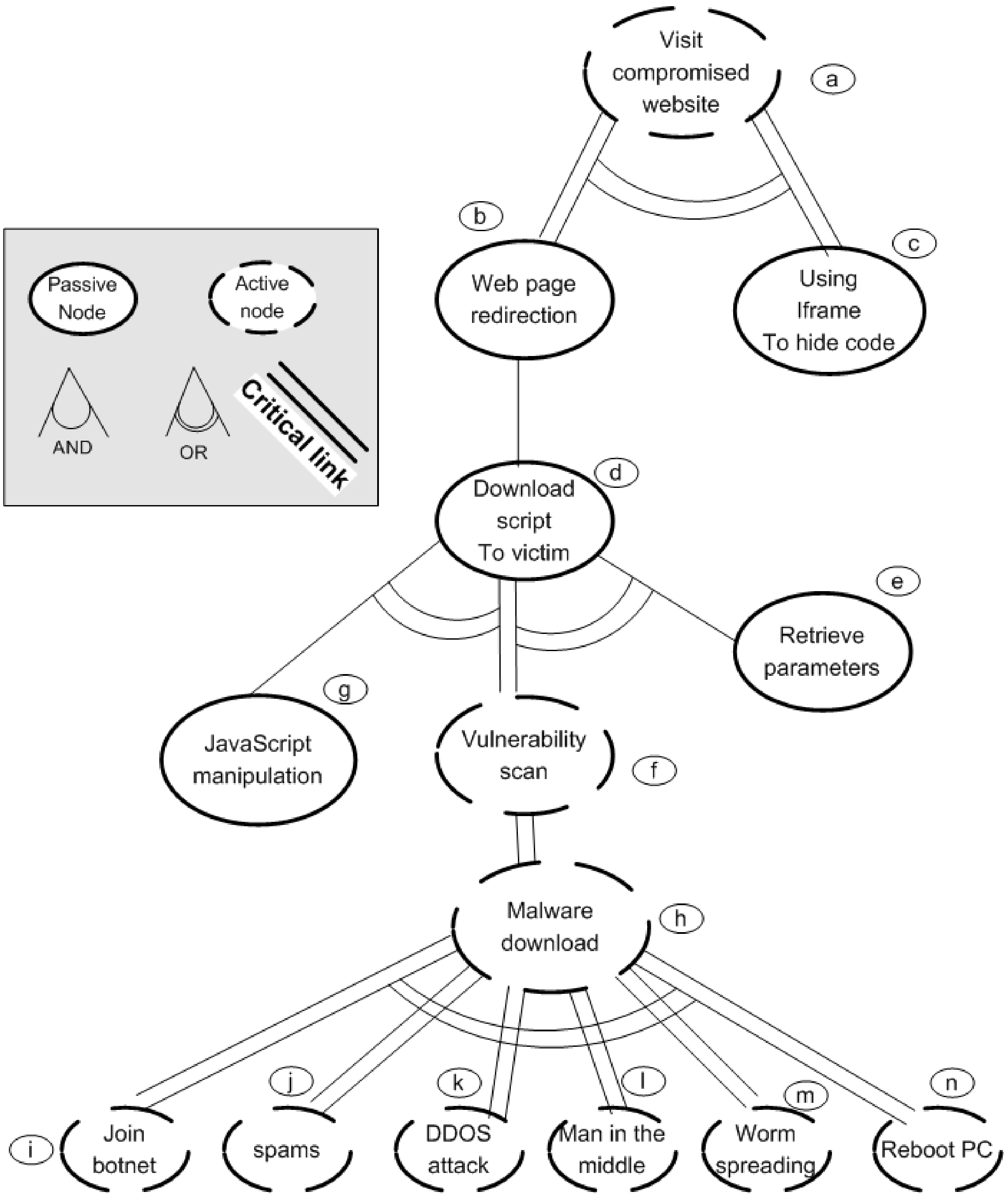


Figure 4-25: attack tree - malware download

Many attack paths can be deducted from the above tree:

<a,b><b,d><d,g><g,h><g,h><h,i>

<a,b><b,d><d,g><g,h><g,h><h,j>

<a,b><b,d><d,g><g,h><g,h><h,k>

<a,b><b,d><d,g><g,h><g,h><h,l>

<a,b><b,d><d,g><g,h><g,h><h,m>

<a,b><b,d><d,g><g,h><g,h><h,n>

<a,b><b,d><d,f><g,h><g,h><h,i>

<a,b><b,d><d,f><g,h><g,h><h,j>

<a,b><b,d><d,f><g,h><g,h><h,k>

<a,b><b,d><d,f><g,h><g,h><h,l>

<a,b><b,d><d,f><g,h><g,h><h,m>

<a,b><b,d><d,f><g,h><g,h><h,n>

<a,b><b,d><d,e><g,h><g,h><h,i>

<a,b><b,d><d,e><g,h><g,h><h,j>

<a,b><b,d><d,e><g,h><g,h><h,k>

<a,b><b,d><d,e><g,h><g,h><h,l>

<a,b><b,d><d,e><g,h><g,h><h,m>

<a,b><b,d><d,e><g,h><g,h><h,n>

<a,c><b,d><d,g><g,h><g,h><h,i>

<a,c><b,d><d,g><g,h><g,h><h,j>

<a,c><b,d><d,g><g,h><g,h><h,k>

<a,c><b,d><d,g><g,h><g,h><h,l>

<a,c><b,d><d,g><g,h><g,h><h,m>

<a,c><b,d><d,g><g,h><g,h><h,n>

<a,b><b,d><d,f><g,h><g,h><h,i>

<a,c><b,d><d,f><g,h><g,h><h,j>

<a,c><b,d><d,f><g,h><g,h><h,k>

<a,c><b,d><d,f><g,h><g,h><h,l>

<a,c><b,d><d,f><g,h><g,h><h,m>

<a,c><b,d><d,f><g,h><g,h><h,n>

<a,c><b,d><d,e><g,h><g,h><h,i>

<a,c><b,d><d,e><g,h><g,h><h,j>

<a,c><b,d><d,e><g,h><g,h><h,k>

<a,c><b,d><d,e><g,h><g,h><h,l>

<a,c><b,d><d,e><g,h><g,h><h,m>

<a,c><b,d><d,e><g,h><g,h><h,n>

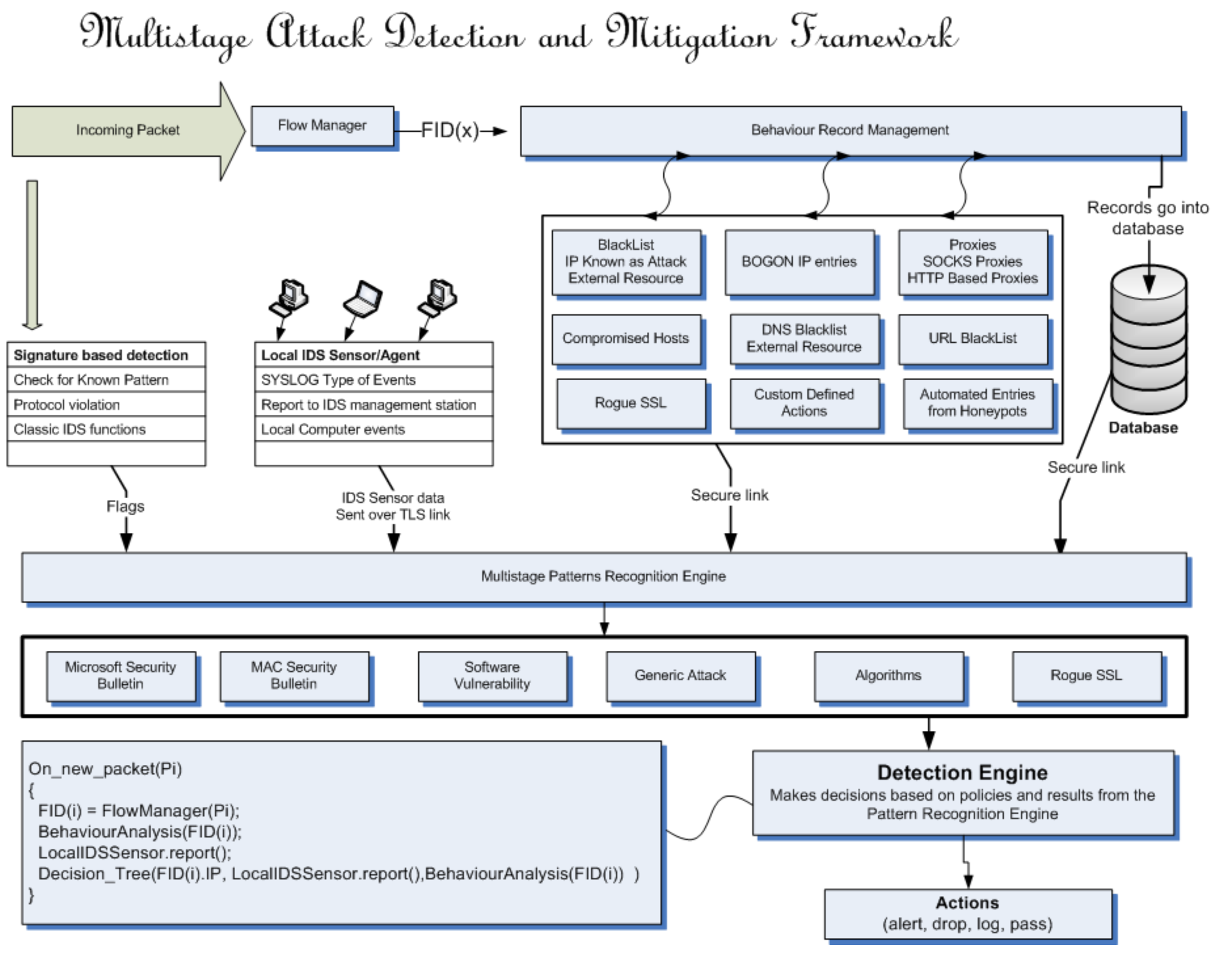
Documenting threats: documenting the threats will help in the configuration of the target system. [150] suggests the options threat description, threat target, risk, attack techniques, and countermeasures. An example of threat documentation would be:

Table 11: Threat description sample

Threat Description	Attacker to deceive IDS by using spoofing IP
Threat target	SQL Server – SQL Injection
Risk	Steal valuable information
Attack techniques	Use multiple virtual machine to perform each step
Countermeasures	Use attack tree to link the different actions

4.7 Multistage attack detection and mitigation framework

Our multi-stage attack detection and mitigation framework will look at attacks from various angles. When packets arrive, they are checked against known patterns. If a match has occurred, the flag will be raised. Concurrently, each packet will be assigned the flow ID and then pass those IDs (FID(x)) to the Behaviour Record Manager. The behavioural record manager will tag each FID(x) to a specific action. At the same time the local IDS sensor will report to the detection engine. The detection engine will check for existing patterns against a database of patterns already defined. Various algorithms can be applied in the detection engine. For instance, any IP that is flagged with any critical path will be blocked and added into the blacklist.


Multistage Attack Detection and Mitigation Framework
Incoming Packet
Flow Manager
FID(x)
Behaviour Record Management
BlackList IP Known as Attack External Resource
BOGON IP entries
Proxies SOCKS Proxies HTTP Based Proxies
Compromised Hosts
DNS BlackList External Resource
URL BlackList
Rogue SSL
Custom Defined Actions
Automated Entries from Honeypots
Records go into database
Database
Signature based detection
Check for Known Pattern
Protocol violation
Classic IDS functions
Local IDS Sensor/Agent
SYSLOG Type of Events
Report to IDS management station
Local Computer events
Flags
IDS Sensor data Sent over TLS link
Secure link
Secure link
Multistage Patterns Recognition Engine
Microsoft Security Bulletin
MAC Security Bulletin
Software Vulnerability
Generic Attack
Algorithms
Rogue SSL
Detection Engine
Makes decisions based on policies and results from the Pattern Recognition Engine
Actions
(alert, drop, log, pass)

```
On_new_packet(Pi)
{
  FID(i) = FlowManager(Pi);
  BehaviourAnalysis(FID(i));
  LocalIDSSensor.report();
  Decision_Tree(FID(i).IP, LocalIDSSensor.report(),BehaviourAnalysis(FID(i))  )
}
```

Figure 4-26: Functional diagram multistage attack detection and mitigation framework

4.8 U-Case

In their latest security intelligence report [153], Microsoft describes a typical distribution scenarios used by botnet, when spreading the attacks. The attack process will start by a SPAM message sent by a bot. The message sent out contains a link to malicious software. The victim user is convinced to click on the link within the message. Social engineering is generally used to convince user to click. The fake message will be designed around very common theme, generally a theme current to the society such as Christmas. The victim users the download the malware either by downloading directly the malware to his computer or by opening a crafted page, that contain all the necessary to exploit browser exploits. This technique is generally referred to as “drive by download” [153]. Alternatively, the victim user is sent the malware directly by attachment.

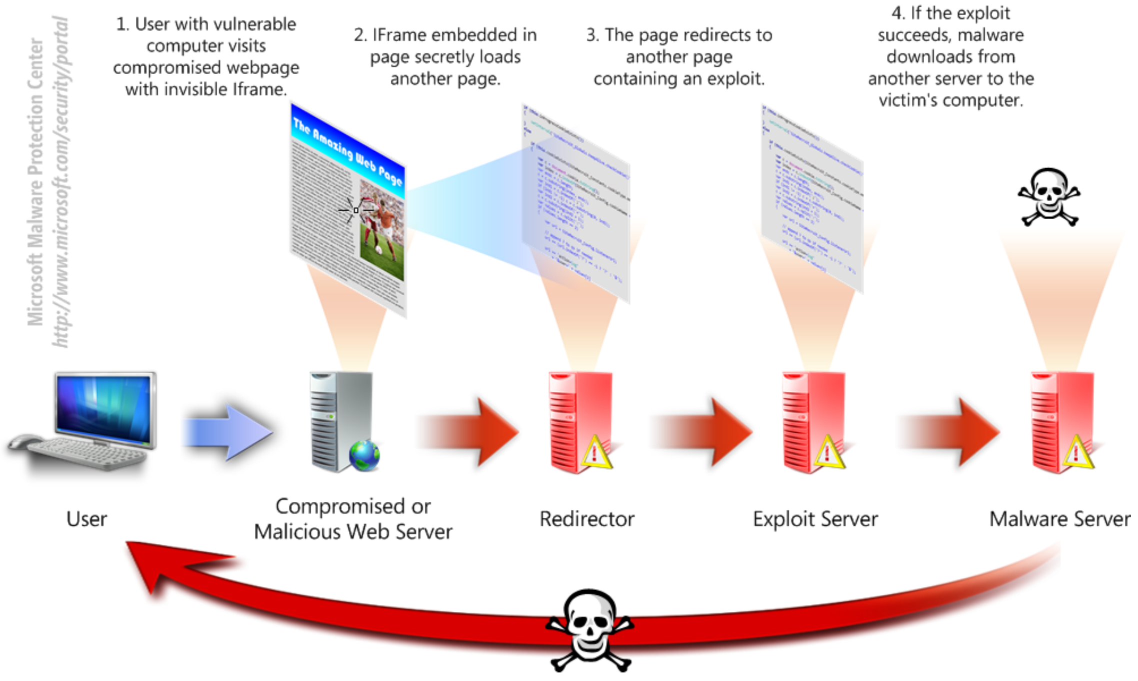

Microsoft Malware Protection Center
http://www.microsoft.com/security/portal
1. User with vulnerable computer visits compromised webpage with invisible Iframe.
2. IFrame embedded in page secretly loads another page.
3. The page redirects to another page containing an exploit.
4. If the exploit succeeds, malware downloads from another server to the victim's computer.
The Amazing Web Page
User
Compromised or Malicious Web Server
Redirector
Exploit Server
Malware Server

Figure 4-27: Drive by download scenario [153]

Detection scenario

a. Usr1 visit page – this action is classed as a behaviour (i.e. with or without risk). If the page is recorded as a page previously used for malicious purposes, the action will be recorded as: usr1 → visit malicious page. In the former case, the action is considered precursor to attack.

b. Usr1 → visit page with iframe, if the page is not encrypted. This action is considered as a potential danger and precursor to attack. Alternatively, the page can be encrypted. If the page uses a popular encryption technique, the encrypted block will be decrypted and the iframe will be revealed. If the page cannot be decrypted, the action is recorded as: user1 → encrypted page identified. The two actions: user1→ visit malicious page and usr1: iframe are good enough to raise an alert. At this level there is a very little chance that the alert is a false positive as action 1 has been recorded previously as malicious. However, if action 1 was only recorded as “visit page” and action 2 recorded as “iframe”, no serious flag will be raised. An alert could indicate a potential danger and not an imminent danger. Since both action 1 and action 2 have two variants, there is a total of 4 possibilities.

c. Usr1 → page redirect. Redirecting a page has nothing in itself that cause a security threat. However, based on the “drive by download” scenario and in the light of previous actions, this action 3 could be an indicator that the malicious user is on its way to complete a drive by download process. Using the doubtful quality of the website visited in action1, the attack can be blocked at this level. Taking this attack further, another action could be recorded

d. Usr1 → download form encrypted page, or download from website previously recorded as malicious. When the download is completed, more actions are likely to be produced.

e. Systems file change in usr1. In this action, the malicious user should have had access to the usr1 system and possibly take control of it.

In all, the case can be interpreted as follow:

4.9 Conclusion

In this chapter, honeynet have evidenced the dark side of the Internet. Sophisticated attacks were captured, modelled to create a strong detection and mitigation engine for complex multistage attacks. Multistage attack referring to attacks performed in multiple steps. The design presented here is geared at multicore architecture to ensure the maximum performance possible. The big number of features could however generate many problems related to performance it this architecture is implemented in a top down way. There is a risk that some features performance in this architecture could impact on other features. Many studies need to be done in relation to the interdependence performance for each of the element here identified for the detection and mitigation of complex attacks.

5 Distributed Denial of Service Attack (DDOS): Detection and Mitigation

5.1 Introduction

Recent recorded attacks have indicated that the level of sophistication used by the malicious users have risen significantly. As a consequence, the activities of malicious users are still very high especially those of botnets Figure 5-1. In the previous chapter, a generic attack detection system was built for multistage attacks. Malicious users employ methods that are almost identical to legitimate users' actions. In Figure 4-22 the author demonstrates that an attacker can take control of a whole network without much indication of illegal activities.

In this chapter, analysis of live traffic capture will be done. Based on the analysis done here, DDOS detection and mitigation solution will be proposed. Both corporate networks and honeynet data will be used for the analysis. In addition, complex detection algorithms will be written to support the proposed detection and mitigation architecture. The work presented in this chapter will be used as a module, an extension, to the core IDS framework that will be proposed later

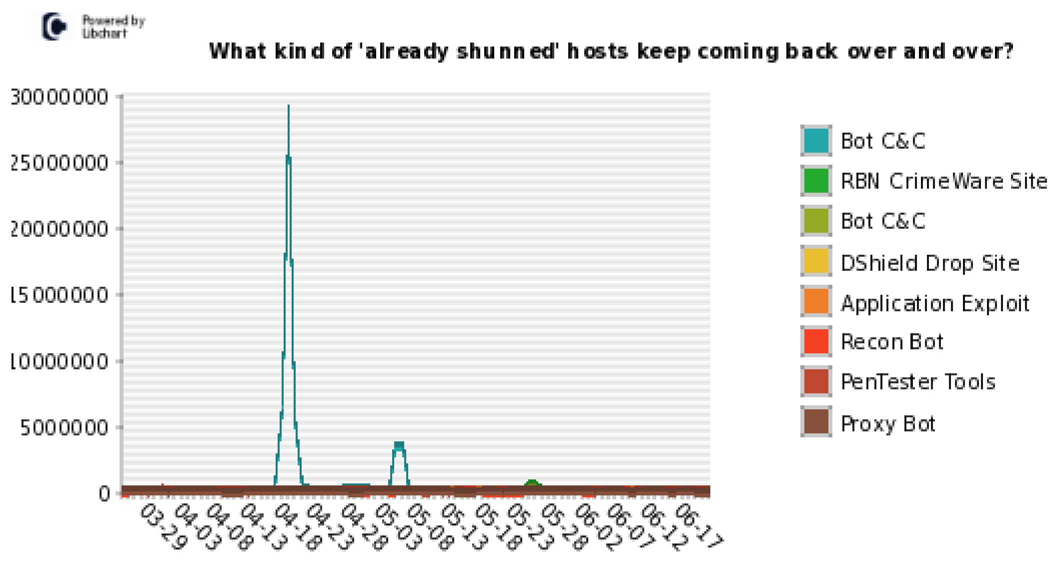


Figure 5-1: Bot activity June 2010

5.2 Threat analysis: real live capture of DDOS attacks revealed

This section presents an analysis of a capture that was done in a corporate Lab using a DeMilitarised Zone (DMZ). Over a fortnight, packets were captured and a summary of the findings are discussed in bellow. The first capture is based on UDP. Looking at the packet structure, the IP header has not been violated and has remained equal to 20bytes. Not only was the header conformed to the protocol description, but the remainder of the packet had not violated any description from the RFC describing UDP packets. The packet at a frame 8 shows of one of the capture shows:

```
0000   00 18 39 dd 6c a2 00 03 0d 7c 5a d7 08 00 45 00  ..9.l....|Z...E.
0010   00 1e 39 e3 40 00 80 11 00 00 c0 a8 0f 66 0a 9c  ..9.@........f..
0020   87 55 ef a8 47 86 00 0a 62 1b 87 00              .U..G...b...
```

The next packet along shows:

```
0000   00 18 39 dd 6c a2 00 03 0d 7c 5a d7 08 00 45 00  ..9.l....|Z...E.
0010   00 1e 3a 04 40 00 80 11 00 00 c0 a8 0f 66 0a 9c  ..:.@........f..
0020   87 55 ef a8 47 86 00 0a 62 1b 87 00              .U..G...b...
```

As shown in this offset hexadecimal text representation of the packet, one would notice their integrity to the protocol definition. Looking deep into the packet, one could notice that the payload (data) has not changed "**87 00**". About 10 packets were recorded per transaction at this stage of the capture process. An extract of Wireshark capture shows the consistency in field throughout the early stages of the attack. A quick summary of the capture of the surrounding packets shows

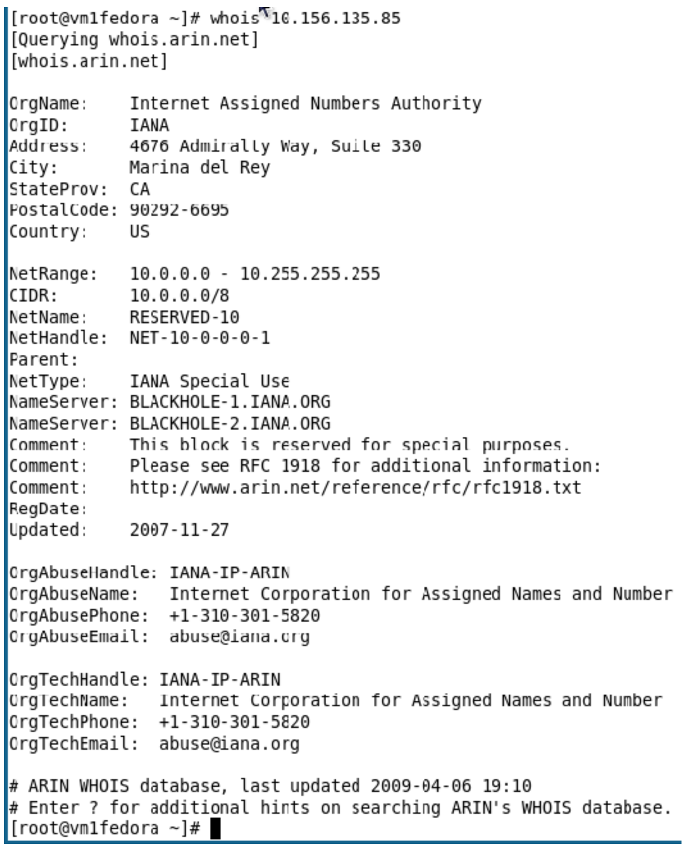
- data are sent from the local source to foreign IP address
- the source port is the same = 61352
- destination port is the same 1831
- payload size = 2 bytes (the same size and payload content remained the same)

Using small packets have proven to be very efficient in DDOS attack has they consume a lot of CPU [151], [154], [155] whereas big packets consume bandwidth.

Frame	Time	SourceIP	Destination IP	Proto	Comments
7	1.003061	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
8	1.251077	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
9	1.500085	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
10	1.749109	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
11	1.998123	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
12	2.248137	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
13	2.501157	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
14	2.752169	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
15	3.002183	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
16	3.2552	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310
17	3.491208	192.168.15.102	10.156.135.85	UDP	Source port: 61352 Destination port: 18310

Table 12: UDP traffic showing DDOS attack

Looking at the IP address, it is clear that the IP belongs to an IANA Reversed IP range that should not normally appear on the Internet routing table (Internet Assigned Numbers Authority, 2005.) 10.0.0.0/8 block is reserved to be used in private networks. Hence no address from that range should appear on the Internet Table (RFC 3330)[156]



```
[root@vm1fedora ~]# whois 10.156.135.85
[Querying whois.arin.net]
[whois.arin.net]

OrgName:    Internet Assigned Numbers Authority
OrgID:      IANA
Address:    4676 Admiralty Way, Suite 330
City:       Marina del Rey
StateProv:  CA
PostalCode: 90292-6695
Country:    US

NetRange:   10.0.0.0 - 10.255.255.255
CIDR:       10.0.0.0/8
NetName:    RESERVED-10
NetHandle:  NET-10-0-0-0-1
Parent:
NetType:    IANA Special Use
NameServer: BLACKHOLE-1.IANA.ORG
NameServer: BLACKHOLE-2.IANA.ORG
Comment:    This block is reserved for special purposes.
Comment:    Please see RFC 1918 for additional information:
Comment:    http://www.arin.net/reference/rfc/rfc1918.txt
RegDate:
Updated:    2007-11-27

OrgAbuseHandle: IANA-IP-ARIN
OrgAbuseName:   Internet Corporation for Assigned Names and Number
OrgAbusePhone:  +1-310-301-5820
OrgAbuseEmail:  abuse@iana.org

OrgTechHandle: IANA-IP-ARIN
OrgTechName:   Internet Corporation for Assigned Names and Number
OrgTechPhone:  +1-310-301-5820
OrgTechEmail:  abuse@iana.org

# ARIN WHOIS database, last updated 2009-04-06 19:10
# Enter ? for additional hints on searching ARIN's WHOIS database.
[root@vm1fedora ~]#
```

Figure 5-2: IANA record showing private address related information

The traffic generated by the malicious user at first glance seems legitimate and good. However, giving that a reserved IP from IANA is used, the same traffic that was supposedly legitimate and “clean” is no longer “clean” as the IP used should not appear in the routing table. Ensuring that IPs that should not appear in the routing table are not used in the Internet communication is a recommended feature to implement in system security as this will reduce a lot of unwanted traffic [157].

Snort can be used to report on any occurrence of such traffic with the rule:

Rule 1

```
alert UDP $HOME_NET any -> 10.156.135.85 61352 (msg:"UDP flooding – DDOS attack"; classtype:ddos-attack; reference: threshold:type both, count 10, seconds 1, track by_dst;  sid:; rev:1;)
```

Similar traffic was captured in another instance and is represented in the table below:

Frame	Timing	SIP	DIP	Proto	comment
21	0.036635	192.168.15.102	149.254.200.237	UDP	Source port: 38140 Destination port: 19304 [UDP CHECKSUM INCORRECT]
22	0.26065	192.168.15.102	10.156.135.85	UDP	Source port: 38140 Destination port: 18310 [UDP CHECKSUM INCORRECT]
23	0.262307	192.168.15.102	149.254.200.237	UDP	Source port: 38140 Destination port: 19285 [UDP CHECKSUM INCORRECT]

Table 13: Table showing UDP DDOS attack - Same port for multiple IPs

From the table above, another entry of the IANA reserved IP was used. The payload of the three packets was identical and equal to “87 00”. A quick search on the other IP used in is known to be from Tmobile . Given the size of the packets, 22 bytes, the attack aimed at exhaust system resources [158], [159].

Hostname	**Country Code**	**Region Name**	**City**	**ISP**
149.254.200.237	GB	Nottinghamshire	Mansfield	T-Mobile International UK Limited

Table 14: IP resolved to its country

A snort rule can be written in order to detect this attack:

Rule 2

alert UDP $HOME_NET any -> [10.156.135.85, 149.254.200.237] 38140 (msg:"UDP flooding – DDOS attack"; classtype:ddos-attack; reference: threshold:type both, count 10, seconds 1, track by_dst; sid:; rev:1;)

Rules 1 and 2 can be optimized to more generic rules in order to detect the use of IANA reserved IPs. An entry to snort variable can be added such as IANA_IP= [list of IP all IPs and ranges reserved] [160]

A more generic rule would be:

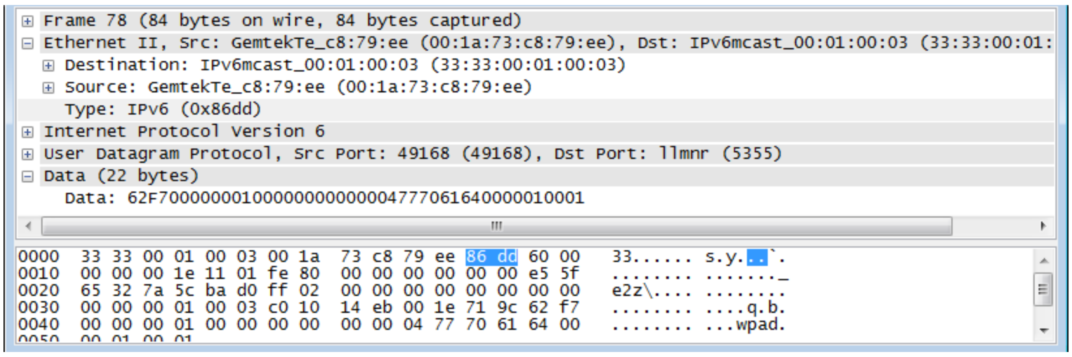
Rule 3

Alert udp $HOME_NET and → $IANA any (msg: "Private IP in routing table"; classtype:bad-traffic;reference:; sid:; rev:1))

In rule 3, any traffic using IANA reserved IPs will be detected. This solution is not applicable if the attacker use a wide range of IPs address. Currently, Emerginthreats [161] a Snort research community uses a list of IPs provided to represent the IPs that have been subject to an attack in the recent hour or so. In this scenario, one has to be victim of an attack; the attack has to be reported, then the IP will be added to the list of compromised IPs. The IP is then populated into a Snort rule that is available to download. Unfortunately, bot master infect computers randomly and the list of IP may not be the same every time the attack is launched [52], [162].There is a need to introduce a new mechanism that will understand the behavior rather than relying on a static field. Relying on a static field here would mean being successfully attacked at least once before writing the rules. Unfortunately, DDOS traffic has been generated by botnet which could involve a few thousands of computers. A recent demonstration by BBC has shown 20,000 computers infected and participing in a botnet [52], [163-165].

Scenario 2: unused protocols

A sudden change of protocol has been noticed in the communication but further investigations suggests that the communication could originate from the same malicious user. Frame 78 of one of the capture shows the presence of IPv6. A closer look at the surrounding packets show that the payload found in the packet using IPv6 is the same as the payload use in packet using IPv4. In this instance the payload was 22 bytes.

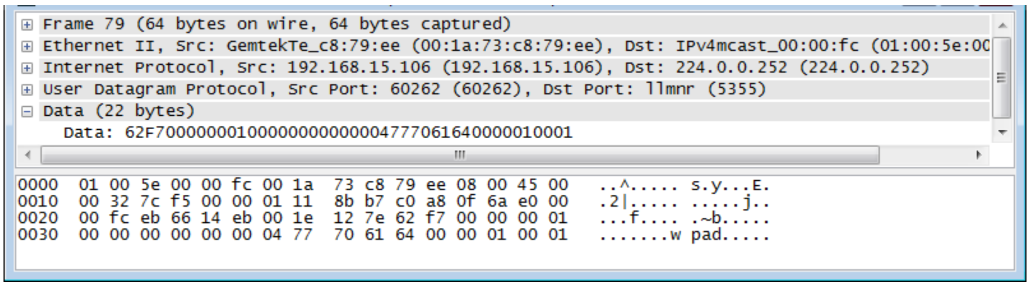


```
⊞ Frame 78 (84 bytes on wire, 84 bytes captured)
⊟ Ethernet II, Src: GemtekTe_c8:79:ee (00:1a:73:c8:79:ee), Dst: IPv6mcast_00:01:00:03 (33:33:00:01:
   ⊞ Destination: IPv6mcast_00:01:00:03 (33:33:00:01:00:03)
   ⊞ Source: GemtekTe_c8:79:ee (00:1a:73:c8:79:ee)
     Type: IPv6 (0x86dd)
⊞ Internet Protocol Version 6
⊞ User Datagram Protocol, Src Port: 49168 (49168), Dst Port: llmnr (5355)
⊟ Data (22 bytes)
     Data: 62F700000001000000000000047770616400000010001

0000  33 33 00 01 00 03 00 1a  73 c8 79 ee 86 dd 60 00   33...... s.y...`.
0010  00 00 00 1e 11 01 fe 80  00 00 00 00 00 00 e5 5f   ........ ......._
0020  65 32 7a 5c ba d0 ff 02  00 00 00 00 00 00 00 00   e2z\.... ........
0030  00 00 00 01 00 03 c0 10  14 eb 00 1e 71 9c 62 f7   ........ ....q.b.
0040  00 00 00 01 00 00 00 00  00 00 04 77 70 61 64 00   ........ ...wpad.
0050  00 01 00 01
```

Figure 5-3: DDOS attack using IPv6

A malicious user could take great advantages of poorly configured computer system environments. It is important to turn off all unused services or protocols as these can be used for the benefit of malicious users.

IPv4 view



```
⊞ Frame 79 (64 bytes on wire, 64 bytes captured)
⊞ Ethernet II, Src: GemtekTe_c8:79:ee (00:1a:73:c8:79:ee), Dst: IPv4mcast_00:00:fc (01:00:5e:00
⊞ Internet Protocol, Src: 192.168.15.106 (192.168.15.106), Dst: 224.0.0.252 (224.0.0.252)
⊞ User Datagram Protocol, Src Port: 60262 (60262), Dst Port: llmnr (5355)
⊟ Data (22 bytes)
     Data: 62F700000001000000000000047770616400000010001

0000  01 00 5e 00 00 fc 00 1a  73 c8 79 ee 08 00 45 00   ..^..... s.y...E.
0010  00 32 7c f5 00 00 01 11  8b b7 c0 a8 0f 6a e0 00   .2|..... .....j..
0020  00 fc eb 66 14 eb 00 1e  12 7e 62 f7 00 00 00 01   ...f.... .~b.....
0030  00 00 00 00 00 00 04 77  70 61 64 00 00 01 00 01   .......w pad.....
```

Figure 5-4: DDOS attack pattern in IPv4 identical to pattern in IPv6

Further analysis of frame 79 reveals that another IP, the destination IP [224.0.0.252] is a reversed IP from IANA. There is no real security mechanism offered by snort to detect unused services. An effective security mechanism would be to perform a behavioral analysis of the traffic whereby if a behavior appears from nowhere then it can be flagged.

- ***Scenario 3: Randomly generated IPs***

Having the chance to witness the attack live, the decision was made to challenge the attacker. After blocking the destination port that was used, there was no activity for few seconds. Then, the local host started sending data to various IP addresses. Interestingly, all transactions were originating from the same source IP.

Frame	Time	SourceIP	Destination IP	Proto	comments
1828	298.533096	192.168.15.102	88.102.78.141	UDP	Source port: 26811 Destination port: 33752
1829	298.535096	192.168.15.102	91.145.5.58	UDP	Source port: 26811 Destination port: 9403
1830	298.538095	192.168.15.102	163.1.175.92	UDP	Source port: 26811 Destination port: 31151
1831	298.541102	192.168.15.102	71.130.243.117	UDP	Source port: 26811 Destination port: 34610
1832	298.544098	192.168.15.102	145.97.196.243	UDP	Source port: 26811 Destination port: 48211
1833	298.547119	192.168.15.102	66.56.10.99	UDP	Source port: 26811 Destination port: 13098
1834	298.549093	192.168.15.102	24.237.34.41	UDP	Source port: 26811 Destination port: 18853
1835	298.553103	192.168.15.102	88.181.92.134	UDP	Source port: 26811 Destination port: 28818
1836	298.556096	192.168.15.102	68.226.102.249	UDP	Source port: 26811 Destination port: 37750

Table 15: Randomly generated IPs with identical packet patterns

An offline analysis of the IPs was done using MAXMIND [166] tools and the findings were as follow:

Hostname	**Country Name**	**Region Name**	**City**	**ISP**	**Organization**
88.102.78.141	Czech Republic	Jihlava	Dvur Kralove	Cesky Telecom, A.S.	XDSL NETWORK-ADSL
91.145.5.58	Sweden	Gavleborgs Lan	Edsbyn	Helsinge Net AB	Helsinge Net AB
163.1.175.92	United Kingdom	Oxfordshire	Oxford	Oxford University	Oxford University
71.130.243.117	United States	California	Alhambra	SBC Internet Services	SBC Internet Services
145.97.196.243	Netherlands	Utrecht	Utrecht	Surfnet	Stichting Sociale Huisvesting Utrecht
66.56.10.99	United States	Georgia	Acworth	Comcast Cable	Comcast Cable
24.237.34.41	United States	Alaska	Anchorage	GCI	GCI Communications
88.181.92.134	France	Midi-Pyrenees	Toulouse	Free SAS	Free SAS
68.226.102.249	United States	Arizona	Tucson	Cox Communications	Cox Communications

Table 16: IP resolved to their country name

Using the two tables above, transactions were generated from the local IP to various foreign IPs in less than a second, all these IPs being from various regions in the world. There is little explanation as to why a local IP, a local system with no special service would be sending from the same port data to nine IPs that do not seem to have much in common. Also, the data size of each of the packet was different in each frame.

Further down the attack capture, the same scenario was repeated with more intensity. Not only was the local IP was sending data to many foreign IPs from the same ports, the foreign IPs were also sending data to the local IP on the same port. This implies that the local system was being flooded at the same time it was being used to flood other systems.

Frame	Time	SourceIP	Destination IP	Proto	comments
17429	5329.836217	192.168.15.102	86.71.201.195	UDP	Source port: 26811 Destination port: 5037
17430	5329.843232	69.14.80.32	192.168.15.102	UDP	Source port: 50032 Destination port: 26811

17431	5329.85222	91.145.5.58	192.168.15.102	UDP	Source port: 9403 Destination port: 26811
17432	5329.856221	64.181.41.45	192.168.15.102	UDP	Source port: 59069 Destination port: 26811
17433	5329.862217	91.67.120.221	192.168.15.102	UDP	Source port: 8133 Destination port: 26811
17434	5329.866223	78.42.101.192	192.168.15.102	UDP	Source port: 54140 Destination port: 26811
17435	5329.869222	155.41.152.133	192.168.15.102	UDP	Source port: 9660 Destination port: 26811
17436	5329.875218	92.140.95.73	192.168.15.102	UDP	Source port: 11818 Destination port: 26811
17437	5329.895224	89.25.9.62	192.168.15.102	UDP	Source port: 18034 Destination port: 26811
17438	5329.908224	87.6.132.100	192.168.15.102	UDP	Source port: 29880 Destination port: 26811
17439	5329.911229	86.71.201.195	192.168.15.102	UDP	Source port: 5037 Destination port: 26811

Table 17: Variation of DDOS attack

Resolving the IPs to their location, the following table was created:

Hostname	Country Name	Region Name	City	ISP	Organization
69.14.80.32	United States	Michigan	Warren	WideOpenWest	WideOpenWest
91.141.5.58	Austria	Wien	Vienna	Orange Austria Telecommunication GmbH	Network of Orange Austria Telecommunication GmbH
64.181.41.45	United States	West Virginia	Weston	FiberNet of West Virginia	FiberNet of West Virginia
91.67.120.221	Germany	Nordrhein-Westfalen	Kabel	Kabel Deutschland Breitband Service GmbH	Kabel Deutschland
78.42.101.192	Germany	Baden-Württemberg	Dauchingen	Kabel Baden-Wuerttemberg GmbH & Co. KG	Kabel Baden-Wuerttemburg GmbH & Co. KG
155.41.152.133	United States	Massachusetts	Boston	Boston University	Boston University
92.140.95.73	France	Ile-de-France	Paris	France Telecom	France Telecom
89.25.9.62	Bulgaria	Plovdiv	Asenovgrad	ITD Network SA	Asenovgrad.net
87.6.132.100	Italy	Toscana	Florence	Telecom Italia	Telecom Italia
86.71.201.195	France	Ile-de-France	Paris	Neuf Cegetel	Neuf Cegetel
131.215.35.197	United States	California	Pasadena	California Institute of Technology	California Institute of Technology
88.102.78.141	Czech Republic	Jihlava	Dvur Kralove	Cesky Telecom, A.S.	XDSL NETWORK-ADSL
74.75.228.38	United States	Maine	Kennebunk	Road Runner	Road Runner
59.127.100.126	Taiwan	T'ai-pei	Taipei	CHTD, Chunghwa Telecom Co., Ltd.	Chunghwa Telecom Data Communication Business Group
137.189.133.163	Hong Kong	00	Central District	CUHK	CUHK

Table 18: Distribution of Host taking part in the attack in less than a Second

As shown by the table, the distribution of IPs by their geographical area make it difficult to find a pattern under which “normal” and legal network transactions will take place. The honeypot did not have any service running such as web server, FTP server or any other type of service that would require many connections from around the world in the same second.

- ***Scenario 4: Error Messages (Host Unreachable) – ICMP messages***

Another important element has drawn attention in the capture. UDP packets were sent using random ports. As a result, when trying to communicate with a host that is not available an ICMP message was sent back to the local host. Snort does not provide any mechanism to analyse ICMP error message yet they carry a lot of meaningful message that could help improving security [167], [168]. If the states of connections are kept, then a simple algorithm could analyse the reason why the ICMP was generated [Table 19]. In this particular case, such analysis would inform that there is a one way communication that is taking place. Most importantly, the same scenario is repeated for many IPs. Hence, a possible DDOS attack.

Frame	Time	Source IP	Destination IP	Proto	comments
14883	4303.29447	192.168.15.102	125.224.103.243	UDP	Source port: 26811 Destination port: 52749
14889	4303.610473	125.224.103.243	192.168.15.102	ICMP	Destination unreachable (Port unreachable)

Table 19: ICMP message tracking

Further investigations show that the host is real but communication on the port that was used was not accepted. Hence an indication that something could be wrong. Also, there is a tendency of systematically blocking ICMP packets yet blocking ICMP removes the stateful nature of UDP connections.

Hostname	Country Name	Region Name	City	ISP	Organization
125.224.103.243	Taiwan	T'ai-pei	Taipei	CHTD, Chunghwa Telecom Co., Ltd.	Chunghwa Telecom Data Communication Business Group

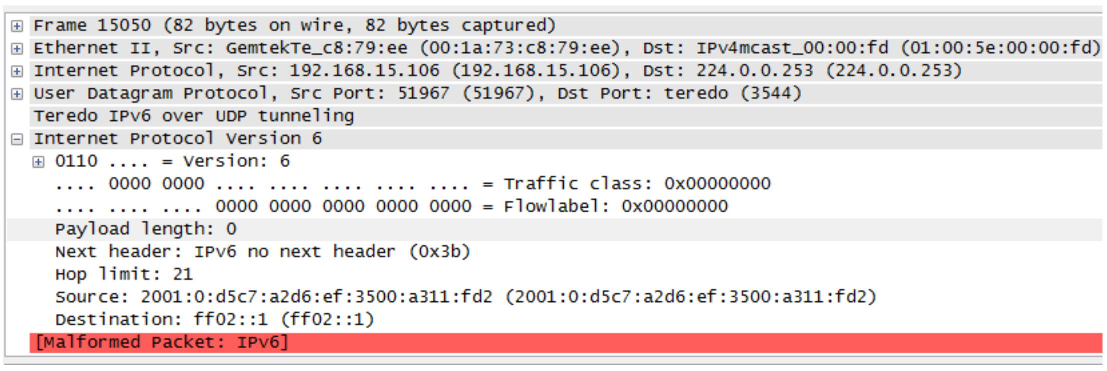
Figure 5-5: IP revolved to its country

Each error message is related to a particular system behavior. These behaviors are generally known and each time one of them is encountered, an investigation should be done as per why the message as occurred. In this case, the ICMP message indicates that the system is talking to another system that is not live or does not allow communication. After investigation, our local system has received a message from the foreign system, yet the foreign system is not accepting a message back. Looking at the issue further, the port used between our local system and the foreign system has been in used by other systems IPs during the attack.

Creating a snort rule that drops ICMP error messages at this level would help to limit the traffic load. However, important information about the state of the connection would be lost. Prior to dropping the packet, analysis of the state of the connection should be done.

- ***Scenario 5: Terodo IPv6 over UDP tunneling IPv4***

Some elements of the captured file reveals that the malicious user has attempted to hide traffic using Terodo IPv6 over UDP tunneling. Further analysis shows that protocol integrity was violated.



```
⊞ Frame 15050 (82 bytes on wire, 82 bytes captured)
⊞ Ethernet II, Src: GemtekTe_c8:79:ee (00:1a:73:c8:79:ee), Dst: IPv4mcast_00:00:fd (01:00:5e:00:00:fd)
⊞ Internet Protocol, Src: 192.168.15.106 (192.168.15.106), Dst: 224.0.0.253 (224.0.0.253)
⊞ User Datagram Protocol, Src Port: 51967 (51967), Dst Port: teredo (3544)
  Teredo IPv6 over UDP tunneling
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 0
    Next header: IPv6 no next header (0x3b)
    Hop limit: 21
    Source: 2001:0:d5c7:a2d6:ef:3500:a311:fd2 (2001:0:d5c7:a2d6:ef:3500:a311:fd2)
    Destination: ff02::1 (ff02::1)
  [Malformed Packet: IPv6]
```

Table 20: IANA reserved IP used for DDOS

Hostname	Country Name	Region	City	ISP	Organization
224.0.0.253	N/A	N/A	N/A		

Classic tunneling methods envisaged for IPv6 transition operate by sending IPv6 packets as payload of IPv4 packets [169];

Scenario 6: Malicious payload

Malicious users at times use a payload that could give a good indication of an attack.

```
9l);"E&mHxplX-R own you bitch!
```

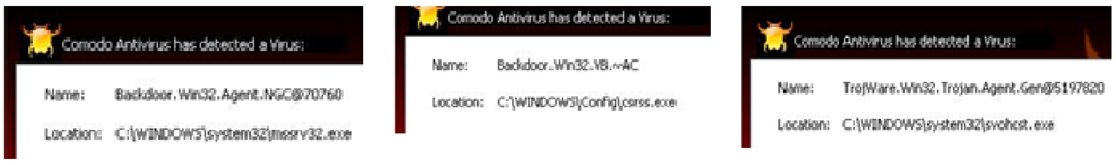
```
0000   00 18 39 dd 6c a2 00 0c 29 3b c9 22 08 00 45 00  ..9.l...);."..E.
0010   00 84 a5 2a 00 00 80 11 8e ff c0 a8 0f 6d 48 b1  ...*.........mH.
0020   ed 78 0c e0 0c 02 00 70 02 61 ff ff ff ff 58 2d  .x.....p.a....X-
0030   52 20 6f 77 6e 20 79 6f 75 20 62 69 74 63 68 21  R own you bitch!
0040   00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  ................
0050   01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  ................
0060   01 01 01 01 01 01 01 01 01 00 01 01 01 01 01 01  ................
0070   01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  ................
0080   01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  ................
0090   01 01                                            ..
```

Retrieving the payload from the above packet will give "….X-R own you bitch!”.

In this attack, the malicious user was changing the source port number on every single connection. A snort rule which can then be writen to stop this attack is the following:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any: (msg:"ET TROJAN Backdoor.Win32.VB.brg C&C DDoS Outbound"; flow:established,from_server; dsize:>100; content:"|ff ff ff ff|"; depth:12; content:" own you bitch!"; within:25; content:"|01 01 01 01 01 01 01 01 01 01 01 01 01|"; classtype:trojan-activity; threshold gen_id 1, sig_id 1853, type both, track by_dst, count 100, seconds 3;reference: VIRUS/TROJAN_Backdoor.Win32.VB; sid:; rev:1;)
```

Further analysis led to investigate what program was sending these packets. Using netstat, the command *netstat -aob -p UDP* has help to identify the program responsible of the damages. In this instance the attacking executable were csrss.exe, mssrv32.exe and svohcst.exe. A scan of the system by Comodo Antivirus has confirmed the same problem Figure 5-6: Virus Captured.



Comodo Antivirus has detected a Virus:
Name: Backdoor.Win32.Agent.NGC@70760
Location: C:\WINDOWS\system32\mssrv32.exe

Comodo Antivirus has detected a Virus:
Name: Backdoor.Win32.VB.~AC
Location: C:\WINDOWS\Config\csrss.exe

Comodo Antivirus has detected a Virus:
Name: TrojWare.Win32.Trojan.Agent.Gen@5197820
Location: C:\WINDOWS\system32\svohcst.exe

Figure 5-6: Virus Captured

Statistics and traffic pattern

Looking at traffic patterns, once could easily note that there is an excess of packets when compare to the normal routine. Under normal circumstances (no attacks) an average of 3.902 packets per second was recorded.

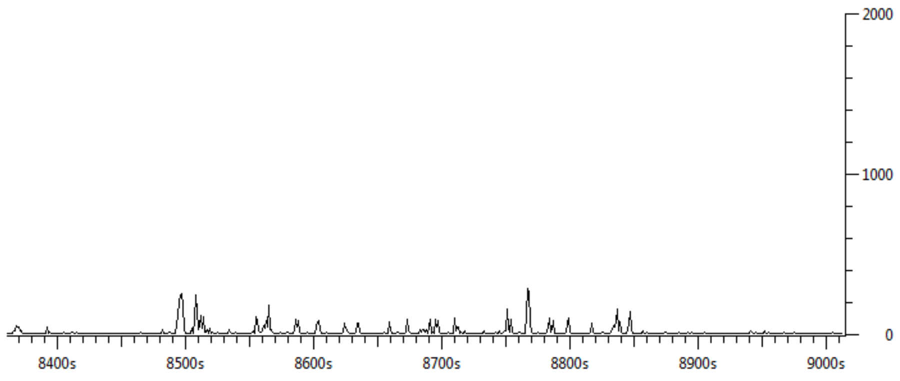


Figure 5-7: *normal traffic pattern- traffic not under attack*

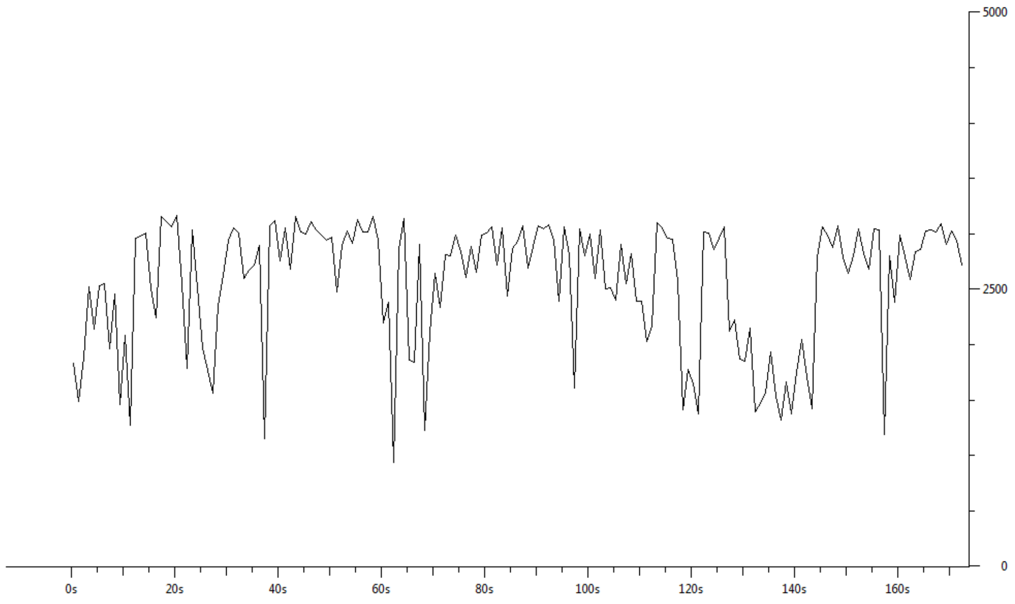


Figure 5-8: packet per second under medium UDP DDOS attack

Under attacks, a medium DDOS attack, 2554.251packets per second on average was going across the network *Figure 5-7*.

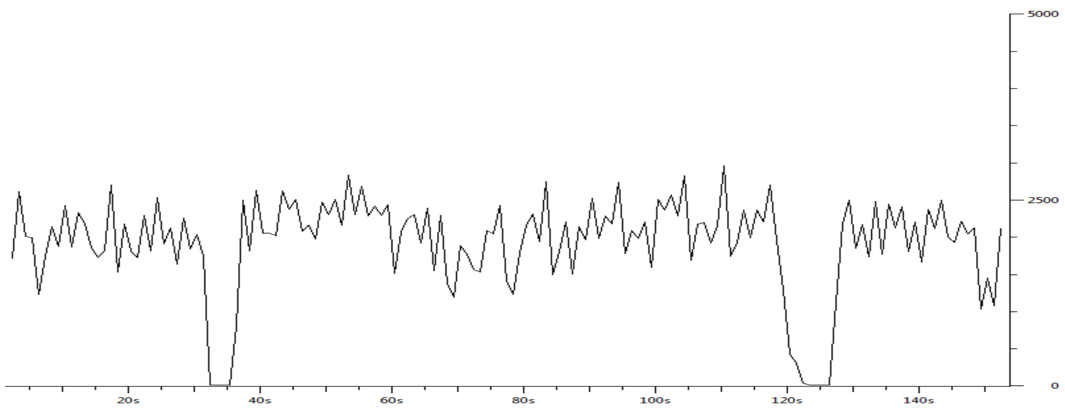


Figure 5-9: packet per second HTTP DDOS attack

Based on the traffic pattern, one could easily identify a misused or abuse of the system. However, systems are not static and momentarily may have heavy load due to a VOIP application or a video over the internet. Monitoring traffic requires a lot of time and patience as the normal behavior of a system has to be built over certain period of time. Once this normal behavior has been built, a range a threshold values can then be set to alarm in case of any drastic change.

5.3 Summary of DDOS attacks

There are certainly known ways of launching a DDOS attack. However, the techniques used by malicious users go beyond any classification. Based on the knowledge of publicly available DDOS attacks recorded, an attempt to classify DDOS attack was made.

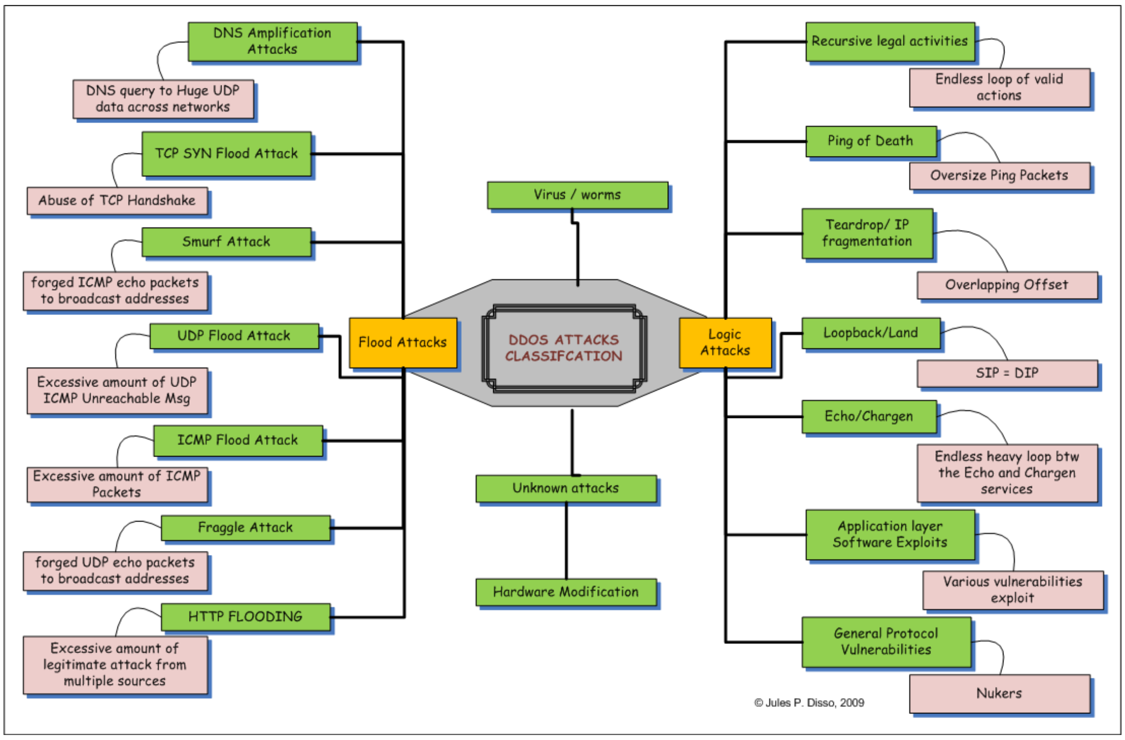

DDOS ATTACKS CLASSIFCATION
Flood Attacks
DNS Amplification Attacks
DNS query to Huge UDP data across networks
TCP SYN Flood Attack
Abuse of TCP Handshake
Smurf Attack
forged ICMP echo packets to broadcast addresses
UDP Flood Attack
Excessive amount of UDP ICMP Unreachable Msg
ICMP Flood Attack
Excessive amount of ICMP Packets
Fraggle Attack
forged UDP echo packets to broadcast addresses
HTTP FLOODING
Excessive amount of legitimate attack from multiple sources
Virus / worms
Unknown attacks
Hardware Modification
Logic Attacks
Recursive legal activities
Endless loop of valid actions
Ping of Death
Oversize Ping Packets
Teardrop/ IP fragmentation
Overlapping Offset
Loopback/Land
SIP = DIP
Echo/Chargen
Endless heavy loop btw the Echo and Chargen services
Application layer Software Exploits
Various vulnerabilities exploit
General Protocol Vulnerabilities
Nukers
© Jules P. Disso, 2009

Figure 5-10: DDOS attack classification

There are many types of DDOS attacks. Sometimes experts in the field will refer to an attack by the name used to perform the attack such as the stacheldraht [170].

In this classification, flooding attacks are considered as one type of attack in opposition to logic attack. In the flooding attack, there is no specific need to identify a vulnerability of the system. As long as a port number is open, a malicious user can flood that particular port [171-176]. As long as a webserver is running, a malicious user can request a large number of open connections, in the hope of making the server very busy until it crashes. More recently, when performing DDOS attacks, malicious users will request a page or series of pages repeatedly from various sources (compromised IP) [177], [178]. The requests whilst being syntactically correct, have a malicious intent. Due to their nature, it has been very difficult to distinguish between the requests made with good intention and those that are not.

Another class of DDOS attack is identified by the different attacks resultant from a certain violation of Protocol behaviour. In this category, the protocol definition is not violated but rather it is abused.

Another interesting group of DDOS attacks are attacks based on virus. The behaviour of these attacks is not very predictable. However, when the virus is identified, the problem can be rather easy to solve.

A more serious type of DDOS attack occurs when the hardware that has been sold was modified leaving a bug that will then be used for attack or by updating the firmware of the attack [179], [180]. This category or class of attack is very difficult to identify. However, when such a problem is identified, it can be easily fixed by replacing the faulty hardware. The work performed in this research will not address that type of attack as it is out of the author's competences. To the best of the author's knowledge there is no framework available or solution to detect such attack.

5.4 Solution Architect

This section will discuss the design of the new architecture

5.4.1 DDOS features requirements

The number of features to be considered for an optimal DDOS protection is important (Figure 5-11). These features have been organized into five main categories:

- Static list: these are the lists that exist in the community but are not currently fully considered for an IDS
- Dynamic elements: algorithmic based and flow management
- Analyzer: classic protocol analyser & flow management
- Mixed: features that have both static and dynamic elements
- Signature: classic Snort rules

For the purpose of this diagram, the static list and Signature will be represented with the same colour.

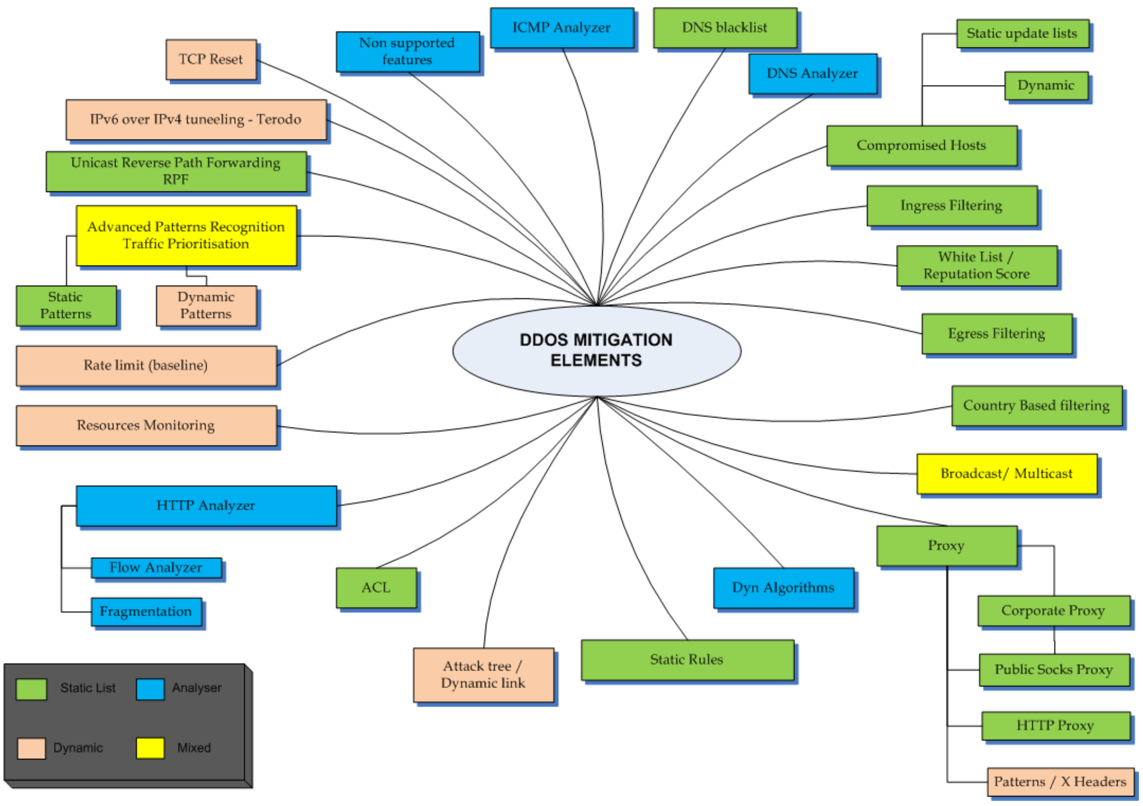


Figure 5-11: DDOS Protection Elements

5.4.2 IDS States

During the course of its action, an IDS should be able to switch between simple operation mode i.e. when there is no attack detected to attack mode where an attack is detected, and to mitigation mode when trying to get rid of the attack. As well as detecting, IDS should be able to react to attack, hence switch to IPS.

Three states are identified in this architecture Figure 5-12

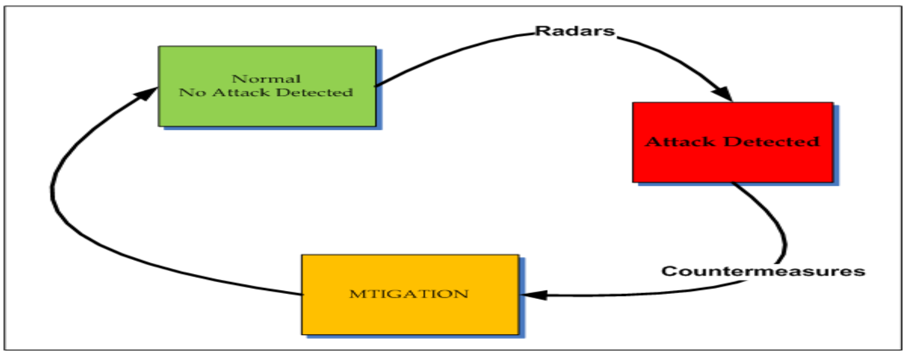


Figure 5-12: IDS States

5.4.3 Normal state

During the normal state, the IDS runs without knowledge of DDOS attack. In this state, the IDS can be subject to unlimited attack not classed as DDOS. The core security engine will be dealing with those attacks. However, a number of “radars” are activated to identify any potential DDOS activity. A radar is a behavioural monitoring agent. The radar would generally sits on the protected system sending regular updates to the IDS. The link between the radar and the IDS would be protected by a layer of TLS to avoid any malicious user tampering with the data being sent across. Radars that have been identified are known as:

5.4.3.1 Radar on Destination IP Address: R_DIP

The R_DIP is a radar that monitors the number of incoming request to the server over a rolling period predetermined by configuration. Over a rolling period of time Tx, if the number of IP participant go above the 80% percentile, a trigger will be sent to the management station (IDS) to change the state. This is considered as a hotspot.

5.4.3.2 Radar on Destination Port number: R_DPORT

The R_DPORT is a radar that monitors the number of incoming connections and the amount of data sent through that port number. If over a rolling period Ty configurable in the settings of the IDS is reached, a trigger will be sent to the management console to request a change of state. This is also considered as a hotspot.

5.4.3.3 Radar on resource monitoring

This radar monitors the overall performance of the protected system. On a regular basic, over a period of time Tz, the radar will send the level of resources available on the protected PC. A configurable range of resource level will indicate various actions that the IDS will take, and change the state accordingly.

5.4.4 Server response time

At the management console, a radar querying the Server (the protected system) would be installed. This will monitor the response time of the server and instruct the IDS to take the appropriate action.

5.4.5 Attack state

When radars have sent triggers to the management console signalling the presence of an attack, the IDS will enter the attack stage. Different levels of attacks are set by the severity of the attack.

5.4.6 Mitigation state

The IDS will come into the mitigation state when an attack is detected, and when that attack is rate critical. This is similar to a survival mode

5.5 Countermeasures

Mitigating DDOS attack is a very complicated task due to the nature of the attack itself. The model designed in this chapter is a multi-layered mitigation approach with three states Figure 5-13.

In the normal state, the number of features running as security measure is fairly limited to Ingress traffic, ACLs, protocols analysers which include basic threshold, and the resource monitoring agent. When an attack is detected and the status is changed to “attack mode”, many other features are activated: compromised host, Socks proxies, HTTP Proxies, non-supported protocols rules, and advanced patterns recognition.. If the attack persists, more security features are turn on. These are: country based filtering, corporate proxies, white lists and scoring algorithms.

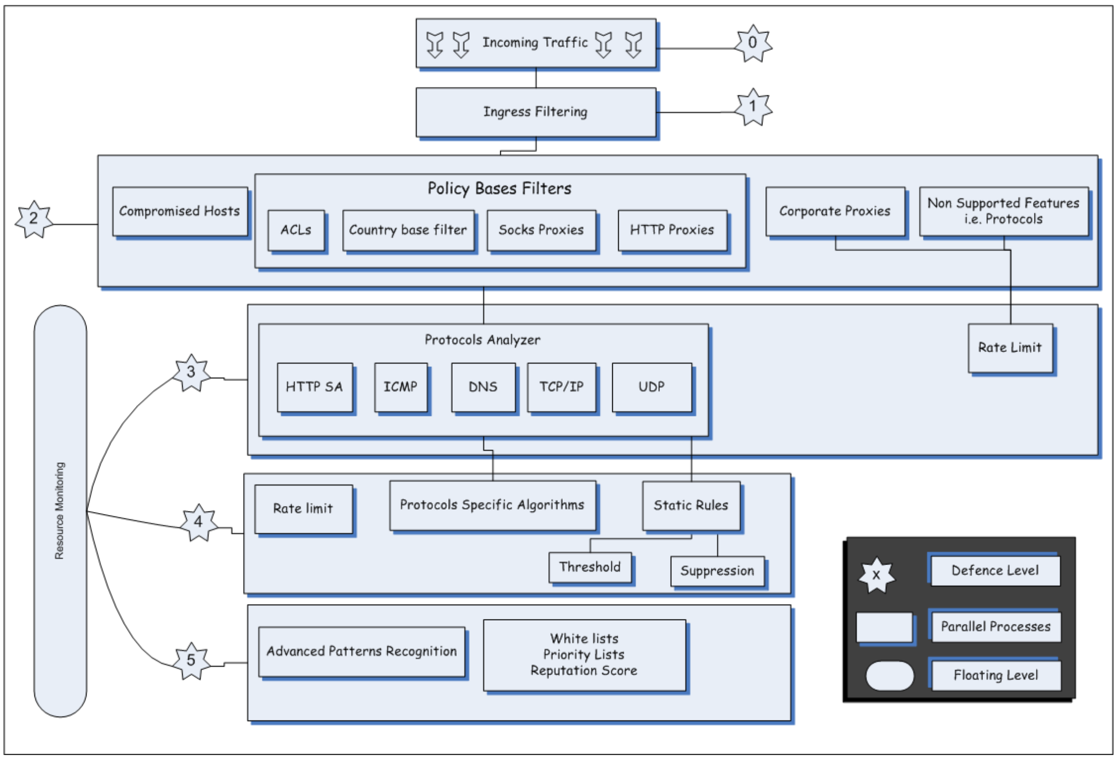
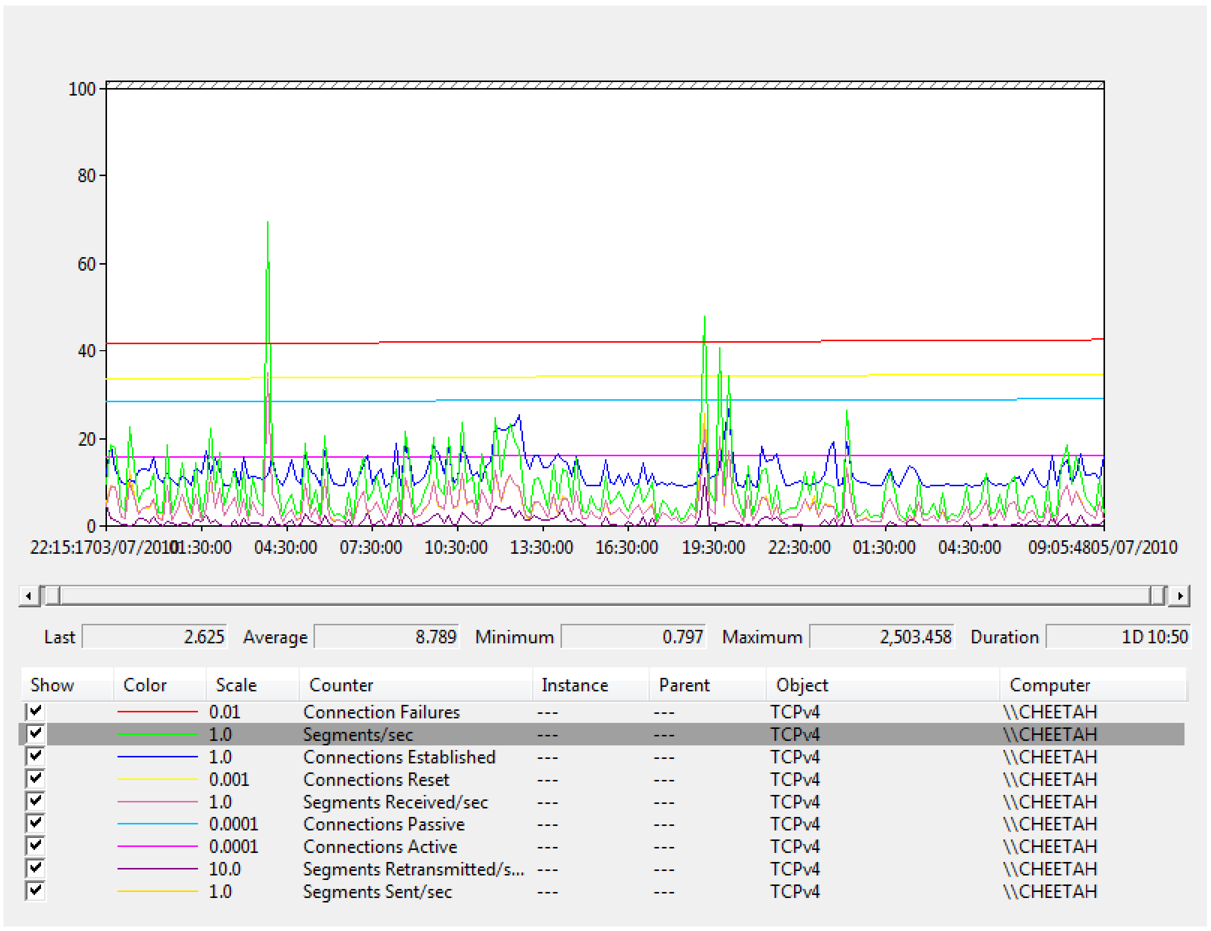


Figure 5-13: the DDOS architecture

5.6 DDOS attack detectors: RADAR

In this section, a number of studies are performed in order to determine attack indicators.

Based on the services provided and the number of participants, servers have different loads. The level of detection of attack for each server would be different depending on its normal activity curve. It is therefore important that the server keeps a baseline of the services provided. In this series of experiments, the baseline would be set and the appropriate security measure to detect attack will be produced. Data analysed here were collected on a commercial server from Vision Intel Ltd



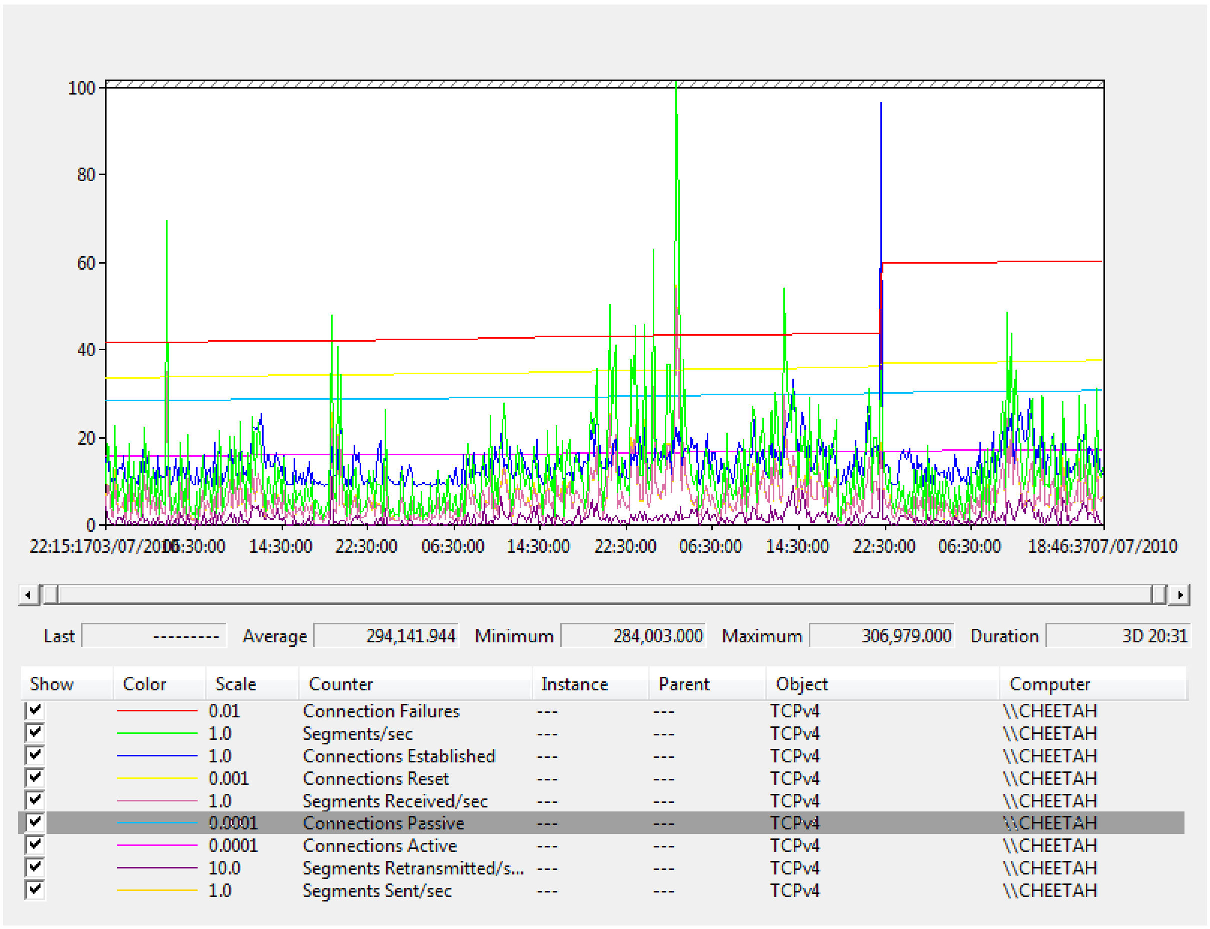
Last 2.625 Average 8.789 Minimum 0.797 Maximum 2,503.458 Duration 1D 10:50

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	0.01	Connection Failures	---	---	TCPv4	\\CHEETAH
✓	🟩	1.0	Segments/sec	---	---	TCPv4	\\CHEETAH
✓	🟦	1.0	Connections Established	---	---	TCPv4	\\CHEETAH
✓	🟨	0.001	Connections Reset	---	---	TCPv4	\\CHEETAH
✓	🟪	1.0	Segments Received/sec	---	---	TCPv4	\\CHEETAH
✓	🟦	0.0001	Connections Passive	---	---	TCPv4	\\CHEETAH
✓	🟪	0.0001	Connections Active	---	---	TCPv4	\\CHEETAH
✓	🟪	10.0	Segments Retransmitted/s...	---	---	TCPv4	\\CHEETAH
✓	🟨	1.0	Segments Sent/sec	---	---	TCPv4	\\CHEETAH

Figure 5-14: Server Cheetah - normal activity stream - HTTP performances

An analysis of Figure 5-14 reveals the number of connections failures during the capture. However, the reasons why there are so many connection failures are unknown since the author did not have access to the log files or any traffic capture to look into the problem. The trace file represents 34hours of web activity. The number of connections established was relatively low.

Similar analyses were repeated many times and during normal activity, the graphs of activities are very similar. Given that the patterns across the different captures are very similar, the rest of the analysis will be based on the capture that lasted three days. The latter file will then be compare to another capture where an experimental server was attacked.



Last --------- Average 294,141.944 Minimum 284,003.000 Maximum 306,979.000 Duration 3D 20:31

Show	Color	Scale	Counter	Instance	Parent	Object	Computer
✓	🟥	0.01	Connection Failures	---	---	TCPv4	\\CHEETAH
✓	🟩	1.0	Segments/sec	---	---	TCPv4	\\CHEETAH
✓	🟦	1.0	Connections Established	---	---	TCPv4	\\CHEETAH
✓	🟨	0.001	Connections Reset	---	---	TCPv4	\\CHEETAH
✓	🟫	1.0	Segments Received/sec	---	---	TCPv4	\\CHEETAH
✓	🟦	0.0001	Connections Passive	---	---	TCPv4	\\CHEETAH
✓	🟪	0.0001	Connections Active	---	---	TCPv4	\\CHEETAH
✓	🟪	10.0	Segments Retransmitted/s...	---	---	TCPv4	\\CHEETAH
✓	🟨	1.0	Segments Sent/sec	---	---	TCPv4	\\CHEETAH

Figure 5-15: HTTP observation 3 days activities

In Figure 5-15: HTTP observation 3 days activities the patterns observed in Figure 5-14 are very similar. However, during the second capture that lasted three days, the servers seemed busier as the average of established connection is higher than previously recorded. At the same time, the number of connection failures was pretty high. Again, not having access to the trace files, the reasons for the high number of connection failures remained unknown.

In this subsection, the two servers' behaviours are compared with regards to connections failures, segments and connections established.

Referring to Figure 5-16 and Figure 5-17 the ratio of connection failures is [0.01 / 0.000001] which workouts to be 1 to 10000. There is clearly a significant difference between the two behaviours.

Regarding the segments per seconds the ratio is 1 to 100 which again a significant gap.

As to the connections established, the ratio is 1 to 10.

Very few reasons can justify this change of pattern when there is no attack. In the recent events, the number of hit Google received significantly increased when Michael Jackson passed away. However, this behaviour can be expected from big companies. From small to medium companies, such a change in behaviour would indicate anomalies. The question now that arise is how to determine when to raise the alarm that an attack has started.

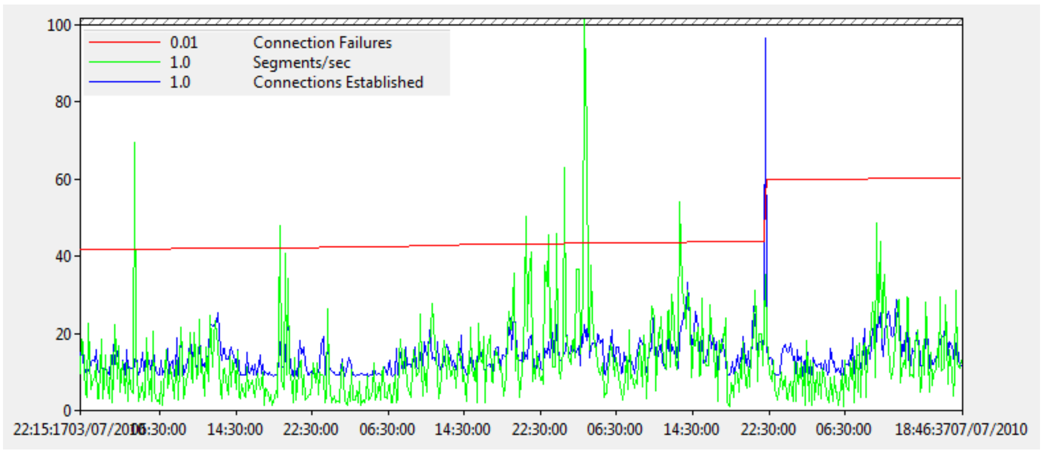


Figure 5-16: Cheetah Server behaviour 1

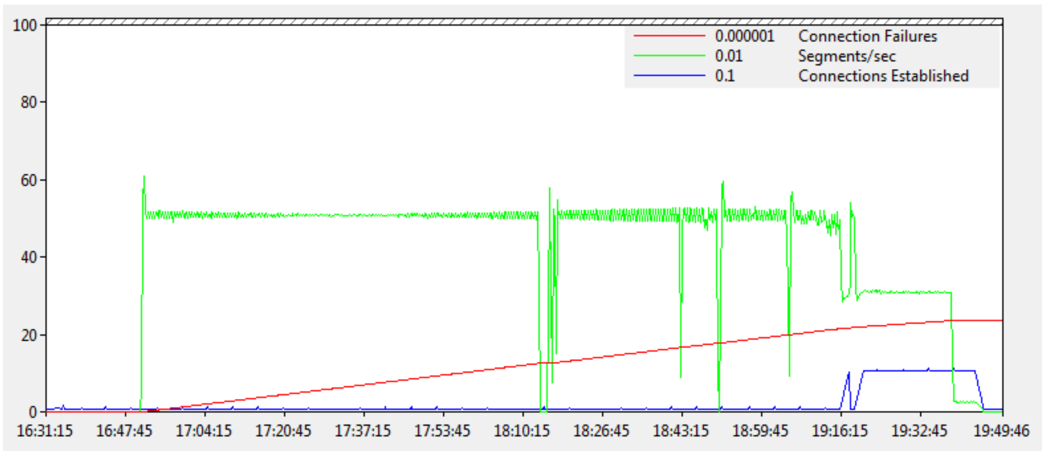


Figure 5-17: Lynx server behaviour 1

In order to find a point in time when an alert should be raised, this research looked at the percentile. However the author changed the default behaviour of the percentile. A rolling period was set over which the percentile would be computed. For the first stage of attack, if the average of the radar over the rolling period was more than 50% of the X^n percentile over n cycle an alert would be raised as the first level of attack.

If the mean values of the radar was more than 70% or more of the X^n percentile, the alert would be raised as attack level 2.

5.7 Conclusion:

In this chapter, the DDOS mitigation and detection framework is presented. One of the unique features of the framework presented here is the multi-level detection capabilities. Three levels were defined under which the framework will have different behaviour. In addition, radars were introduced: attack detector that alert in case of any system performance degradation. All the features included in the DDOS framework could work as separate units and offer their level of protection. This framework will be later integrated into the multistate Intrusion detection and prevention. When the system protected is under severe attack (DDOS), the DDOS framework would be the priority of the MIDaPS framework.

6 Multistage Intrusion Detection and Prevention System: MIDaPS

The work presented in this chapter is the result of extensive experiments based on the problem found either by literature reviews or by personal experiments.

In this chapter, the comprehensive list of features of the MIDaPS is presented here. The author identifies four levels of visibility of attacks around which the new IDS will be built. The work presented in this chapter is a form of summary of all the work that has been achieved in earlier chapters. The author goes on presenting a new yet audacious Intrusion Detection and Protection architecture that is built around the fact that most recent attacks are vectors and multistage attacks that generally lead to a DDOS attack. The architecture presented here is based on multistage attack detection scenarios as well as DDOS mitigation and detection technique. In addition, all the functionalities have been designed to be fully compatible with a multicore environment. The author stresses that multicore is not the main focus in this chapter or of this thesis.

After defining the V-BANI framework, the chapter will compare the Snort features to MIDaPS features. MIDaPs is designed as modular IDS. The rest of the chapter will discuss the default modules that form the base of the IDS and the reasons why these modules are important.

6.1 The V-BANI Framework

Intensive research has been carried so far to understand and model attacks in order to build a solid detection and mitigation system. There are 4 different categories that emerged from the different analyses done earlier when looking at vector and multistage attacks and the kind of protection needed against these attacks.

The first category of attacks is composed of attacks that are visible to network security systems. These attacks can be detected and stopped by a well configure detection and mitigation system. This could be a violation of protocol definition, a protocol abuse, a known pattern used by malicious users in order to disrupt, change, or stop any legitimate activity

The second category is composed of attacks or at least part of attacks that are generally considered to be legitimate actions and therefore not a subject of concern for security systems. For instance, a computer could be sending information out to another computer. This is completely legitimate and it is the basis of any communication. However, sending data from one computer to another would stop been legitimate if the previous action was a brute force attack on root passwords.

The third category is the type of attack that affects a system without any physical contact to the system. For instance the command will return username and password of website that use Frontpage extension. Even though the password is encrypted, it can be unencrypted by “john the ripper” in few minutes.

```
"# -FrontPage-" filetype:pwd inurl:(service | authors | administrators | users)
```

This is an excellent starting point for a malicious user. The victim system in this case is not aware of the details that are made public and has no knowledge of someone accessing them.

In the fourth category attacks are partially performed inside the local system. Some rootkit or malware require rebooting after installation. Rebooting the system can be visible within the local host and not on a network level. Also, malwares generally perform modifications of system files. In order to have the full picture of the attack, it is important to understand the changes that are made on local systems. Depending on where the IDS is installed, there might not be any direct communication within the systems protected and the IDS. A good security system should consider investigating the critical changes on the protected systems.

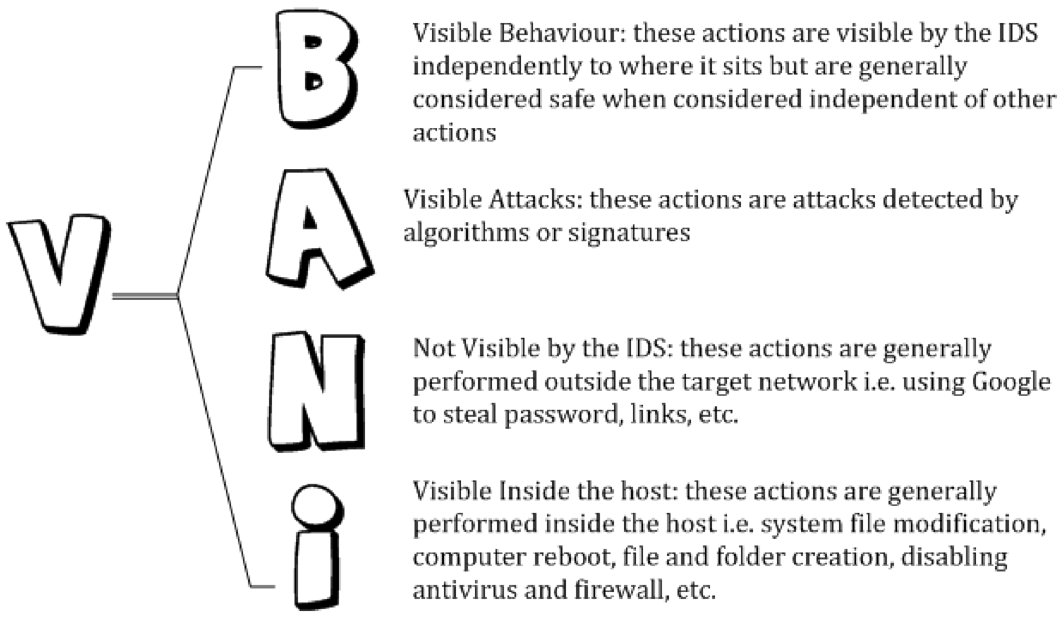

V
B
Visible Behaviour: these actions are visible by the IDS independently to where it sits but are generally considered safe when considered independent of other actions
A
Visible Attacks: these actions are attacks detected by algorithms or signatures
N
Not Visible by the IDS: these actions are generally performed outside the target network i.e. using Google to steal password, links, etc.
i
Visible Inside the host: these actions are generally performed inside the host i.e. system file modification, computer reboot, file and folder creation, disabling antivirus and firewall, etc.

Figure 6-1: V-BANI framework

The V-BANI framework [Figure 6-1: V-BANI framework] takes its name from the fact that four levels or categories of visibility of attacks. Current IDS or systems tend to address one or two at most at a time yet the level of sophistication of attacks is such that, if any level of visibility of attack is ignored, attack will go undetected.

Based on the V-BANI framework a number of features that need to be considered when designing IDS have been identified and compared to Snort IDS.

6.2 Comparing MIDaPS features to Snort features

Features	MIDaPS	Snort	Comment
Ingress traffic • Filter incoming traffic based on source and destination IP • Filter incoming traffic based on source and destination ports	✓ ✓	✓ ✓	
Outgress traffic • Filter outgoing traffic based on source and destination IP • Filter outgoing traffic based on source and destination ports	✓ ✓	✓ ✓	
Limit simultaneous communication • Block many connections matching a criteria	✓		
Logging • Log traffic on request based on matching condition	✓	✓	Existing in Snort but not fully functional
Grouping and naming of IPS • Ability to group IPs by network	✓		
Grouping by ports number • Ability to group traffic by services	✓		Snort can block traffic based on port but have no knowledge on the type of services
Operating System identification • Ability to filter traffic based on OS	✓		
Layer 2 filtering • Ability to bridge interface and filter traffic between them	✓		
Packet normalisation • Normalising packet for protocol analysis	✓	✓	

Deep packet inspection • Ability to look into packet content	✓	✓	
Protocol Analyser • Comprehensive interpretation of protocol definition • Comprehensive interpretation of limitation	✓ ✓	✓	
Full state management • Flow manager • Limit states per host • Limit concurrent connection per unit of time • Limit concurrent response per unit of time • Synproxy state management	✓ ✓ ✓ ✓ ✓		
X-header analysis	✓	✓	Can be implemented in Snort as a signature
Slow path • In-depth packet analysis	✓		This will be an in-depth analysis
Compromised IP list • Known list of IPs compromised	✓	✓	Exist in Snort as rule
RBN Russian Bot Network	✓	✓	Exist in Snort as rule
DNS Blacklist			
Privileged List • Known IPs that access specific services (admin access)	✓		Very limited in Snort but can be implement with some rules
MD5 rogue list • Known list of server using vulnerable SSL	✓		
Attack classification Categorise attack by group of severity Categorise attack by level of violation	✓ ✓ ✓	✓ ✓ ✓	
Packet fragmentation handling	✓	✓	
Baselines	✓	✓	Exist as threshold
Packet obfuscation	✓	✓	Very limited in Snort
Stream segmentation	✓		
RPC Fragmentation handling	✓		
URL Obfuscation	✓	✓	
Remote sensor/Agent	✓		
HTML Obfuscation	✓		
URL filtering	✓		Can be implemented in Snort as rules
Stateless capability	✓	✓	

Raw packet processing	✓	✓	
HTTP Load management	✓		
Slow path – packet decoding	✓		
Modular – Plugins	✓	✓	
TCP Reset	✓		

Table 21: Features comparison between Snort and MIDaPS

There are clear differences between Snort and MIDaPS. There are many more features in MIDaPS than they are in Snort. However, our experiments have shown that the Snort did not handle perform well under high speed. This under performance was due to the fact that Snort was performing unnecessary work. This actually suggests that the number of tasks to perform when analysing the traffic should be kept to minimum. Even though MIDaPS was built with many more features that what Snort has, all the features are not to be used at the same time. Also, the filtering mechanism introduced in MIDaPS is such that the amount of traffic decrease as it goes down the chain of the IDS functionalities. For instance, module1 is responsible of eliminating any unnecessary traffic that comes into the system. The ingress traffic is the first level of filtering. In this module, the traffic can be limited to a certain range of IPs. Also, depending on policies, any unknown proxy traffic can be filtered. If any reserved IP is used, it will be filtered at this level. Snort does not provide these facilities.

Module 2 is another module that aims at reducing to workload of the IDS. By using a multilayer of classification (by port, destination IP, service, flow) the traffic is organised In such a way that packets related to the same flow are directed toward the same core. However, the redirection of traffic is managed by a load balancer which ensures that each core within the architecture receives the same amount of work. This module takes into account the load variation that may be cause either by the variability in the load or by the computational power of the various nodes, or by the computation required for each task (i.e. we use an iterative scheme to compute $spp(x)$and the number of iterations depends on the difficulties encountered to solve the problem).

Module 5 is another module that introduced traffic filtering. This module can significantly reduce the traffic load by adding a protection over a number of IP that do not need to be checked (corporate proxies) or by applying limits to a range of IP address.

In module 6, when URL filtering is active, the traffic load will be great reduced. By filtering by URL, the packets will be blocked and prevented from going any further into the system. Depending on the options that need filtering, they could be thousands or even millions of URL that could be affected. A company may choose not to allow fashion website, gaming, sexual adult theme website, social networking, and many more.

We argue that the number of features is not always an inconvenient when the features are used appropriately. In their study, [181] argues that the number of features will reduce the performance of the IDS and went on experimenting the behaviour of the IDS with reduced features. From the empirical results they obtain, it is seen that by using the hybrid model Normal, Probe and DOS could be detected with 100% accuracy and U2R and R2L with 84% and 99.47% accuracies, respectively. This shows that reducing the number of features is not always an advantage.

6.2.1 Remote sensor/Agent

Depending on the settings of the network, the intrusion detection system will be located in or at the border of the system being monitored. Remote sensors are responsible for checking the changes that occurs on the system. Once changes on system files or system parameters have occur, an alert will be sent to SYSLOG who will then send another alert to MIDaPS as shown in Figure 6-2. Both system monitoring and syslog are on the remote system.

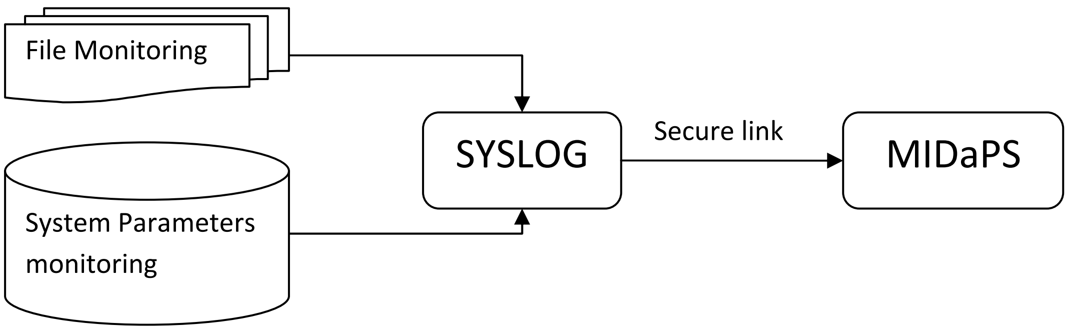


Figure 6-2: remote agent architecture

6.3 The architecture

The Multistage Detection & Prevention System is built around seven core components and three functional modes: Normal operation mode, Attack mode and Mitigation mode.

MIDaPS is design (Figure 6-3) to change its operation mode depending on the attack level. The overall architecture is presented in Figure 6-3

MIDaPS is organised into modules, each module carrying a set of function. However, the different functionalities of each module will be available depending on the level of attack. The general description of modules is as follow:

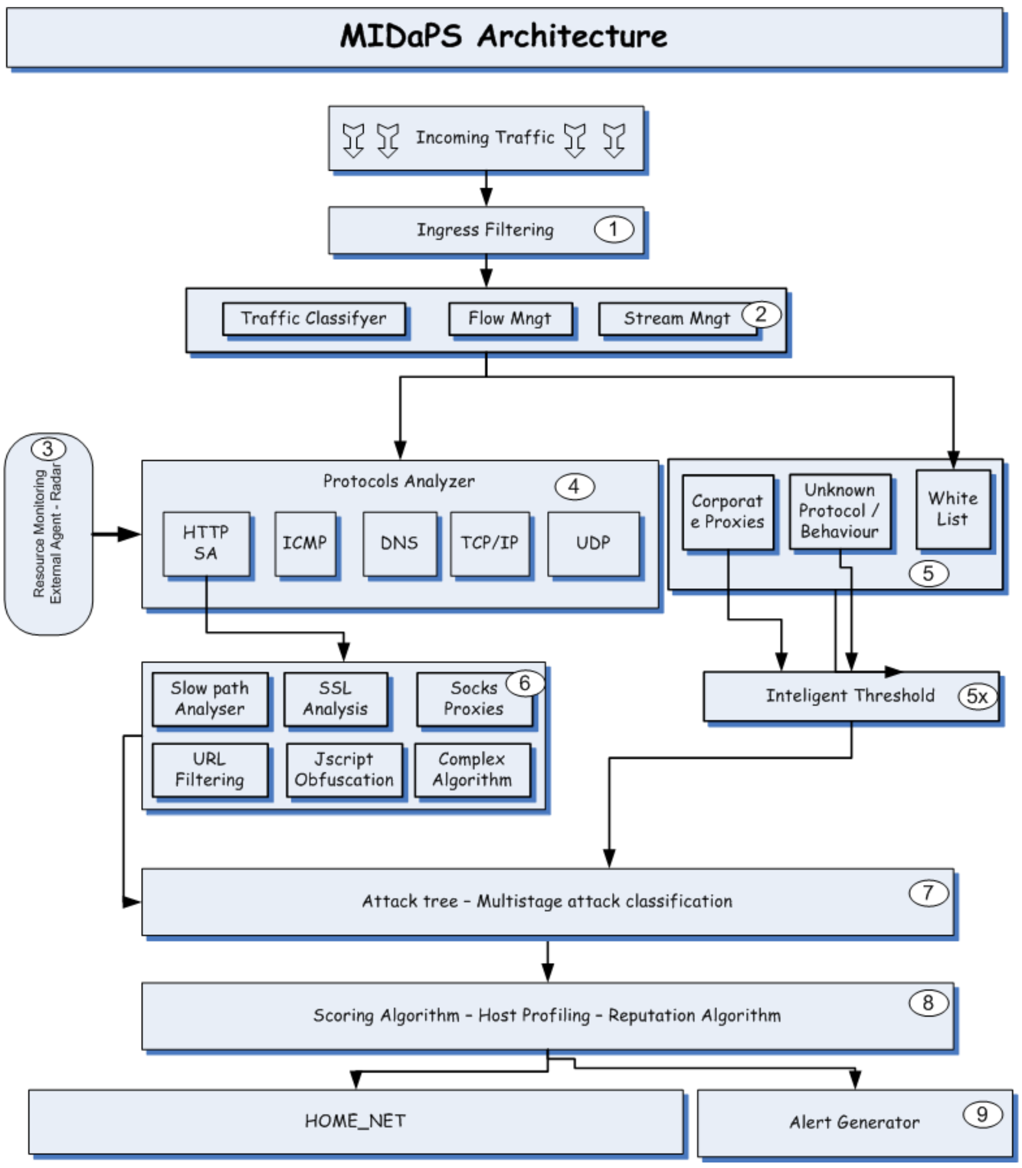


Figure 6-3: MIDaPS architecture

6.3.1 Module 1: Ingress traffic filter

This module is responsible for determining what traffic is accepted into the network. The Internet Assigned Numbers Authority (IANA) made public the list of IP addresses that are not in used. When spoofing IPs, attackers generally use this IPs to generate illegal traffic, causing the attack system to generate many ICMP messages [182]. In addition, the range of local IPs that is allowed to access the network is specified in this module. Depending on its position in the network, the IDS will deal with both local IPs and External IPs. The checks are performed on the IP header only. This eliminates the need of any other check and reduces the load of all subsequent modules. [183] describe ingress filtering as one of the most effective way of protecting against spoofing IP addresses

This feature is not is not implemented in Snort. However, it is possible to write rules that will perform the ingress filtering. The drawback of using rules for this purpose is that the spoofed IP will go through all the checks right up to the security engines before stopping any traffic originating from a spoof IP. The method of implementing ingress traffic in MIDaPS is therefore more effective that using Snort rules.

The algorithm for filtering ingress traffic would be:

```
var
IANA_RESEVED_IP = {list of IP range reserved by the IANA}
HOME_NET = {List of IPs currently in used in the network}

ingress_traffic(SIP)
 {
         // SIP = Source IP
   if (SIP exist in IANA_RESEVED_IP) AND (not In HOME_NET)
                  terminateflow() // this function will terminate (kill) all traffic related to the
SIP
   else
          Proceed with packet
 }
```

6.3.2 Module 2: traffic classifier and its associate elements.

Module 2 is composed of 3 core elements: The traffic classifier, the flow manager and the stream manager.

6.3.2.1 The traffic classifier:

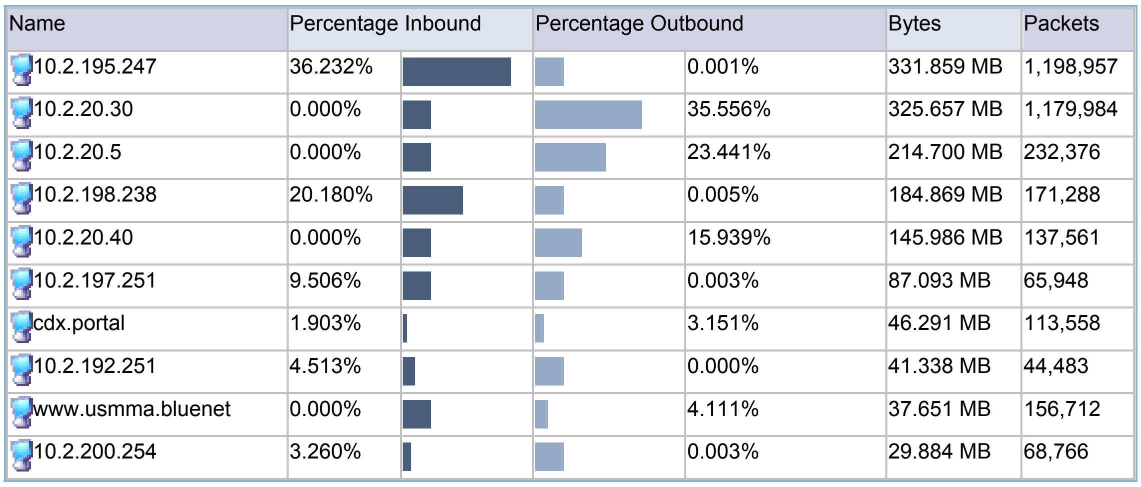
Network traffic is heterogeneous and can be categorised in many ways depending on the objectives. The IDS proposed in this research uses three modes of operations: Normal, Attack, and Mitigation. The Normal mode is the operations in which the IDS check for irregularities in the traffic as well as checking any indication of serious attack. In this research, DDOS is considered to be a serious attack. All other attacks will be dealt with during the normal operation mode. DDOS attacks can be performed a many ways. For instance, attackers can target one particular port or service (one or many ports numbers), or a protocol (P2P). In such scenarios, a traffic filter is part of the mitigation method. It is important to detect attacks before they become very severe. The objective of setting various statistics is to apply an algorithm that will detect any irregularity in the traffic pattern. Each network has its own unique pattern. In order for the IDS to recognise a significant change, the IDS should learn to recognise the traffic pattern: It's only then that anomalies could be detected. The intelligent threshold imposed on the traffic is not used as a defence tool but rather as a detection of possible attack.

Different traffic classification methods and their impact were studied:

- Payload
- Application
- Protocol
- Port number
- Statistical methods

Classification by payload

In regards to payload, traffic group by tuples (flow) i.e. source IP and destination IP, source IP + destination IP + source Port + destination Port, TCP options (SYN, PSH, ACK, RST, etc.) [184], [185]



Name	Percentage Inbound	Percentage Outbound	Bytes	Packets
10.2.195.247	36.232%	0.001%	331.859 MB	1,198,957
10.2.20.30	0.000%	35.556%	325.657 MB	1,179,984
10.2.20.5	0.000%	23.441%	214.700 MB	232,376
10.2.198.238	20.180%	0.005%	184.869 MB	171,288
10.2.20.40	0.000%	15.939%	145.986 MB	137,561
10.2.197.251	9.506%	0.003%	87.093 MB	65,948
cdx.portal	1.903%	3.151%	46.291 MB	113,558
10.2.192.251	4.513%	0.000%	41.338 MB	44,483
www.usmma.bluenet	0.000%	4.111%	37.651 MB	156,712
10.2.200.254	3.260%	0.003%	29.884 MB	68,766

Table 22: traffic classification - Top 10 IPs

Based the information provided by Table 22: traffic classification - Top 10 IPs an administrator is able to make the decision to turn off any communication to or from that host. An IDS should be able to provide live statistics on the network traffic. The objective of classifying traffic by payload enables the IDS to identify the level at which each IP is involved in the different communications. Based on intelligent threshold (dynamic threshold), the IDS will identify IP who have significantly changed their behaviour. A profile will be built for each IP. When there is a significant change, the IDS will flag that IP.

Classification by application:

Most applications can have a signature that can be used to identify the presence in the network. Traffic classification is often used in deep packet analysis [186]. Classification by application enables the administrator to have to power to decide to block a particular application. For instance, all P2P application traffic can be stopped if an administrator wishes to do so. This will reduce the flow of traffic coming into the network as well as reducing the amount of traffic subject to checks. An overview of patterns used to filter traffic is given in Table 23

Application Name	Patterns
Apple Juice - P2P filesharing	`^ajprot\x0d\x0a`
Jabber (XMPP) - open instant messenger protocol - RFC 3920	`<stream:stream[\x09-\x0d ][ -~]*[\x09-\x0d ]xmlns=['"]jabber`
GTalk, a Jabber (XMPP) client	^<stream:stream to="gmail\.com"
HTTP by Download Accelerator Plus	`User-Agent: DA [678]\.[0-9]`
VNC - Virtual Network Computing. Also known as RFB - Remote Frame Buffer	`^rfb 00[1-9]\.00[0-9]\x0a$`
SSH Secure shell	`^ssh-[12]\.[0-9]`

Table 23: Application patterns [187]

Classification by protocol

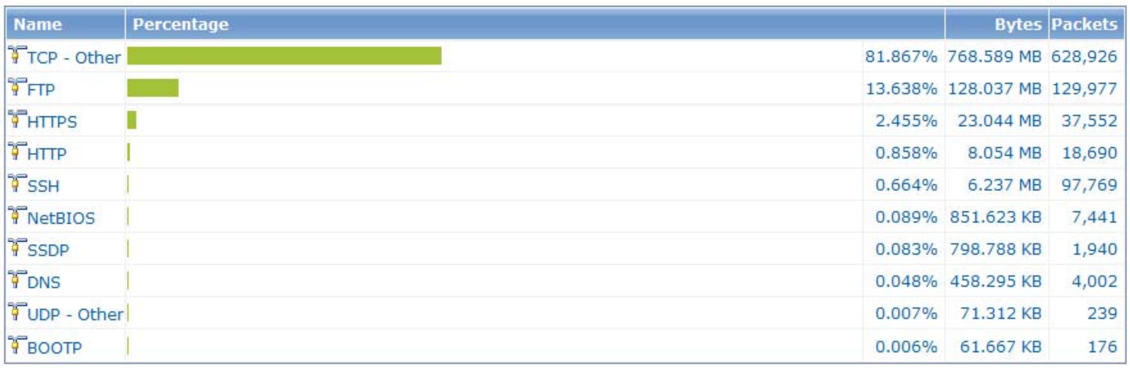
Each network would have fairly standard proportions of traffic. If the patterns generally observed change significantly, an alert should be raised. The most common protocols are:

Protocol	ID		Protocol	ID
ICMP	1		ESP	50
IGMP	2		AH	51
TCP	6		EIGRP	88
ICMP	1		OSPF	89
EGP	8		PIM	103
UDP	17		VRRP	112
IPv6	41		L2TP	115
RSVP	46		Other	0-255
GRE	47			

Table 24: common protocols

Classification by port number

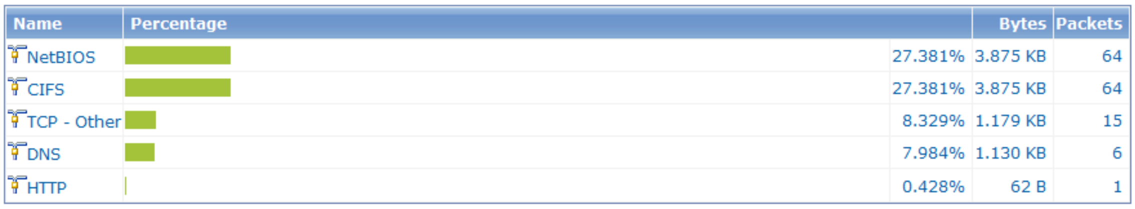
Classification by port number is fairly common. However, grouping traffic by port number alone is not efficient enough as there are no physical limitations on which port number an application can use. The classification used here will allow the IDS to track any irregular activities.



Name	Percentage		Bytes	Packets
TCP - Other		81.867%	768.589 MB	628,926
FTP		13.638%	128.037 MB	129,977
HTTPS		2.455%	23.044 MB	37,552
HTTP		0.858%	8.054 MB	18,690
SSH		0.664%	6.237 MB	97,769
NetBIOS		0.089%	851.623 KB	7,441
SSDP		0.083%	798.788 KB	1,940
DNS		0.048%	458.295 KB	4,002
UDP - Other		0.007%	71.312 KB	239
BOOTP		0.006%	61.667 KB	176

Figure 6-4: Top 10 application protocol based on [25]

An efficient way to detect a DDOS attack would be the change in regular patterns. For a given network, if TCP connections are over 80% of the traffic observed in the network and UDP less than 1%, an increase of UDP traffic over 10% or a decrease of TCP traffic to 50% would indicate a serious anomaly. This change of behaviour should be flagged and monitored. In general, any protocol that goes beyond its normal usage should be subject to inspection.



Name	Percentage		Bytes	Packets
NetBIOS		27.381%	3.875 KB	64
CIFS		27.381%	3.875 KB	64
TCP - Other		8.329%	1.179 KB	15
DNS		7.984%	1.130 KB	6
HTTP		0.428%	62 B	1

Figure 6-5: Top 10 application protocol based on a DDOS capture

Looking at Figure 6-4 and Figure 6-5 one could easily notice the big difference between the two traffic patterns. In the first case, the traffic can be considered fairly normal, but for the second case, DNS traffic is as high as TCP traffic which is a very rare pattern on normal behaviour. Port 139 [Netbios] is very popular amongst DDOS attacks. An increase of traffic on port 139 would indicate serious irregularities. Once an alert is fired, the IDS will change its mode of operation to a more defensive mode.

Classification by statistical method

Based on the analysis done earlier in this research, one would notice that different IPs had the same payload content and hence the same payload length; the frequency of packets sent was similar; the number of bytes downloaded was similar. In this section, any metric could be computed

The following graphs (Figure 6-6, Figure 6-8, and Figure 6-9) are based on the "2009 Inter-Service Academy Cyber Defence Competition" [25]

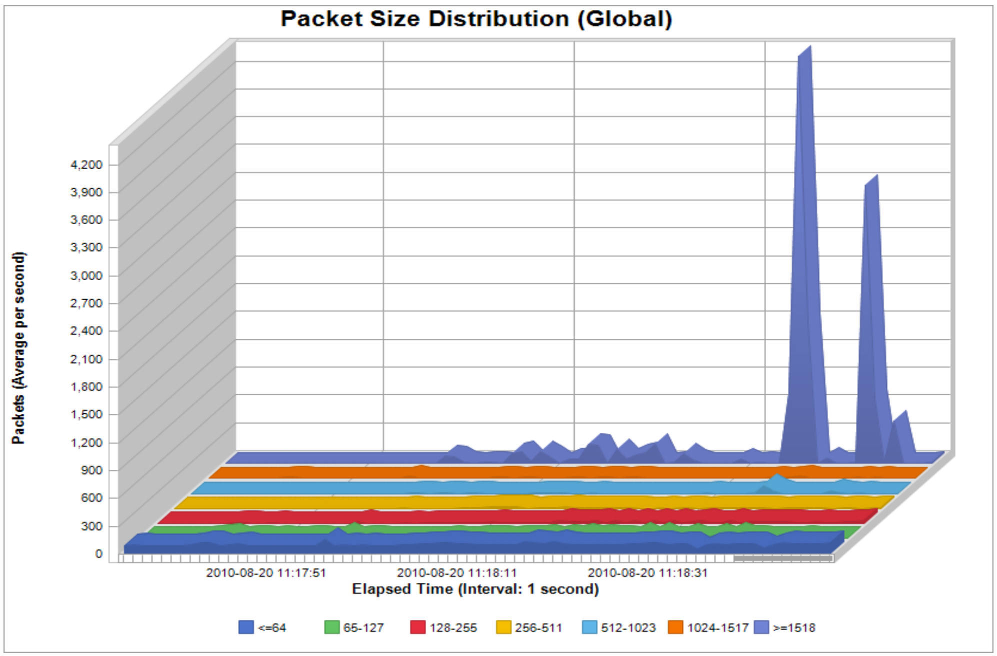
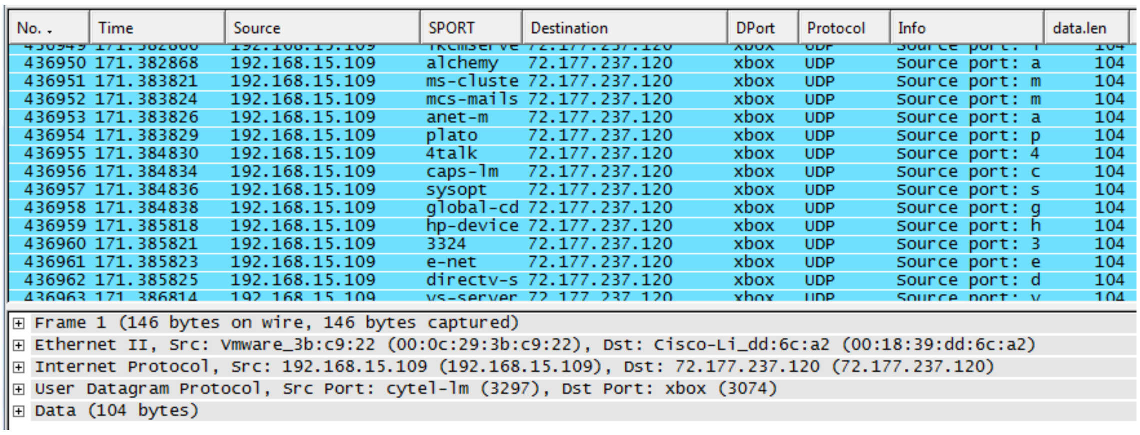


Figure 6-6: traffic classification - packet size distribution

In this scenario, most packet send are bigger or equal to 1518bytes. During an attack, if the problem is found to be the amount of data sent, Figure 6-6 gives enough information to make an informed decision. Blocking all traffic for which the packet Len >= 1518 will considerably reduce data flow. The packet distribution size is a good way to identify DDOS attacks. Most script kiddies do not use intelligent packets size distribution during the attack. Looking at Figure 6-7, it is fairly easy to notice that the attacker was using different source port against one destination port. More interestingly, the payload_Len (data size) did not change during the course of the attacks. Classifying the traffic by payload helps to identify such attack and stop them.



No.	Time	Source	SPORT	Destination	DPort	Protocol	Info	data.len
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	xbox	UDP	Source port: [illegible]	104
436950	171.382868	192.168.15.109	alchemy	72.177.237.120	xbox	UDP	Source port: a	104
436951	171.383821	192.168.15.109	ms-cluste	72.177.237.120	xbox	UDP	Source port: m	104
436952	171.383824	192.168.15.109	mcs-mails	72.177.237.120	xbox	UDP	Source port: m	104
436953	171.383826	192.168.15.109	anet-m	72.177.237.120	xbox	UDP	Source port: a	104
436954	171.383829	192.168.15.109	plato	72.177.237.120	xbox	UDP	Source port: p	104
436955	171.384830	192.168.15.109	4talk	72.177.237.120	xbox	UDP	Source port: 4	104
436956	171.384834	192.168.15.109	caps-lm	72.177.237.120	xbox	UDP	Source port: c	104
436957	171.384836	192.168.15.109	sysopt	72.177.237.120	xbox	UDP	Source port: s	104
436958	171.384838	192.168.15.109	global-cd	72.177.237.120	xbox	UDP	Source port: g	104
436959	171.385818	192.168.15.109	hp-device	72.177.237.120	xbox	UDP	Source port: h	104
436960	171.385821	192.168.15.109	3324	72.177.237.120	xbox	UDP	Source port: 3	104
436961	171.385823	192.168.15.109	e-net	72.177.237.120	xbox	UDP	Source port: e	104
436962	171.385825	192.168.15.109	directv-s	72.177.237.120	xbox	UDP	Source port: d	104
436963	171.386814	192.168.15.109	vs-server	72.177.237.120	xbox	UDP	Source port: v	104

```
⊞ Frame 1 (146 bytes on wire, 146 bytes captured)
⊞ Ethernet II, Src: Vmware_3b:c9:22 (00:0c:29:3b:c9:22), Dst: Cisco-Li_dd:6c:a2 (00:18:39:dd:6c:a2)
⊞ Internet Protocol, Src: 192.168.15.109 (192.168.15.109), Dst: 72.177.237.120 (72.177.237.120)
⊞ User Datagram Protocol, Src Port: cytel-lm (3297), Dst Port: xbox (3074)
⊞ Data (104 bytes)
```

Figure 6-7: DDOS patterns

Classifying the traffic using TCP flags are one improves the detection of SYN flood attacks. At any given time, the administrator will be able to see how many IP have opened a connection without activity. When too many connections are open, the resources are used and the system runs out of resources causing the system to crash. A baseline should be defined per system in order to ensure that the threshold set reflects the environment in which the IDS is installed.

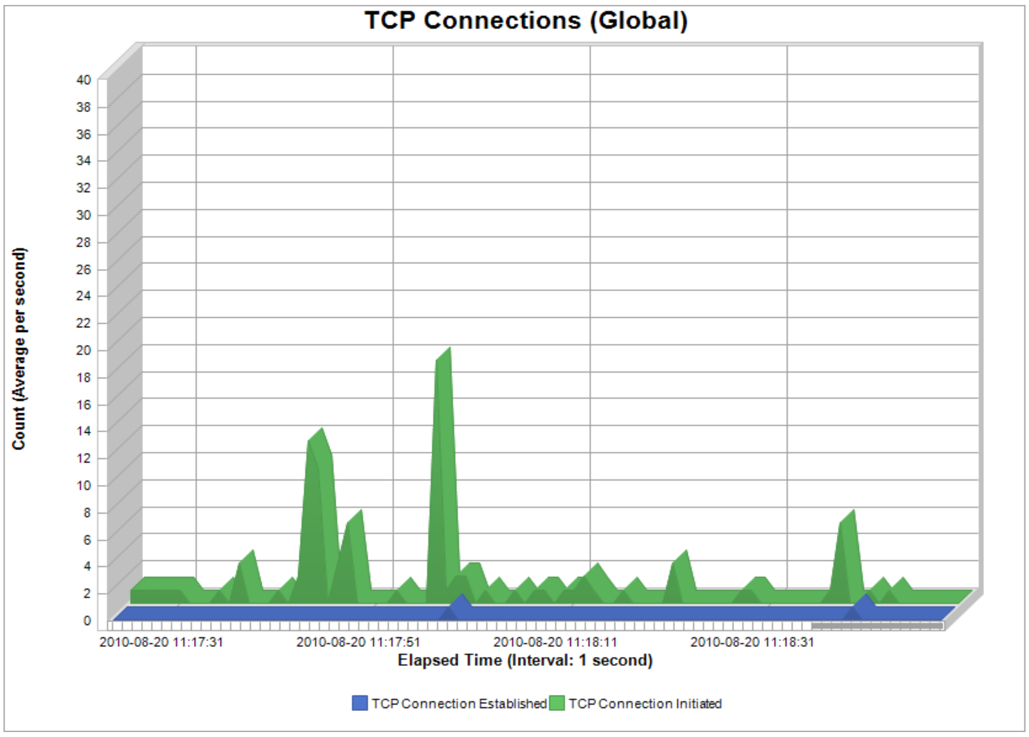


Figure 6-8: traffic classification - TCP connections

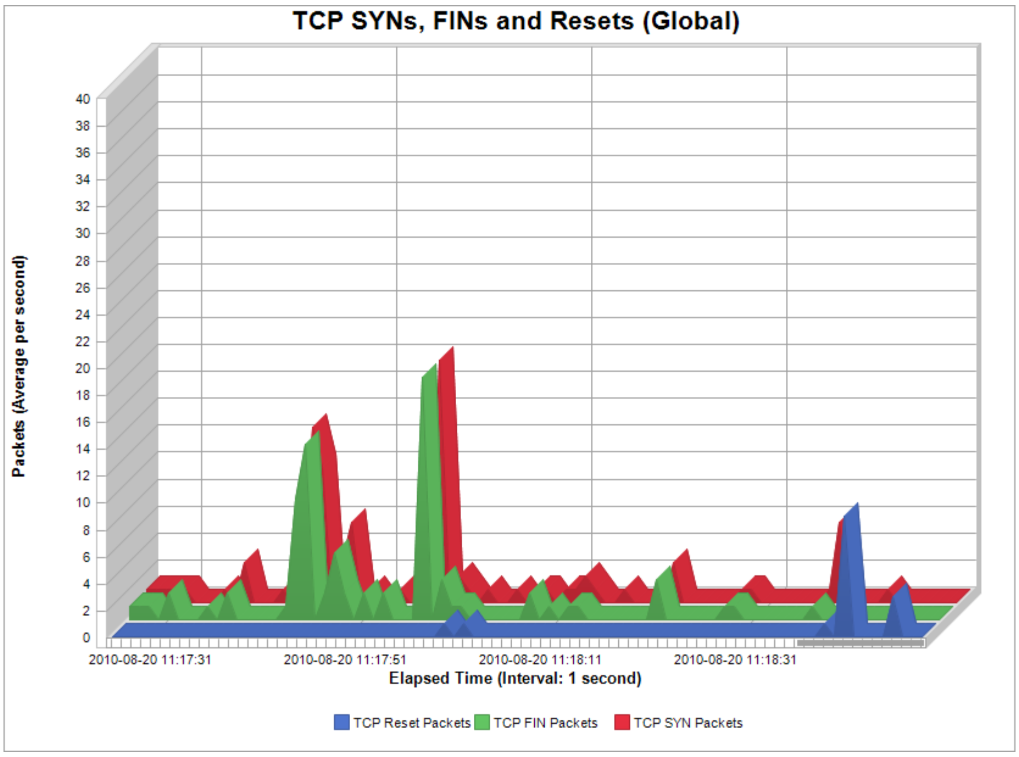


Figure 6-9: Traffic classification - TCP Flags

The average number connections per seconds Figure 6-8 and the average number of packet per seconds Figure 6-9 can be used as indicators as to a serious change in the pattern distributions. A SYN Flood attack is considered to be in progress if the number of unanswered SYN/ACK's sent by the receiving host (half-opened TCP connections) exceeds the threshold set in "Flood rate until attack logged (unanswered SYN|ACKs per second"; on average the default value is 25, the minimum is 5, and the maximum is 999). However, this threshold is protocol dependant and application dependant. In a P2P scenario, dozens or hundreds of connections can be opened at the same time. The threshold set above is mainly for HTTP connections.

Flow manager

In this research the author considers a flow as being the source IP, source port, destination IP, and destination Port. The flow manager as defined by this research will have the following functionalities:

- Organise traffic by flow: tuples of the same nature will be analysed by threat or within the same core (referring to a multicore architecture) unless the threat becomes saturated. In that case, the flow manager would:
- Manage load balancing: MIDaPS is an architecture that is aimed at multicore environment. In a multicore framework, if the load is not well balanced, one core would perform more tasks than order defying the real purpose of multicore. It is important that a balance is kept amongst the flow the ensure maximum performance
- Flow threshold management: In a DDOS scenario, script kiddies usually keep the same tuple in the course of a flooding attack. A load balancing algorithm will continually direct (in the best case) the same tuple to the same threat. This behaviour could result in creating a bottleneck in the

thread. As a solution, the flow manager keeps basic threshold values to ensure that a threat is not overloaded by data

- SYN flood attacks generate infinite number of flow per source IP. The flow manager keeps threshold value regulating the number of simultaneous flow that can be open by a single source IP. In this case, the flow manager would consider a limit of flows in which a source IP can be found.

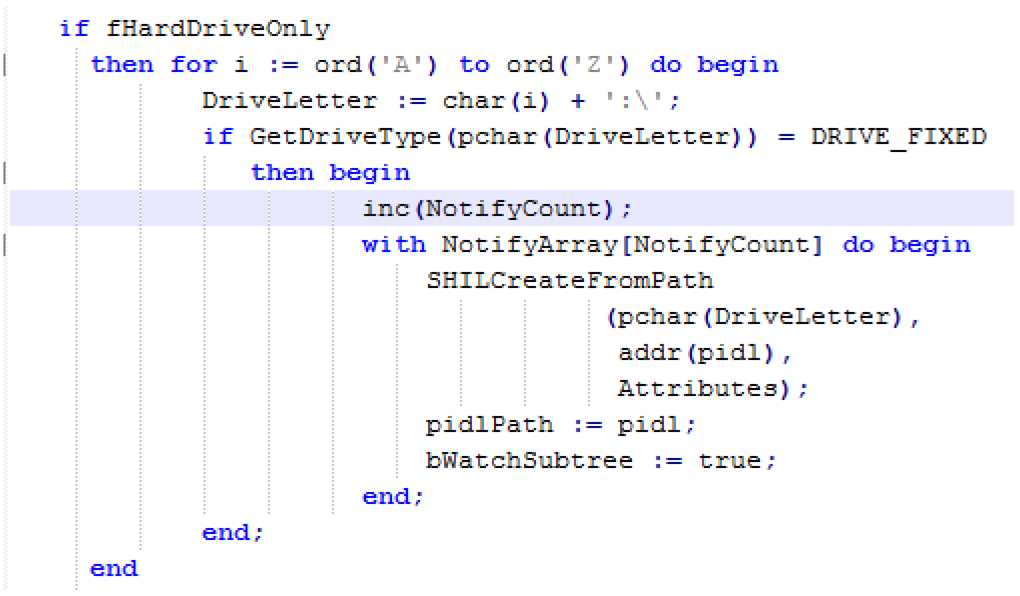
6.3.3 Module 3: the remote monitory agent

This feature does not exist in Snort. The remote external agent acts as a host intrusion detection system. Based on the analysis done previously in this research, it is very important to have knowledge of what is happening in the victim system as well as knowledge of what is happening at the network level. Many of the recent attacks are performed with such sophistication that any IDS will not alert. It is only the combination of all the actions that would indicate the presence of an attack. The remote agent will be responsible for

- Monitoring local shell i.e. file and directory changes
- Monitory CPU usage – this section has been fully discussed in the previous chapter.

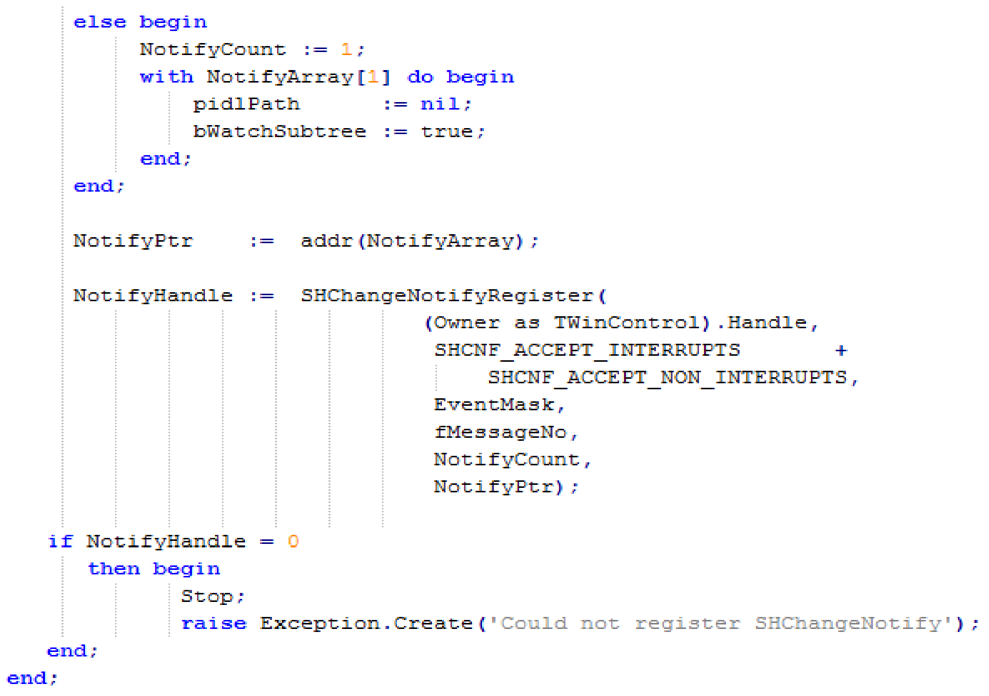
File monitor agent

The list of files to monitor will be specified in a configuration file. Few options are presented. The user can choose to monitor specific hard (Figure 6-10) or choose to monitor the whole system (Figure 6-11)



```
  if fHardDriveOnly
    then for i := ord('A') to ord('Z') do begin
           DriveLetter := char(i) + ':\';
           if GetDriveType(pchar(DriveLetter)) = DRIVE_FIXED
             then begin
                    inc(NotifyCount);
                    with NotifyArray[NotifyCount] do begin
                      SHILCreateFromPath
                                (pchar(DriveLetter),
                                 addr(pidl),
                                 Attributes);
                      pidlPath := pidl;
                      bWatchSubtree := true;
                    end;
           end;
    end
```

Figure 6-10: Monitoring Specific Hard Disk



```
      else begin
          NotifyCount := 1;
          with NotifyArray[1] do begin
              pidlPath      := nil;
              bWatchSubtree := true;
          end;
      end;

      NotifyPtr     :=  addr(NotifyArray);

      NotifyHandle  :=  SHChangeNotifyRegister(
                                (Owner as TWinControl).Handle,
                                 SHCNF_ACCEPT_INTERRUPTS       +
                                     SHCNF_ACCEPT_NON_INTERRUPTS,
                                 EventMask,
                                 fMessageNo,
                                 NotifyCount,
                                 NotifyPtr);
    if NotifyHandle = 0
       then begin
              Stop;
              raise Exception.Create('Could not register SHChangeNotify');
    end;
end;
```

Figure 6-11: Monitoring the whole system

6.3.4 Module 4: Protocol analyser

The protocol analyser is responsible for checking the integrity the traffic making sure that the protocol definition is not violated. Besides checking the integrity of protocols, most protocol analysers do not report when this occurred. Various protocol analysers have already been implemented. The idea in this thesis is not to recreate what is already existent, but rather extend the current capabilities.

6.3.5 Module 5: special features

This module has a fairly limited set of functionalities.

- Whitelists: a set of IPs that have unmonitored access. However, a threshold monitoring will be applied to any of these IP to prevent abuse from the system. Also, whitelists are set IPs that will be allowed to use the system under heavy attacks.
- Corporate proxies: some companies route the internet traffic of their users via proxies or Network Address Translators. By so doing, companies prevent their users to be targeted directly by an external malicious user. However, securing against proxy traffic, especially corporate traffic is very difficult as the actions of users cannot be individually identified.
- Non supported feature: in the occurrence of a non-supported feature, and unless defined in a specific algorithm, a basic threshold will be applied.

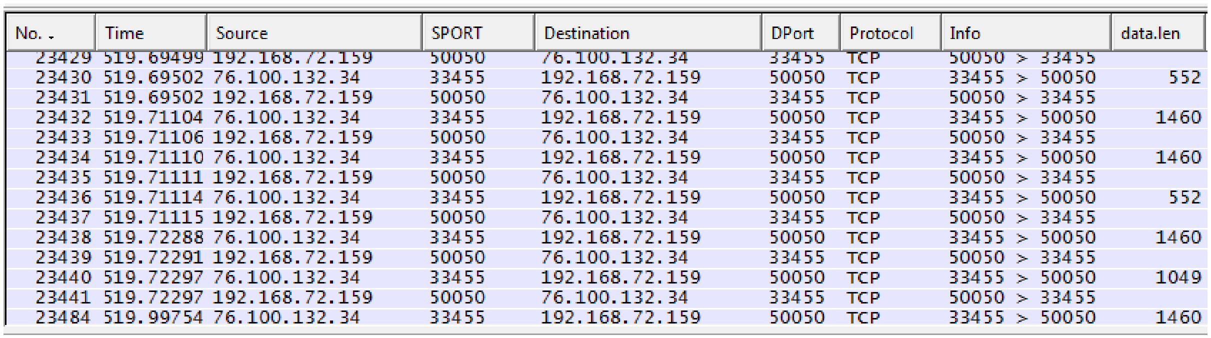
6.3.6 Module 6: Dynamic algorithms

This module is responsible of handling the difficult scenarios. For instance, from the different attacks scenarios of DDOS and multistage attacks discussed in the previous chapter, various patterns were identified.

Table 25: Patterns Identified

SIP	SPORT	DIP	DPORT	PROTO	Data length	Data content hash	note
One	One	Many	Many	UDP	identical	identical	
many	many	one	one	UDP	identical	identical	

The patterns identified were clear and simple to understand. However, further investigations were made to avoid blocking legitimate transactions. The patterns identified are similar to those find in P2P conversations Figure 6-12.



No.	Time	Source	SPORT	Destination	DPort	Protocol	Info	data.len
23429	519.69499	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23430	519.69502	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	552
23431	519.69502	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23432	519.71104	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	1460
23433	519.71106	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23434	519.71110	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	1460
23435	519.71111	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23436	519.71114	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	552
23437	519.71115	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23438	519.72288	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	1460
23439	519.72291	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23440	519.72297	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	1049
23441	519.72297	192.168.72.159	50050	76.100.132.34	33455	TCP	50050 > 33455	
23484	519.99754	76.100.132.34	33455	192.168.72.159	50050	TCP	33455 > 50050	1460

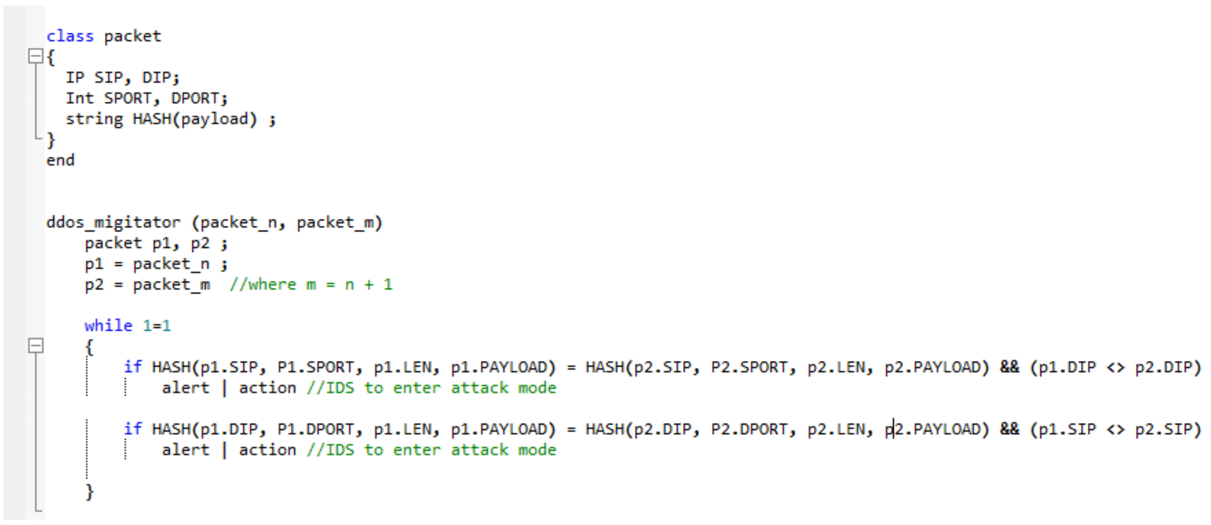
Figure 6-12: P2P traffic

There are important differences between the traffic recorded during the attack and the traffic recorded during P2P conversations.

- Attack used UPD and P2P used TCP
- Some of the packets have the same Len but investigations revealed that the payload content are different
- The traffic is not unidirectional

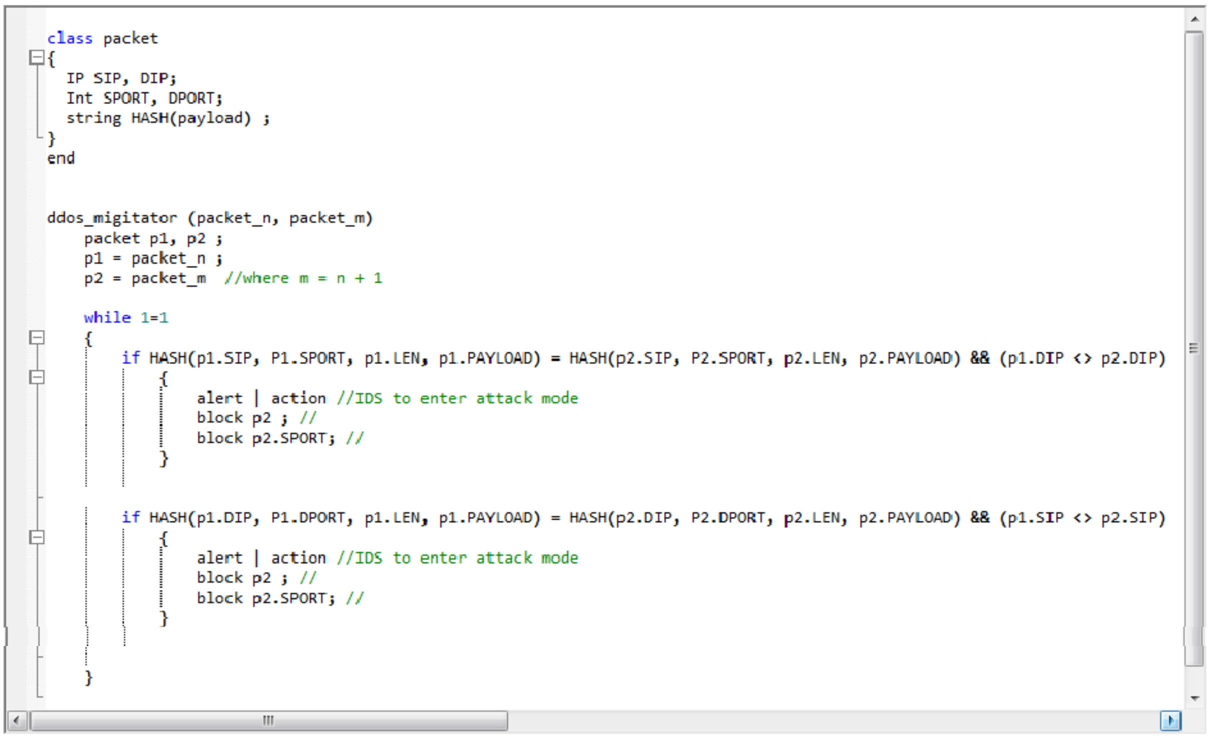
Algorithms

In this algorithm, the system will compare the values that do not change between two packets based on the variances presented in Table 25. When a match is found, the IDS will enter the attack stage, which will then turn the IDS into an IPS.



```
class packet
{
  IP SIP, DIP;
  Int SPORT, DPORT;
  string HASH(payload) ;
}
end


ddos_migitator (packet_n, packet_m)
    packet p1, p2 ;
    p1 = packet_n ;
    p2 = packet_m  //where m = n + 1

    while 1=1
    {
        if HASH(p1.SIP, P1.SPORT, p1.LEN, p1.PAYLOAD) = HASH(p2.SIP, P2.SPORT, p2.LEN, p2.PAYLOAD) && (p1.DIP <> p2.DIP)
            alert | action //IDS to enter attack mode

        if HASH(p1.DIP, P1.DPORT, p1.LEN, p1.PAYLOAD) = HASH(p2.DIP, P2.DPORT, p2.LEN, p2.PAYLOAD) && (p1.SIP <> p2.SIP)
            alert | action //IDS to enter attack mode

    }
```

Various actions can be taken from the moment an attack is detected. For instance, every single IP found matching the rules, and block the port number related to the attack. Blocking the port number will stop all subsequent communication to that port.



```
class packet
{
  IP SIP, DIP;
  Int SPORT, DPORT;
  string HASH(payload) ;
}
end

ddos_migitator (packet_n, packet_m)
    packet p1, p2 ;
    p1 = packet_n ;
    p2 = packet_m  //where m = n + 1

    while 1=1
    {
        if HASH(p1.SIP, P1.SPORT, p1.LEN, p1.PAYLOAD) = HASH(p2.SIP, P2.SPORT, p2.LEN, p2.PAYLOAD) && (p1.DIP <> p2.DIP)
            {
                alert | action //IDS to enter attack mode
                block p2 ; //
                block p2.SPORT; //
            }

        if HASH(p1.DIP, P1.DPORT, p1.LEN, p1.PAYLOAD) = HASH(p2.DIP, P2.DPORT, p2.LEN, p2.PAYLOAD) && (p1.SIP <> p2.SIP)
            {
                alert | action //IDS to enter attack mode
                block p2 ; //
                block p2.SPORT; //
            }

    }
```

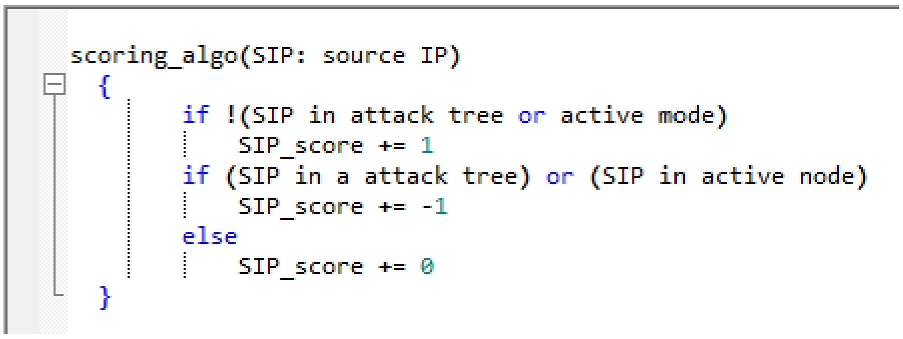
Figure 6-13: Algorithm for complex traffic patterns

6.3.7 Module7: Attack trees

In this work, attacks are organised into attack trees which represents the different ways an attack can occur. This section was developed earlier in “Multistage attack detection and mitigation framework”.

6.3.8 Module 7: scoring algorithm

A scoring algorithm is introduced to prioritise the traffic during the recovery or mitigation period. Based on a configuration file, the administrator will decide which host will have access to the network. Each packet that arrives in the network is analysed and passed through the attack tree system. The participant host will score +1 when there have been no matches with any critical path or active nodes.

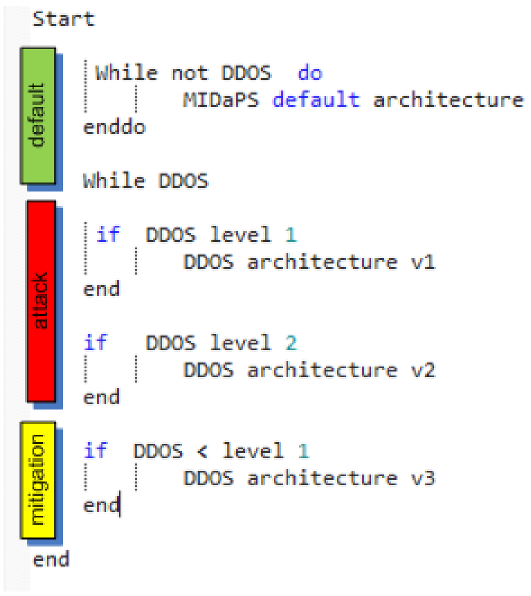


```
scoring_algo(SIP: source IP)
  {
        if !(SIP in attack tree or active mode)
            SIP_score += 1
        if (SIP in a attack tree) or (SIP in active node)
            SIP_score += -1
        else
            SIP_score += 0
  }
```

Figure 6-14: Scoring algorithm

6.4 Attack modes

There are three different modes of functioning of our IDS: a default mode (normal) where the IDS is not subject to particular challenges. During the default mode, a DDOS attack could be detected by an external agent i.e. a malicious attack such as a botnet attack or it could simply be a malfunctioning program that is causing the computer system to freeze. Either way, the DDOS attack should be prevented and stopped. MIDaPS is built to change its behaviour depending on the level of attack.



```
Start
    While not DDOS  do
        MIDaPS default architecture
    enddo

    While DDOS
    if  DDOS level 1
        DDOS architecture v1
    end

    if  DDOS level 2
        DDOS architecture v2
    end

    if  DDOS < level 1
        DDOS architecture v3
    end
end
```

Figure 6-15: MIDaPS Modes

6.5 Additional experiment and results

In this section, results from testing the MIDaPS architecture will be presented. One of the difficulties in testing multistage detection is the availability of data. The logging facilities that exist in current systems are generally triggered when an alert is raised. Unfortunately, this does not represent the full picture of the attack and does not allow a good offline analysis. Most offline analyses are based on alert rather than related traffic.

During the hacking conference context named DEFCON 17, data were capture and made public for purposes such as this theses [188].

Objectives of tests:

The MIDaPS architecture has been implemented and, the efficiency of its detection mechanism tested as long with the effectiveness of using attack tree in a multistage environment. In addition, the number of false positive tested and compared with Snort.

Data: the results presented here have been tested using the 7.5GB of data provided by DEF CON 17.

Test 1: in this first series of tests, the analysis is based on the source IP.

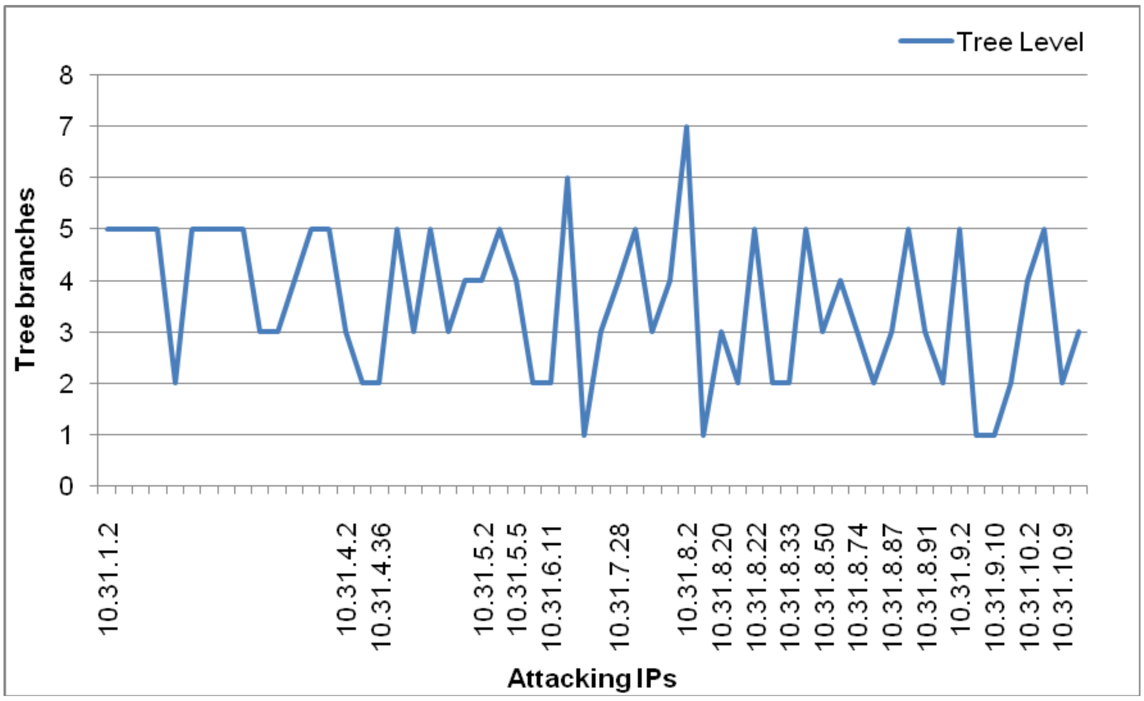


Figure 6-16: Tree attack detection

In this test Figure 6-16, an analysis of how often an attack IP appears in a tree. This literally represents the number of steps taken by a particular attacker. For instance, IP = 10.31.6.100 was found to match fives nodes of attacks when attacking IP = 10.31.4.2. The different steps are given in [Table 26: Attack tree breakdown]

Attack	Occurrences	Start time	End time
Portscan - TCP Portscan	116	2009-08-01	2009-08-02

		00:05:23	21:47:59
Portscan - Open Port	18330	2009-08-01 00:00:26	2009-08-02 21:59:38
portscan TCP Decoy Portscan	65	2009-08-01 22:27:22	2009-08-02 21:02:54
portscan TCP Portsweep	113	2009-08-01 00:00:24	2009-08-02 21:59:28
portscan TCP Distributed Portscan	3	2009-08-01 01:50:19	2009-08-02 04:26:08

Table 26: Attack tree breakdown

IP = 10.31.6.100, was found in six different nodes. However, each attack step was performed many times as represented in Figure 6-17. During the completion at which the data used here was collected, they was no security IDS installed on the system.

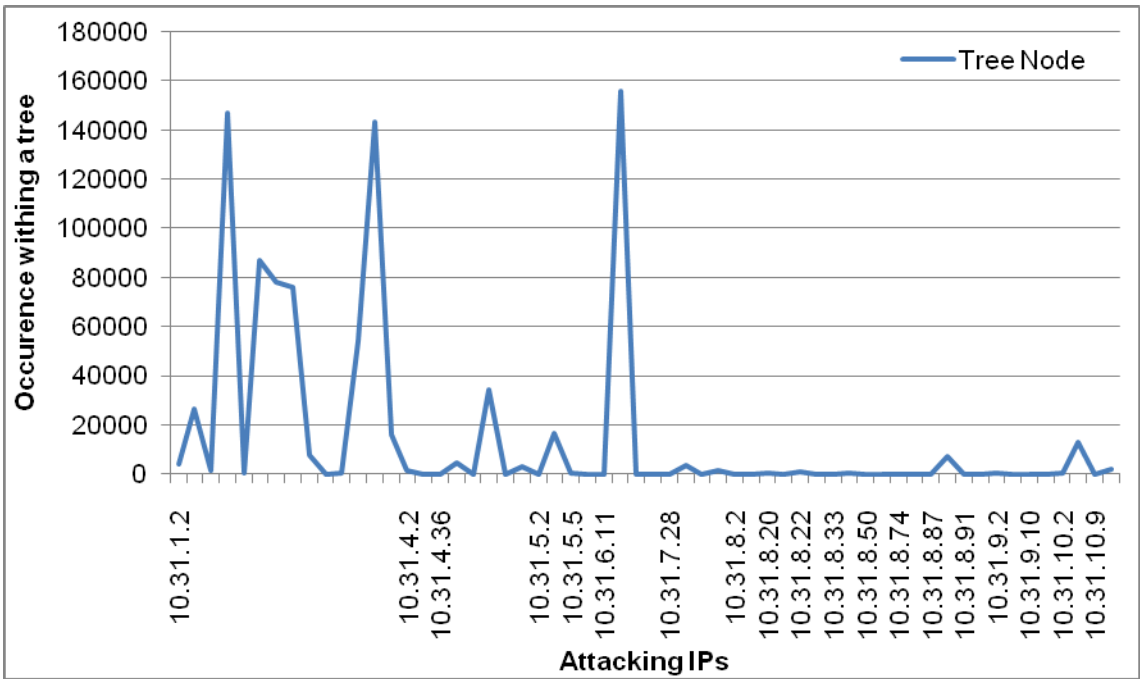


Figure 6-17: attacking IPs occurrence per tree path

However, the data was replayed in Snort during an offline analysis and compared with the result obtain in MIDaPS

Comparing Snort to MIDaPS is rather interesting as the two system aim at protecting computer system but they are actually very different in their operations. The objective of MIDaPS is not to generate a security alert for each step of the attack. Rather, attacks are synchronised in an intelligent way to reduce the output as much as possible. It was noted that some of the IP performed exactly the same sequence of attacks to various IPs. This is where there is an important difference between Snort and MIDaPS. Snort does not keep any record of what has happened, Snort does not keep any history of the different IPs activities. In MIDaPS, each IP activity is logged. The logged information then becomes valuable information for future actions by the same IP. Snort looks at identifying each attack or occurrence of partial attack as a separate alert. In MIDaPS, attacks are analysed as they go through the tree structure. When an attacker has been found on at least one active node, then the alert is generated and the attacker is stopped. Alternatively, if there is no active node, the attacker would need to reach a level of the tree marked as critical. An alert will then be generated.

Most attacks were found to in at least five different nodes Figure 6-18. This is another clear indication that if an IP is blocked after two attempts, the system will save on resources as they will be no need for further check on the same IP.

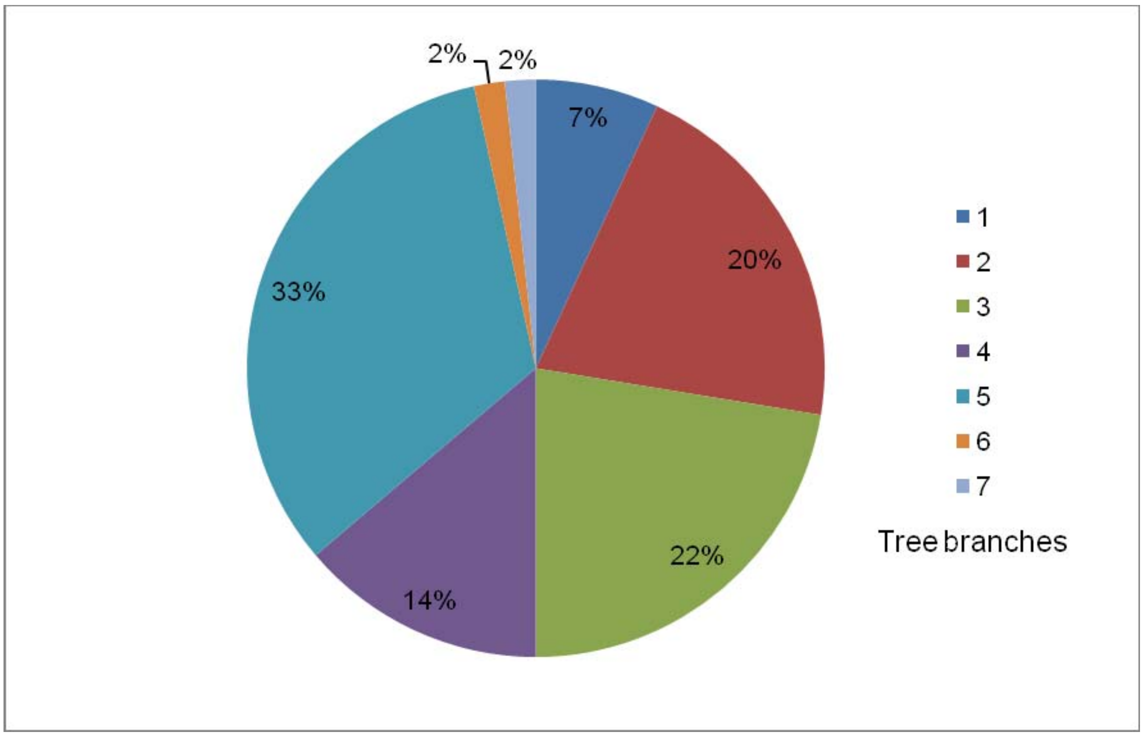


Figure 6-18: Attacking IPs per tree node

Attacking IPs	Total occurrence in tree	Tree level
10.31.8.2	79	7
10.31.6.100	155974	6
10.31.1.2	3862	5
10.31.2.3	26273	5
10.31.2.100	1363	5
10.31.3.103	147071	5
10.31.3.110	87112	5
10.31.3.129	77955	5
10.31.3.130	76224	5
10.31.3.140	7889	5
10.31.3.172	143212	5
10.31.3.175	16170	5
10.31.4.99	4357	5
10.31.4.152	34284	5
10.31.5.3	16806	5
10.31.7.99	3295	5
10.31.8.22	701	5
10.31.8.40	142	5

10.31.8.90	7251	5
10.31.9.2	572	5
10.31.10.3	12964	5
10.31.3.160	53967	4
10.31.4.254	3132	4
10.31.5.2	18	4
10.31.5.5	339	4
10.31.7.28	40	4
10.31.7.199	1177	4
10.31.8.51	25	4
10.31.10.2	411	4
10.31.3.141	57	3
10.31.3.153	141	3
10.31.4.2	1549	3
10.31.4.123	82	3
10.31.4.201	9	3
10.31.7.2	3	3
10.31.7.156	11	3
10.31.8.20	142	3
10.31.8.50	16	3
10.31.8.74	7	3
10.31.8.87	45	3
10.31.8.91	119	3
10.31.10.10	1898	3
10.31.3.109	314	2
10.31.4.13	22	2
10.31.4.36	5	2
10.31.6.2	15	2
10.31.6.11	4	2
10.31.8.21	20	2
10.31.8.30	8	2
10.31.8.33	5	2
10.31.8.79	25	2
10.31.8.142	3	2
10.31.9.17	7	2
10.31.10.9	8	2
10.31.6.218	5	1
10.31.8.10	1	1
10.31.9.6	3	1
10.31.9.10	7	1

Table 27: Attacking IPs per tree nodes

Interpretation

In this experiment, the author has been able to identify 58 unique occurrences of attack paths. However, each of these attack paths have been repeated a number of time. The largest number of attacks found had five branches which in order terms represent five stages in attacks. There is a tendency that when an IP is found using a port scan technique, that it appears many times at the same node level in the tree architecture. It is important that threshold values should be set in order to avoid a large number of false positive. Blacklisting appears to be an efficient method of reduce the number of attack occurrences by the same IP. Once the IP is blacklisted, all subsequent attacks from the same IP will not exist.

From these results, one would learn that an IP can be stopped before damage is done in most case when using attack trees. Attack detection rate really relies on the efficiency of the nature of the attack tree. The more elaborate an attack tree is, the more attacks will be detected. However, it is important to define critical paths as the model defined here uses active nodes and passive nodes.

Test 2: In this second series of tests, the analysis is based on the destination IP. The rationale behind analysing traffic based on the destination IP is that the number of destination IPs is generally known as compared to the number of source IP that can be unlimited. The results are as follow:

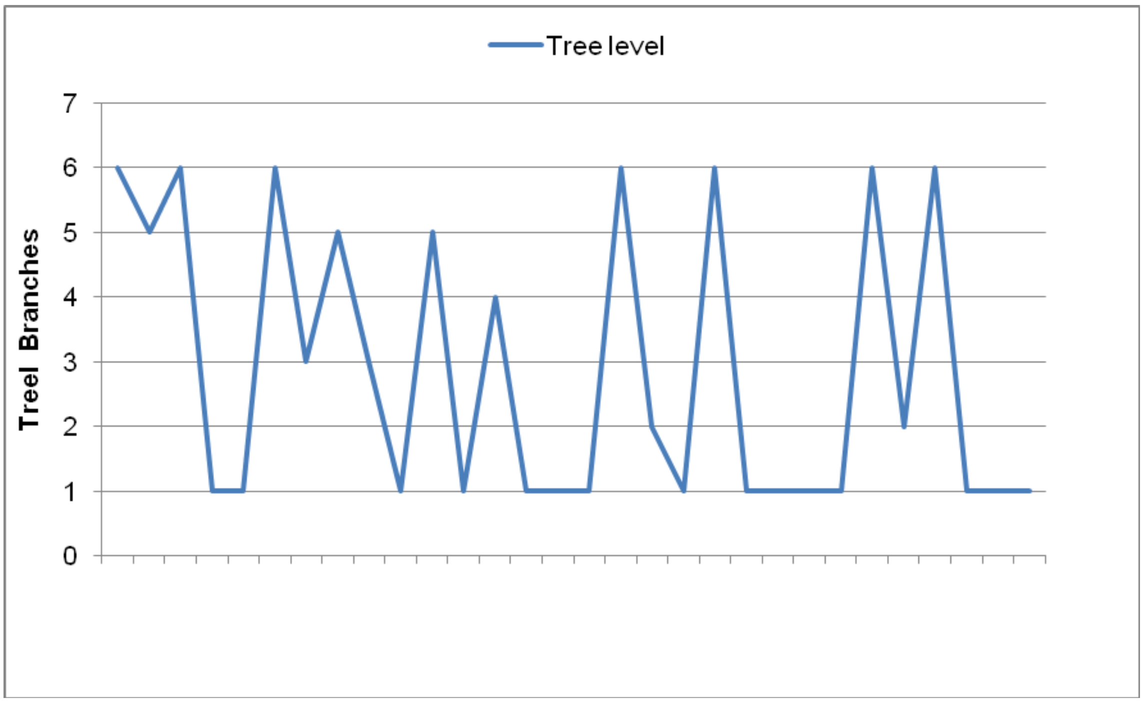


Figure 6-19: attack tree view from destination IP

From the receiving point, we look at how often an IP was hit by an attack. as shown in Figure 6-19, each IP was hit at least one by a type of attack and at most by 6 types of different attacks. The reason why this research looks at book the attacker and the victim is that attacker can easily obfuscate their identity by using a proxy server or by using a combination of virtual machines installed on the same host. Each virtual machine installed on a host will have its own IP address but they will all have the same netbios name. if the attacking IP is hit by a number of attacks that match a certain path in the attack tree, all subsequent attack attempts will be blocked immediately. In addition, the overall level of attack can be raised.

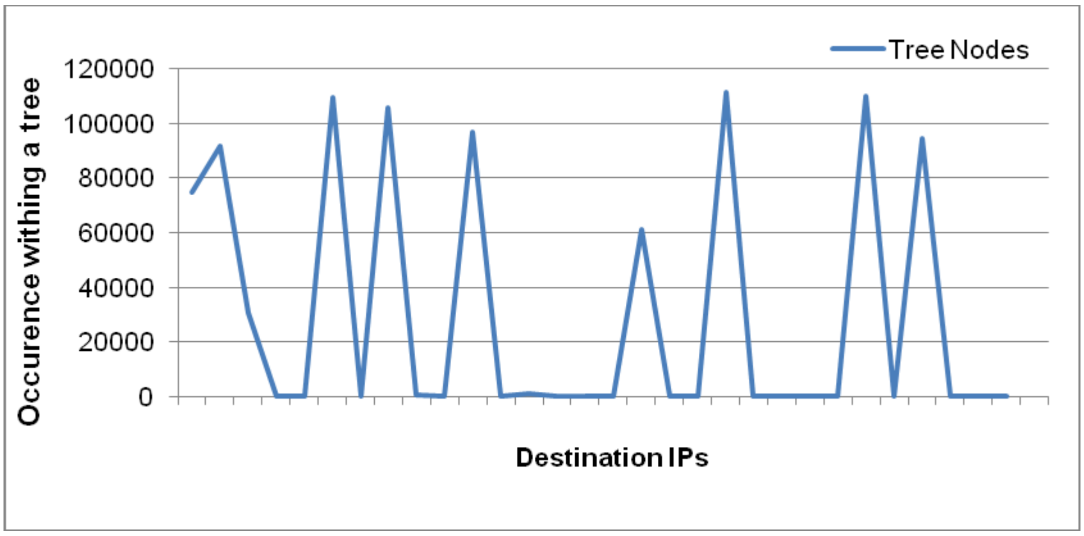


Figure 6-20: attacks view from the destination IPs

Interpretation

Looking at attacks from a destination IP, each IP was the most attacks in 6 stages. Compared to looking at attacks from source IPs, there is clearly less effort to look at destination address as the number of destination IP is generally very limited. The trees visible by each destination address are different from the trees viewed by source IPs. A source IP may attempt to attack multiple destination addresses. However, one destination IP will not have any knowledge of the activities (attacks) taking place at other destination address.

Also, the number of attempts per source is much higher that the number of attempts received by each destination IP. Protecting against attacks by protecting destination IP rather than tracking source IP is definitely more efficient.

6.6 False positive rate

The number of alert between Snort and MIDaPS was compared. For the same subset of data, Snort raised 411170 alerts whereas MIDaPS fired only 278861 alert. The difference is due to the fact Snort does not keep any history of IP activities. When an IP is found to repeat the same attack at most twice, the later IP is suspended and further activities are blocked.

Snort	MIDaPS
451170	278861

Table 28: False positive result

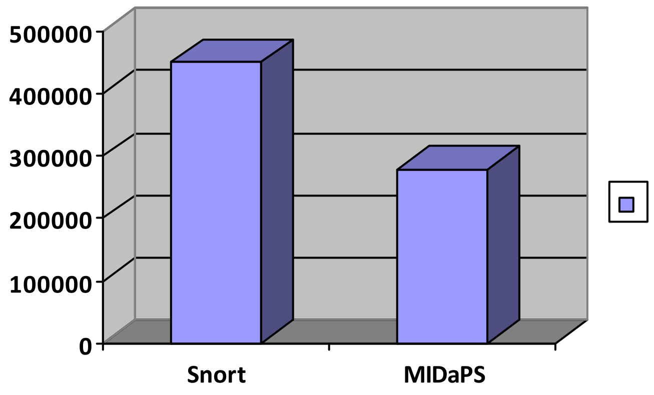


Figure 6-21: Snort vs. MIDaPS false positive

As shown in Figure 6-21 and Table 28, MIDaPS produced 38.19% less alerts than Snort

6.7 Conclusion

This chapter summarises the different efforts made to design a multistage detection and mitigation intrusion detection system. The author argue that with the level of attacks encountered in recent months and years, any security system should provide preventive and defensive capabilities. This research defined a four level visibility of attacks around which the MIDaPS framework was built: the V-BANI framework. The author argues that any good system should be able to provide those four levels of visibility. Also, the author assembles the different parts of this research into a novel and highly effective solution: MIDaPS which is a multistage Intrusion Detection and Prevention System. MIDaPS is built around 3 main modes of functioning: default, attack and mitigation. Experiments performed in this research show that IDS are more likely to drop packets without analysing them when the speed of packets increased. This is generally the case with DDOS attack. This research then defined a different mode of functioning for our architecture depending on the level of severity of the attack to ensure that legitimate users can continue using their services.

Tests of MIDaPS were performed based on the detection system. During the tests, results identified 58 attack trees that repeated many times. Based on this detection, alerts can be reduced and IPs can be blacklisted to avoid the same IPs causing the same repetitive attacks. A great deal of research still need to be done to have a fully system ready off the shelves but the author strongly believes that the work achieved in this will set the ground for future research.

7 Conclusions and future work

This chapter will start by summarising the outcomes that have been accomplished and the make recommendations accordingly for future work at a higher level.

7.1 Thesis Contribution

In this research, the current design and implementation of Snort IDS was challenged. Numerous vulnerabilities were found from which the most important are discussed below.

Snort weakness in handling fast traffic for any protocol. The level of packet drop was very high when the traffic was above 1.5mbps. Snort was not able to detect up to 26% of IPs when the traffic was accelerated. An attacker could look a network with a lot of noise to perform attacks. When packets are dropped at the IDS, they traverse the network without any prior analysis. Hence, the network is exposed to any sort of attack that such packets will carry. This weakness was address by adding another dimension to Snort rules.

Snort displayed an inability to detect HTTP DDOS attack when many IPs are used. Snort was able to detect repetition for an attack using 150 IPs or less; yet, current implementations of botnet use thousands, hundreds of thousands and up to, in some cases, few millions of IPs all at once. The consequence of this weakness is that Snort will not protect or detect any recent DDOS attacks.

Damages caused by DDOS attacks can be tragic if the target system is a Critical National Infrastructure (CNI). MIDaPS is designed to stop such attacks.

Snort does not provide any mechanism of securing against encrypted attack. Even though they are well known encryption algorithm used by attackers, Snort does not provide any mechanism for analysis encrypted attacks. MIDaPS, the novel IDS introduced source code analysis. Many patterns of encryption coupled with the attacker activity can be used to identify attacks. Hence the new approach of attack tree with passive nodes. Attackers use the latter technique to steal information.

Snort remained blind to JavaScript encrypted attack. Yet, attacker use commonly used tools to obfuscate their script for which de-obfuscator are also available. The author suggested in the novel IDS to use a slow path for analysis of ambiguous source code.

In a multistage scenario, Snort was only able to detect very little indication that an attack was taking place. MIDaPS was designed with attack radar to indicate an early stage of attacks. This new feature could be a life saver for critical business who can then take actions before any serious damage is done.

Snort rules are not optimized for performance and this causes the system to be less efficient in detecting attacks. When checking the rules, Snort spent a lot of time checking rules that are not relevant to the system protected. As part of the new architecture, the author designed and tested a new rules extension that ensured that only rules relevant to the system being monitored are checked. Snort performance was greatly improved allowing more packets to be checked before they are passed into the network.

Snort architecture is built with sequential implementation. Yet, hardware is more capable of handling heavy process by using concurrent processing. This clearly indicates the need for a parallel IDS implementation. MIDaPS is designed with multicore capabilities. This feature will improve the IDS performance.

The ultimate aim of this research was to produce a new IDS architecture capable of multistage attacks whilst working in a multicore framework. The architecture was presented and named MIDaPs. Within the overall architecture a few distinct elements can be noted

- An extension to Snort rules that enable the IDS to only search through the rules relevant to the protected system. An improvement of 84% was achieved with our system compared to the current performance of Snort
- A multistage detection architecture capable of analysing stealthy attack. Also, the architecture presented is capable of behaviour analysis
- A DDOS framework capable of detecting most DDOS attacks with a record of detecting flooding attack within 10 minutes of the start of the attack. It is important to note that the DDOS engine can be implemented as a separate complete unit.
- A new IDS architecture capable of detecting multistage attacks and DDOS attacks while compatible with multicore parallel programming.
- A four level of attack visibility framework that maps every single attack

7.2 Challenges and limitations

Throughout the research the author has looked at designing a new Intrusion Detection System. The journey has certainly not been an easy one. There have been many constraints the author have faced. Some of these constraints are discussed below.

The wide range of elements to consider when designing a new IDS. Unfortunately, the expertise the author had was somewhat limited. A team of people with different and strong skills would be required to design and implement an IDS. In the design presented in this thesis, a strong emphasis was put on the security aspect. Not much consideration was giving to the physical limitations of computer systems. As noted, there are a good number of elements to integrate into MIDaPs to ensure maximum security. Some of these elements will require the system to decode traffic before it is analyzed. This could have a serious impact on the performance. More studies need to look at the performance implications of decoding traffic during live traffic analysis.

Hardware knowledge: the hardware knowledge that the author has is limited in the sense that a full understanding is needed on how the components integrate together in a computer. During an industrial experience, as I designed the security framework for a 10GB IDS appliance, there were frequent clashes between the design and the hardware capabilities. Even though the design presented in this work is not for hardware, a full understanding and a full study of multicore systems needs to be done for the best integration of the different component that were suggested during the design.

Ability to fully test the system: The range of tests performed in this research was limited as the full system has not yet been implemented. Individual tests may well be successful, but do not guarantee that once all the components are all integrated together, the same level of success will be achieved.

Lack of data: Even though there are some data available, there is not, to be best of the Author knowledge, a set of data available that leverages the level of technicalities seen in recent attacks. However, this research has made intensive use of data capture at honeynet and on site while working as a consultant. More studies need to be done on how to use honeynet to generate attack patterns. Manually writing attack patterns and attack tree is a tedious task and this may not allow the security community to keep up with malicious users.

Time: the time has been a serious issue. Ideally, each of the processes suggested should have been fully tested. Vern Paxon and his research group have been working for over four years now to come up with the best implementation of IDS into multicore. Even though the security specifications are ready, it is another matter to implement it. Intel Corporation performed a quick modification of Snort to prove the point of their multicore device. However no technical improvement was brought to Snort. Yet, Snort on itself was found vulnerable at different levels during our studies. Each process should have been full tested with various approaches and algorithms. In addition, a full study of the cost of adding each of the features should be done in order to have a realistic implementation where security does not interfere with performance or at least to an acceptable level.

Lab equipment: At the time of doing the research, the lab equipment available was fairly basic. However, even though more equipment was added to the lab, there was not enough time to repeat all the experiments with more sophisticated material.

The transition between classic research and applied research: Most research is based on theoretical model that generally do not end in a lab for production. In this work, the author aimed at presenting a piece of work that, with some improvement, will be able to go to production.

Finance: the availability of finance has been a major problem during the course of this research. The work had to be interrupted on so many occasions because of the lack of finance. Fortunately, opportunities allowed the Author to not only gain a lot of practical experience within the industry but also, gave him enough funding to complete his work for which he is extremely grateful.

The DDOS framework that was designed does not take into consideration the analysis of traffic generated by human users against the traffic generated by a bot army. More studies that would probably require a very complex mathematical model would be needed. The Author does not have the mathematical knowledge required for such analysis. This section would require some strong mathematical computations.

7.3 Recommendations and future work

The nature of this work in itself is a challenge which has generated many other challenges which can serve as full research projects. These projects could be:

- Consequence analysis of interdependencies and potential cascading effects across related processes within the MIDaPs framework.

- Develop and execute a coordinated research to fully utilize the potential of honeypot, honeynet and honeyfarm to capture and analyse attacks trends; to generate complex detection algorithms; to build attacks tree and attack patterns; and to predict possible new attack patterns
- Design and implementation of a parallel implementation of libnet, the default packet used for packet capture.
- Design and develop an attack tree capable of a full integration for a multicore environment.
- Develop and execute a coordinated research to model strictly legitimate traffic against HTTP flooding whereby an attack will launch an army of bots to download various pages on the websites.

References

[1] TechTarget, “What is multi-core processor? - Definition from Whatis.com,” 2004. [Online]. Available: http://searchdatacenter.techtarget.com/definition/multi-core-processor. [Accessed: 10-Nov-2009].

[2] B. Howard, “Analyzing online social networks,” *Commun. ACM*, vol. 51, no. 11, pp. 14-16, 2008.

[3] E. Zheleva, L. Getoor, J. Golbeck, and U. Kuter, “Workshop on Social Network Mining and Analysis,” *The 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008)*, 2008. .

[4] S. Mansfield-Devine, “Anti-social networking: exploiting the trusting environment of Web 2.0,” *Network Security*, vol. 2008, no. 11, pp. 4-7, 2008.

[5] J. Richards, “Cyber-War--the Way of the Future? Times Online (UK) (05/17/07),” *ACM TechNews*, 2007.

[6] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, “Inferring Internet denial-of-service activity,” *ACM Trans. Comput. Syst.*, vol. 24, no. 2, pp. 115-139, 2006.

[7] R. Bajcsy et al., “Cyber defense technology networking and evaluation,” *Commun. ACM*, vol. 47, no. 3, pp. 58-61, 2004.

[8] J. Yen, “Introduction,” *Commun. ACM*, vol. 47, no. 3, pp. 32-35, 2004.

[9] M. Jonkman, “Working Group Kick Off,” *Open Information Security*

Foundation, 2009. [Online]. Available: http://www.openinfosecfoundation.org/.

[10] V. Paxson, R. Sommer, and N. Weaver, “An architecture for exploiting multi-core processors to parallelize network intrusion prevention,” 2009. [Online]. Available: http://www.icir.org/vern/papers/multicore-sarnoff07.pdf.

[11] Y. Xiang and W. Zhou, “Using Multi-Core Processors to Support Network Security Applications,” *Proceedings of the 2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*, 2008. .

[12] E. Ramraj and A. S. Rajan, “Using Multi-core Processor to Support Network Parallel Image Processing Applications,” *2009 International Conference on Signal Processing Systems*, 2009. [Online]. Available: http://ieeexplore.ieee.org/iel5/5166727/5166728/05166782.pdf?isnumber=5166728&arnumber=5166782.

[13] M. Oskin, “How changes in computer architecture are about to impact everyone in the IT business,” *Communications of the ACM*, vol. 51, no. 7, pp. 70-78, 2009.

[14] P. Wheeler and E. Fulp, “A taxonomy of parallel techniques for intrusion detection,” *Proceedings of the 45th annual southeast regional conference*, 2007. .

[15] Intel Corporation, “Supra-linear Packet Processing Performance with IntelÂ® Multi-core Processors,” *White Paper*, 2006. [Online]. Available: http://www.intel.com/technology/advanced_comm/311566.htm.

[16] P. D. Ungsunan, L. Chuang, W. Yang, and G. Yi, “Network processing performability evaluation on heterogeneous reliability multicore

processors using SRN model," *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1-6, 2009.

[17] E. Mollick, "Establishing Moore's Law," *Annals of the History of Computing, IEEE*, vol. 28, no. 3, pp. 62-75, 2006.

[18] A. Valdes and D. Zamboni, *Recent advances in intrusion detection : 8th international symposium, RAID 2005, Seattle, WA., USA, September 7-9, 2005 : revised papers*. Berlin ; New York: Springer, 2006.

[19] J. Pagna Disso, *fx-http-traffic-generator - http traffic generator*. google, 2008.

[20] J. Bashor, "Berkeley Labâ€™s Vern Paxson Honored for Research Characterizing the Internet," *Research News Berkerly Lab*, 2008.

[21] H. Berghel, "Hiding data, forensics, and anti-forensics," *Commun. ACM*, vol. 50, no. 4, pp. 15-20, 2007.

[22] E. Casey, "Investigating sophisticated security breaches," *Commun. ACM*, vol. 49, no. 2, pp. 48-55, 2006.

[23] D. Maynor and K. K. Mookhey, "Metasploit Framework and Advanced Environment Configurations," in *Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research*, Burlington: Syngress, 2007, pp. 77-83.

[24] D. Maynor and K. K. Mookhey, "Introduction to Metasploit," in *Metasploit Toolkit for Penetration Testing, Exploit Development, and Vulnerability Research*, Burlington: Syngress, 2007, pp. 1-64.

[25] ITOC, "ITOC Research: CDX Datasets," 2009. [Online]. Available: http://www.itoc.usma.edu/research/dataset/index.html. [Accessed: 13-May-2010].

[26] R. G. Bace, *Intrusion detection*. Indianapolis, IN: Macmillan Technical Publishing, 2000.

[27] P. P. Purpura, *Security and loss prevention : an introduction*. Oxford: Elsevier Butterworth-Heinemann, 2008.

[28] A. R. Baker, B. Caswell, and M. Poor, *Snort 2.1 intrusion detection*. Rockland, Mass. ; [Great Britain]: Syngress, 2004.

[29] R. Sommer and V. Paxson, “Enhancing byte-level network intrusion detection signatures with context,” *Proceedings of the 10th ACM conference on Computer and communications security*, 2003. .

[30] H. F. Tipton and M. Krause, *Information security management handbook*. Boca Raton, Fl ; London : Auerbach, 2003-, 2005.

[31] P. Fan and H. Shen, *Parallel and Distributed Computing, Applications and Technologies : PDCAT'2003 : proceedings : [August 27-29, 2003, Chengdu, China*. Piscataway, New Jersey: IEEE, 2003.

[32] J. K. Jones and G. W. Romney, “Honeynets: an educational resource for IT security,” *Proceedings of the 5th conference on Information technology education*, 2004. .

[33] C. Kreibich and J. Crowcroft, “Honeycomb: creating intrusion detection signatures using honeypots,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 51-56, 2004.

[34] M. Becchi and P. Crowley, “Extending finite automata to efficiently match Perl-compatible regular expressions,” *Proceedings of the 2008 ACM CoNEXT Conference*, 2008. .

[35] B. C. Brodie, D. E. Taylor, and R. K. Cytron, “A Scalable Architecture For High-Throughput Regular-Expression Pattern Matching,” *Proceedings of*

the 33rd annual international symposium on Computer Architecture, 2006. .

[36] A. Mitra, W. Najjar, and L. Bhuyan, “Compiling PCRE to FPGA for accelerating SNORT IDS,” *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2007. .

[37] J. Moscola, J. W. Lockwood, and Y. H. Cho, “Reconfigurable content-based router using hardware-accelerated language parser,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 13, no. 2, pp. 1-25, 2008.

[38] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel, “Anomalous system call detection,” *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 61-93, 2006.

[39] Y. Chen and Y. Chen, “Combining incremental Hidden Markov Model and Adaboost algorithm for anomaly intrusion detection,” *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*, 2009. .

[40] C. Kruegel and G. Vigna, “Anomaly detection of web-based attacks,” in *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 251-261, 2003.

[41] C. Kruegel, G. Vigna, and W. Robertson, “A multi-model approach to the detection of web-based attacks,” *Computer Networks*, vol. 48, no. 5, pp. 717-738, 2005.

[42] V. Paxson, “Bro: a system for detecting network intruders in real-time,” *Computer Networks*, vol. 31, no. 23, pp. 2435-2463, 1999.

[43] V. Paxson, “Bro Intrusion Detection System Hands-On Workshop: Bro Futures,” 2007. [Online]. Available: http://www.bro-ids.org/bro-workshop-2007/slides/Overview.pdf.

[44] E. Messmer, “Intruvert Inspects high-speed IP traffic,” *Network World*, vol. 9, no. 36, p. 84, 2002.

[45] J. Goldman, “ISP-Planet - Value-Added Services - Intrusion Detection Systems - IntruVert Networks,” *ISP-Planet*, 2002. [Online]. Available: http://www.isp-planet.com/services/ids/intruvert.html. [Accessed: 04-Nov-2010].

[46] McAfee, “Data Sheet | McAfee Network Protection Solutions,” McAfee, 2006.

[47] Check Piont, “IntruVert Networks Inc.: IntruShield IDS,” *OPSEC Partners*, 2010. [Online]. Available: http://www.opsec.com/solutions/partners/intruvert.html. [Accessed: 04-Nov-2010].

[48] S. Staniford-chen et al., “GrIDS - A Graph Based Intrusion Detection System For Large Networks,” *IN PROCEEDINGS OF THE 19TH NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE*, vol. 1, pp. 361--370, 1996.

[49] V. Bamm, “Sguil - Open Source Network Security Monitoring,” *Sguil*, 2007. [Online]. Available: http://sguil.sourceforge.net/. [Accessed: 04-Nov-2010].

[50] NSM, “Sguil on RedHat HOWTO - NSMWiki,” 2008. [Online]. Available: http://nsmwiki.org/Sguil_on_RedHat_HOWTO. [Accessed: 04-Nov-2010].

[51] Interoute, “Internet Barometer,” 2009. [Online]. Available: http://barometer.interoute.com/barom_main.php.

[52] T. Sanders, “Botnet operation controlled 1.5m PCs,” *Hacking*, 2005. [Online]. Available: http://www.v3.co.uk/vnunet/news/2144375/botnet-

operation-ruled-million.

[53] Team-Cymru Research, “Monitoring Graphs,” 2009. [Online]. Available: http://www.team-cymru.org/Monitoring/Graphs/.

[54] O. Fletcher, “DNS Attack Downs Internet in Parts of China,” *PC World*, 2009. [Online]. Available: http://www.pcworld.com/businesscenter/article/165319/dns_attack_downs_internet_in_parts_of_china.html.

[55] Symantec Global Internet, “Security Threat Report Trends for 2008,” *Security Threat Report*, 2008. .

[56] Arbor Network Inc,, “Network Infrastructure Security Research Report | Arbor Networks,” 2008. [Online]. Available: http://www.arbornetworks.com/report. [Accessed: 19-May-2010].

[57] M86 Security, “M86security Security Labs Report Jul 2009-Dec 2009 Recap,” 2010. [Online]. Available: http://www.m86security.com/newsimages/trace/M86_Labs_Report_Jan2010.pdf. [Accessed: 13-Mar-2010].

[58] M. Lesk, “The New Front Line: Estonia under Cyberassault,” *IEEE Security and Privacy*, vol. 5, no. 4, pp. 76-79, 2007.

[59] TeleGeography Research, “Overview: Global Bandwidth: Research Products,” 2008. [Online]. Available: http://www.telegeography.com/products/gb/.

[60] Office For National Statistics, “Internet Access Household and Individuals,” *National Statistics*, 2008. [Online]. Available: http://www.statistics.gov.uk/pdfdir/iahi0808.pdf.

[61] Ofcom, “Ofcom reveals UK's average broadband speed,” 2009. [Online].

Available: http://www.ofcom.org.uk/media/features/brspeeds.

[62] I. Advanced Micro Devices, “MULTI-CORE PROCESSORS â€” THE NEXT EVOLUTION IN COMPUTING,” *White Paper*, 2005. [Online]. Available: http://multicore.amd.com/Resources/33211A_Multi-Core_WP_en.pdf.

[63] V. Romanchenko, “Evolution of the multi-core processor architecture Intel Core: Conroe, Kentsfield,” *CPU & Memory*, 2006. [Online]. Available: http://www.digital-daily.com/cpu/new_core_conroe/.

[64] G. Seibert, “Scaling networking applications to multiple cores,” *MontaVista Vision 2008 Embedded Linux Developers Conference*, 2008. .

[65] M. Marino, “L2-Cache Hierarchical Organizations for Multi-core Architectures,” 2006, pp. 74-83.

[66] T. Gamer, “Distributed detection of large-scale attacks in the internet,” *Proceedings of the 2008 ACM CoNEXT Conference*, 2008. .

[67] J. Hernandez-Herrero and J. A. Solworth, “The need for a multi-perspective approach to solve the DDos problem,” *Bell Labs Technical Journal*, vol. 12, no. 3, pp. 121-130, 2007.

[68] S. A. Shaikh, H. Chivers, P. Nobles, J. A. Clark, and H. Chen, “Towards scalable intrusion detection,” *Network Security*, vol. 2009, no. 6, pp. 12-16, 2009.

[69] N. Ierace, C. Urrutia, and R. Bassett, “Intrusion Prevention Systems,” *Ubiquity*, vol. 6, no. 19, 2006.

[70] D. Barman, J. Chandrashekar, M. Faloutsos, L. Huang, N. Taft, and F. Giroire, “Impact of IT Monoculture on Behavioral End Host Intrusion Detection,” *An ACM SIGCOMM 2009 workshop*, 2009. .

[71] K. Leung and C. Leckie, “Unsupervised anomaly detection in network intrusion detection using clusters,” *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, 2005. .

[72] H. Qiao, J. Peng, C. Feng, and J. W. Rozenblit, “Behavior Analysis-Based Learning Framework for Host Level Intrusion Detection,” *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 2007. .

[73] Y. Chi, “A consumer-centric design approach to develop comprehensive knowledge-based systems for keyword discovery,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 2481-2493, 2009.

[74] K. Kumar, R. Joshi, and K. Singh, “An ISP level Distributed Approach to Detect DDoS Attacks,” 2007, pp. 235-240.

[75] G. Li, C. Li, J. Feng, and L. Zhou, “SAIL: Structure-aware indexing for effective and progressive top-k keyword search over XML documents,” *Information Sciences*, vol. 179, no. 21, pp. 3745-3762, 2009.

[76] Y. Lin, K. Tseng, T. Lee, Y. Lin, C. Hung, and Y. Lai, “A platform-based SoC design and implementation of scalable automaton matching for deep packet inspection,” *Journal of Systems Architecture*, vol. 53, no. 12, pp. 937-950, 2007.

[77] T. N. HINH, S. KITTITORNKUN, and S. TOMIYAMA, “PAMELA: Pattern Matching Engine with Limited-Time Update for NIDS/NIPS,” *IEICE TRANSACTIONS on Information and Systems*, vol. 92, no. 5, pp. pp.1049-1061, 2009.

[78] Robin Sommer, Vern Paxson, and Nicholas Weaver, “An architecture for exploiting multi-core processors to parallelize network intrusion

prevention," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 10, pp. 1255-1279, 2009.

[79] N. Carey, G. Mohay, and A. Clark, "Attack Signature Matching and Discovery in Systems Employing Heterogeneous IDS," *Proceedings of the 19th Annual Computer Security Applications Conference*, 2003. .

[80] W. Li, L. Zhi-tang, and W. Qi-hong, "A novel technique of recognizing multi-stage attack behaviour," *Proceedings of the 2006 International Workshop on Networking, Architecture, and Storages*, 2006. .

[81] Z. Li, A. Zhang, D. Li, and L. Wang, "Discovering Novel Multistage Attack Strategies," *Proceedings of the 3rd international conference on Advanced Data Mining and Applications*, 2007. .

[82] S. Mathew, R. Giomundo, S. Upadhyaya, M. Sudit, and A. Stotz, "Understanding multistage attacks by attack-track based visualization of heterogeneous event streams," *Proceedings of the 3rd international workshop on Visualization for computer security*, 2006. .

[83] S. J. Yang, A. Stotz, J. Holsopple, M. Sudit, and M. Kuhl, "High level information fusion for tracking and projection of multistage cyber attacks," *Inf. Fusion*, vol. 10, no. 1, pp. 107-121, 2009.

[84] W. T. Strayer, C. E. Jones, and B. I. Schwartz, "Architecture for Multi-Stage Network Attack Traceback," *Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*, 2005. .

[85] A. Sargeant, D. Webster, K. Djemame, and J. Xu, "Testing the Effectiveness of Dynamic Binding in Web Services," in *Computer and Information Technology, International Conference on*, vol. 0, pp. 1263-1268, 2010.

[86] J. Arshad, P. Townend, and J. Xu, “Quantification of Security for Compute Intensive Workloads in Clouds,” in *Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems*, pp. 479-486, 2009.

[87] D. J. Russell, L. Liu, Z. Luo, C. C. Venters, D. E. Webster, and J. Xu, “Realizing Network Enabled Capability Through Dependable Dynamic Systems Integration,” in *Computer and Information Technology, International Conference on*, vol. 0, pp. 1269-1274, 2010.

[88] C. Peikari, *Security warrior*, 1st ed. Beijing ;;Sebastopol CA: O'Reilly & Associates Inc., 2004.

[89] J. Beale, *Snort : IDS and IPS toolkit*. Burlington MA: Syngress, 2007.

[90] R. Bragg and M. Rhodes-Ousley, *NETWORK SECURITY*. Strassberg: Mass Market Paperback, McGraw-Hill, 2004.

[91] F. Alserhani, M. Akhlaq, I. U. Awan, J. Mellor, A. J. Cullen, and P. Mirchandani, “Multi-tier Evaluation of Network Intrusion Detection Systems,” *Journal of Information Assurance and Security*, vol. 5, no. 1, pp. 301-310, 2010.

[92] Arbor Network Inc,, “Network Infrastructure Security Research Report | Arbor Networks,” 2009. [Online]. Available: http://www.arbornetworks.com/report. [Accessed: 19-May-2010].

[93] D. McPherson, “Cybercrime - A game of cat and mouse in 2009,” *Network Security*, vol. 2010, no. 2, pp. 15-18, Feb. 2010.

[94] McAfee, “McAfee® - threat_center - McAfee Labs Technical White Papers,” 2009. [Online]. Available: http://www.mcafee.com/us/threat_center/white_paper.html. [Accessed:

19-May-2010].

[95] A. Zeichick, “Reality of botnet cyberwarfare,” *netWorker*, vol. 13, no. 3, pp. 5-9, 2009.

[96] S. M. Furnell and M. J. Warren, “Computer hacking and cyber terrorism: the real threats in the new millennium?,” *Computers & Security*, vol. 18, no. 1, pp. 28-34, 1999.

[97] N. Kshetri, “Pattern of global cyber war and crime: A conceptual framework,” *Journal of International Management*, vol. 11, no. 4, pp. 541-562, Dec. 2005.

[98] B. Baskin and ScienceDirect (Online service), *Combating spyware in the enterprise*. Rockland, MA :: Syngress,, 2006.

[99] SearchSecurityAsia Editors, “Search engine poisoning continues | Security Asia,” 2010. [Online]. Available: http://security.networksasia.net/content/search-engine-poisoning-continues. [Accessed: 19-May-2010].

[100] J. B. Ullrich, “Search Engine Poisoning: Chile Earthquake,” 2010. [Online]. Available: http://isc.sans.org/diary.html?storyid=8317. [Accessed: 19-May-2010].

[101] ICSALabs, “How to select a network IPS,” Verizon, 2010.

[102] M. A. Moya, “Analysis and evaluation of the Snort and Bro network intrusion detection systems,” Intrusion Detection System, Universidad Pontificia Comillas.

[103] Lawrence Berkeley National Laboratory, “Bro Intrusion Detection System - Bro Overview,” 2010. [Online]. Available: http://bro-ids.org/. [Accessed: 05-Nov-2010].

[104] Sourcefire, “Snort :: Community,” 2010. [Online]. Available: http://www.snort.org/community. [Accessed: 05-Nov-2010].

[105] H. Hasenauer, *Sustainable forest management : growth models for Europe*. Berlin ;;London: Springer, 2006.

[106] IUPR Research Lab, “bonesi - Project Hosting on Google Code,” 2008. [Online]. Available: http://code.google.com/p/bonesi/. [Accessed: 13-May-2010].

[107] Computer Security, “Spanish Plane Crash Caused In Part By Malware | Computer Security Articles,” 2010. [Online]. Available: http://www.computersecurityarticles.info/security/spanish-plane-crash-caused-in-part-by-malware/. [Accessed: 14-Nov-2010].

[108] Honeynet Project., “Challenge 2 of the Forensic Challenge 2010 - browsers under attack | The Honeynet Project,” 2010. [Online]. Available: http://www.honeynet.org/challenges/2010_2_browsers_under_attack. [Accessed: 11-Jul-2010].

[109] D. Edward, “/packer/,” 2010. [Online]. Available: http://dean.edwards.name/packer/. [Accessed: 11-Jul-2010].

[110] Snort, “Snort manual 2_8_6,” 2010.

[111] Snort, “Snort :: Additional Downloads,” *Snort*, 2010. [Online]. Available: http://www.snort.org/snort-downloads/additional-downloads#zeroshell. [Accessed: 05-Nov-2010].

[112] F. Knobbe, “About SnortSam,” *SnortSam*, 2010. [Online]. Available: http://www.snortsam.net/. [Accessed: 05-Nov-2010].

[113] Cisco Systems Inc., “SpamCop.net - Beware of cheap imitations,” 2009. [Online]. Available: http://www.spamcop.net/. [Accessed: 17-Jun-2010].

[114] Microsoft Technet, “Microsoft Security Bulletin MS09-059 - Important: Vulnerability in Local Security Authority Subsystem Service Could Allow Denial of Service (975467),” 2009. [Online]. Available: http://www.microsoft.com/technet/security/bulletin/ms09-059.mspx. [Accessed: 01-Jun-2010].

[115] Microsoft, “Securing Your Network with Firewalls and Ports,” 2007. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms960403(CS.70).aspx. [Accessed: 02-Jun-2010].

[116] S. McClure, *Hacking exposed 6 : network security secrets & solutions*, 10th ed. New York: McGraw-Hill, 2009.

[117] C. McNab, *Network security assessment*, 2nd ed. Sebastopol Calif.: O'Reilly Media, 2007.

[118] CISCO, “Security Intelligence Operations - Cisco Systems,” 2007. [Online]. Available: https://tools.cisco.com/security/center/viewIpsSignature.x?signatureId=3308&signatureSubId=0&softwareVersion=6.0&releaseVersion=S268. [Accessed: 02-Jun-2010].

[119] P. Červěn, *Crackproof your software the best ways to protect your software against crackers*. San Francisco :: No Starch Press,, 2002.

[120] P. Villani, *Programming Win32 under the API*. Lawrence Kan.: CMP Books, 2001.

[121] R. Lippmann, *Recent advances in intrusion detection 11th international symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008 : proceedings*. Berlin ;;New York :: Springer,, 2008.

[122] The Honeynet Project, “Scan 28,” 2003. [Online]. Available:

http://old.honeynet.org/scans/scan28/. [Accessed: 11-Jun-2010].

[123] J. Burton and ScienceDirect (Online service), *Cisco security professional's guide to secure intrusion detection systems*. Rockland, MA :: Syngress Pub.,, 2003.

[124] Honeynet Project., *Know your enemy : revealing the security tools, tactics, and motives of the blackhat community*. Boston MA: Addison-Wesley, 2001.

[125] J. V. Braun, “SANS: Intrusion Detection FAQ: What port numbers do well-known trojan horses use?,” 2010. [Online]. Available: http://www.sans.org/security-resources/idfaq/oddports.php. [Accessed: 15-Jun-2010].

[126] M. Brown, *Using Netscape 3*, Special ed. Indianapolis Ind.: Que, 1996.

[127] Ports.My-Addr.com, “tcp port 32784,udp port 32784,udp tcp 32784 description,biggest ports library database,” 2010. [Online]. Available: http://ports.my-addr.com/tcp_port-udp_port-application-and-description.php?port=32784. [Accessed: 15-Jun-2010].

[128] “DNS Poisoning | SANS Internet Storm Center; Cooperative Network Security Community - Internet Security.”

[129] R. Rehman, *Solaris 8 training guide (310-043) : network administrator*. Indianapolis Ind. ;Hemel Hempstead: New Riders ;;Prentice Hall, 2001.

[130] W. Cheswick, *Firewalls and Internet security : repelling the wily hacker*, 2nd ed. Boston: Addison-Wesley, 2003.

[131] P. Hoagland, “The Teredo Protocol: Tunneling Past Network Security and Other Security Implications,” 2007. [Online]. Available: http://www.symantec.com/avcenter/reference/Teredo_Security.pdf.

[Accessed: 20-Jun-2010].

[132] X. Leng, J. Bi, and M. Zhang, “Study on High Performance IPv4/IPv6 Transition and Access Service,” *Parallel and Distributed Processing and Applications*, 2006. [Online]. Available: http://dx.doi.org/10.1007/11946441_21.

[133] H. Langweg and E. Snekkenes, “A classification of malicious software attacks,” in *IEEE International Conference on Performance, Computing, and Communications, 2004*, pp. 827-832.

[134] M. Xiao and D. Xiao, “Alert Verification Based on Attack Classification in Collaborative Intrusion Detection,” in *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, pp. 739-744, 2007.

[135] A. R. Roozbahani, R. Nassiri, and G. Latif-Shabgahi, “Attacks Classification to Improve the Power of Snorts,” in *2009 International Forum on Computer Science-Technology and Applications*, pp. 3-6, 2009.

[136] L. DeLooze, “Classification of computer attacks using a self organizing map,” in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pp. 365-369.

[137] L. DeLooze, “Classification of computer attacks using a self organizing map,” in *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pp. 365-369.

[138] S. Yanhang and Z. Zhou, “Cooperative Control for Target Search, Classification and Attack for AUAVs(Attack Uninhabited Air Vehicles),” in *2007 Chinese Control Conference*, pp. 99-102, 2006.

[139] C. Douligeris and A. Mitrokotsa, “DDoS attacks and defense mechanisms:

a classification," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No.03EX795)*, pp. 190-193.

[140] F. Haddadi, S. Khanchi, M. Shetabi, and V. Derhami, "Intrusion Detection and Attack Classification Using Feed-Forward Neural Network," in *2010 Second International Conference on Computer and Network Technology*, pp. 262-266, 2010.

[141] Zheng Zhang and C. Manikopoulos, "Investigation of neural network classification of computer network attacks," in *International Conference on Information Technology: Research and Education, 2003. Proceedings. ITRE2003.*, pp. 590-594.

[142] Yao Shuping and Gu Yingyan, "Network security situation quantitative evaluation based on the classification of attacks in attack-defense confrontation environment," in *2009 Chinese Control and Decision Conference*, pp. 6014-6019, 2009.

[143] J. Bi, P. Hu, and P. Li, "Study on Classification and Characteristics of Source Address Spoofing Attacks in the Internet," in *2010 Ninth International Conference on Networks*, pp. 226-230, 2010.

[144] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Computer Networks*, vol. 34, no. 4, pp. 579-595, Oct. 2000.

[145] N. Paulauskas and E. Garsva, "Computer System Attack Classification," vol. 2, no. 66, pp. 84-87, 2006.

[146] P. Wright, "Oracle Internals - Ten Stages of a Network Attack - Rootkit installation," 2010. [Online]. Available: http://www.dba-

oracle.com/forensics/t_forensics_network_attack.htm. [Accessed: 07-Nov-2009].

[147] K. Beaver, *Hacking for dummies*, 3rd ed. Hoboken NJ: Wiley Pub., 2010.

[148] B. Schneier, *Secrets and lies : digital security in a networked world*. New York: John Wiley, 2000.

[149] A. Moore, R. Ellison, and R. Linger, “Attack Modeling for Information Security and Survivability,” *Software Engineering Institute | Carnegie University*, 2001. [Online]. Available: http://www.sei.cmu.edu/library/abstracts/reports/01tn001.cfm. [Accessed: 01-Aug-2010].

[150] J. Meier and Microsoft Corporation., *Improving Web application security : threats and countermeasures*. [Redmond Wash.?]: Microsoft, 2003.

[151] J. Mirkovic et al., “Automating DDoS experimentation,” in *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, pp. 4-4, 2007.

[152] StopBadware, “StopBadware - This isn't an attack site... or is it?,” 2010. [Online]. Available: http://www.stopbadware.org/firefox#how_does_firefox_determine. [Accessed: 25-Jan-2010].

[153] Microsoft, “Featured Intelligence,” *Security Inteligence Report*, 2010. [Online]. Available: http://www.microsoft.com/security/sir/story/default.aspx#section_2_3_7. [Accessed: 14-Nov-2010].

[154] S. Young and D. Aitel, *The hacker's handbook : the strategy behind*

breaking into and defending networks. Boca Raton, Fla. ; London: Auerbach, 2004.

[155] S. Malik, *Network security principles and practices*. Indianapolis Ind. ;[Great Britain]: Cisco, 2003.

[156] C. Zou, D. Towsley, Weibo Gong, and S. Cai, “Routing worm: a fast, selective attack worm based on IP address information,” in *Principles of Advanced and Distributed Simulation, 2005. PADS 2005. Workshop on*, pp. 199-206, 2005.

[157] S. Castro, D. Wessels, M. Fomenkov, and K. Claffy, “A day at the root of the internet,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 5, pp. 41-46, 2008.

[158] J. Kang, Y. Zhang, and J. Ju, “Detecting DDoS Attacks Based on Multi-stream Fused HMM in Source-End Network,” in *Cryptology and Network Security*, 2006, pp. 342-353.

[159] J. Chang-Han and S. Shiuh-Pyng, “Detecting Distributed DoS/Scanning by Anomaly Distribution of Packet Fields,” 2006. .

[160] IANA, “IANA IPv4 Address Space Registry,” 2010. [Online]. Available: http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml. [Accessed: 23-Jun-2010].

[161] EmergingThreats, “Emerging Threats,” 2008. [Online]. Available: http://emergingthreats.net/. [Accessed: 05-Aug-2010].

[162] P. Wang, L. Wu, B. Aslam, and C. C. Zou, “A Systematic Study on Peer-to-Peer Botnets.”

[163] W. Strayer, R. Walsh, C. Livadas, and D. Lapsley, “Detecting Botnets with Tight Command and Control,” in *Proceedings. 2006 31st IEEE*

Conference on Local Computer Networks, pp. 195-202, 2006.

[164] R. Walsh, D. Lapsley, and W. T. Strayer, “Effective Flow Filtering for Botnet Search Space Reduction.”

[165] N. Provos and T. Holz, *Virtual honeypots : from botnet tracking to intrusion detection*. Upper Saddle River, NJ: Addison-Wesley, 2008.

[166] Maxmind, Inc, “Geolocation and Online Fraud Prevention from MaxMind,” 2010. [Online]. Available: http://www.maxmind.com/. [Accessed: 05-Aug-2010].

[167] T. Crothers, *CIW master administrator certification kit*. San Francisco Calif. ;;London: SYBEX, 2002.

[168] S. Convery, *Network security architectures*. Indianapolis IN: Cisco Press, 2004.

[169] rfc4380, “rfc4380,” 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4380.txt. [Accessed: 05-Aug-2010].

[170] D. Dittrich, “The "stacheldraht" distributed denial of service attack tool,” 1999. [Online]. Available: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt. [Accessed: 23-Jun-2010].

[171] C. Sun, J. Fan, and B. Liu, “A Robust Scheme to Detect SYN Flooding Attacks,” 2007.

[172] B. Xiao, W. Chen, Y. He, and E. H. -. Sha, “An active detecting method against SYN flooding attack,” *IN: PROC. OF THE 11TH IEEE INT’L CONF. ON PARALLEL AND DISTRIBUTED SYSTEMS. VOL.1*, 2005.

[173] V. A. Siris and F. Papagalou, “Application of anomaly detection algorithms for detecting SYN flooding attacks,” *IN PROCEEDINGS OF IEEE*

GLOBECOM, 2004.

[174] H. W. Danlu, “Detecting SYN Flooding Attacks,” 2002.

[175] H. Wang, D. Zhang, and K. G. Shin, “Detecting SYN Flooding Attacks,” *IN PROCEEDINGS OF THE IEEE INFOCOM*, 2002.

[176] J. Yuan and K. Mills, “Monitoring the macroscopic effect of DDoS flooding attacks,” *IEEE TRANSACTIONS ON SECURE AND DEPENDABLE COMPUTING*, 2004.

[177] H. S. Casner, “RTP: A Transport Protocol for Real-Time Applications.”

[178] S. Ranjan, “DDoS resilient scheduling to counter application layer attacks under imperfect detection,” *IN PROCEEDINGS OF IEEE INFOCOM*, 2006.

[179] Q. Li, H. Goe, B. Xu, and Z. Jiao, “Hardware Threat: The Challenge of Information Security,” presented at the Proceedings : International Symposium on Computer Science and Computational Technology ISCSCT 2008 Shanghai, China, 20-22 December 2008, Los Alamitos Calif., 2008.

[180] K. Higgins, “Permanent Denial-of-Service Attack Sabotages Hardware - Security/Management - DarkReading,” 2008. [Online]. Available: http://www.darkreading.com/security/management/showArticle.jhtml?articleID=211201088. [Accessed: 05-Aug-2009].

[181] S. Chebrolu, A. Abraham, and J. Thomas, “Feature Deduction and Ensemble Design of Intrusion Detection Systems,” *Computers and Security*, vol. 24, no. 4, pp. 295-307, 2005.

[182] M. Gregg, *Hack the stack : using snort and ethereal to master the 8 layers of an insecure network*. Rockland MA: Syngress Pub., 2006.

[183] P. Ferguson and D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing,” 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2827.txt. [Accessed: 15-Jan-2009].

[184] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “BLINC: Multilevel Traffic Classification in the Dark,” *IN PROCEEDINGS OF ACM SIGCOMM*, 2005.

[185] H. Kim, D. Barman, M. Faloutsos, M. Fomenkov, and K. Lee, “Internet Traffic Classification Demystified: The Myths, Caveats and Best Practices,” *IN PROC. ACM CONEXT*, 2008.

[186] S. Zanero, “Analyzing TCP Traffic Patterns Using Self Organizing Maps,” in *Image Analysis and Processing – ICIAP 2005*, vol. 3617, Springer Berlin / Heidelberg, 2005, pp. 83-90.

[187] L7-Filter, “L7-filter Supported Protocols.” [Online]. Available: http://l7-filter.sourceforge.net/protocols. [Accessed: 07-Nov-2010].

[188] DEF CON Communications, Inc.#, “DEF CON® Hacking Conference - The Hacker Community's Foremost Social Network.,” 2009. [Online]. Available: http://www.defcon.org/. [Accessed: 31-Jan-2010].