Amber: A Zero-Interaction Honeypot with Distributed Intelligence

A thesis submitted in partial fullfilment of the
requirements for the degree of

MASTERS OF SCIENCE

of

RHODES UNIVERSITY

by

Adam Schoeman

January 2014

Abstract

For the greater part, security controls are based on the principle of Decision through Detection (DtD). The exception to this is a honeypot, which analyses interactions between a third party and itself, while occupying a piece of unused information space. As honeypots are not located on productive information resources, any interaction with it can be assumed to be non-productive. This allows the honeypot to make decisions based simply on the presence of data, rather than on the behaviour of the data. But due to limited resources in human capital, honeypots' uptake in the South African market has been underwhelming. Amber attempts to change this by offering a zero-interaction security system, which will use the honeypot approach of Decision through Presence (DtP) to generate a blacklist of third parties, which can be passed on to a network enforcer. Empirical testing has proved the usefulness of this alternative and low cost approach in defending networks. The functionality of the system was also extended by installing nodes in different geographical locations, and streaming their detections into the central Amber hive.

Acknowledgements

Security controls have traditionally been based on the principle of Decision through Detection (DtD). The exception to this is a honeypot, which analyses interactions between a third party and itself. As honeypots are not located on production information systems, any interaction with it can be assumed to be likley malicious. This allows the honeypot system to make decisions based simply on the presence of data, rather than on the behaviour charactaristics or contents of the data. But due to limited resources in human capital, honeypots' uptake in the South African market has been underwhelming. Amber prototype system presented in this work, attempts to change this by offering a zero-interaction security system, which reduced the need for human interaction. This approach of Decision through Presence (DtP) is used to generate a blacklist of remote Internet Addresses. These lists can be passed on to a network security enforcement system such as a firewall.. Empirical testing has proved the usefulness of this alternative and low cost approach in defending networks. The functionality of the system was also tested by installing nodes in different geographical locations, and streaming their detections into the central Amber collection system.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

The controls that information security practitioners have at their disposal to protect the information technology assets under their custodianship can be split into two distinct categories. Each of these two categories differ in the way that they implement their specific checks and remediations, which makes it possible to view each category as a different security model.

The first category covers the more common security controls, such as antivirus, firewalls, intrusion detection/prevention systems and email filters. According to Cohen (1989) antivirus software, at its very core, uses signature repositories to compare executed code to known bad snippets of code, labelling them as viruses. Firewalls, as noted by Ioannidis, Keromytis, Bellovin, and Smith (2000) traditionally define what ports and services are allowed to access a network node and block everything else. Intrusion Prevention Systems (IPS) use the same principle as antivirus, inspecting network traffic and attempting to match known bad sequences to incoming traffic as detailed by Heady, Luger, Maccabe, and Servilla (1990). Michelakis, Androutsopoulos, Paliouras, Sakkis, and Stamatopoulos (2004) describes email filters as making a decision on whether an email is spam based on numerous factors such as destination, source and content. They all share the trait that they implement their information security enhancements by first analysing some information that they are presented with. Following the analysis of the information stream, the stream is given a rating which can vary in degrees from 'good' to 'bad'. Armed with a rating, Oberheide, Cooke, and Jahanian (2008) describes how the security control is able to make a decision on how to handle the stream. This process of decision-making can be referred to as Decision through Detection (DtD), because it requires the information security control to base its decision on the behaviour of the information flow that is presented. If the behaviour matches a schematic that is seen as being malicious, the security control is triggered. The process of building a behavioural database is intensive and prone to both false positives and negatives, as detailed by Cohen (1989) in Models of Practical Defenses Against Computer Viruses.

The second category of security controls is one that is seen far less frequently than its detection counterpart, but is still employed by honeypots. A honeypot is an information security system that has no productive business use, because it occupies non-production IP address space, such as an unused IP address. Based on this Spitzner (2002); Provos (2003) theorise that any interaction with the honeypot is automatically suspicious, regardless of the behavioural traits of the interaction. By not needing to take into consideration the behaviour of the analysed data flows, honeypots use a different decision system based on the presence of information and not the behaviour. This is referred to as Detection through Presence (DtP). Decision through Presence (DtP) is unique to honeypots by virtue of its links to the surrounding information resources. The honeypot's security model uses a simplistic method of classifying information streams as possibly malicious, but because there is no reason for it to intervene against the information stream, it cannot action the intelligence that it gathers in the same way as the security model used by firewalls or antivirus security controls. Information streams that a honeypot captures can be offloaded for further investigation and used as a basis of discovering new attack vectors and methods.

While each of these two security models are distinctly different, they can both be split into two phases: Discovery and Action. The firewall security model relies on signatures to discover a malicious information stream, and then takes action against it by blocking it or alerting its presence. The honeypot security model discovers non-productive information streams by occupying non-productive space, and based on that is able to pass the information stream to begin further analysis. Each of the security models has an advantage in a different phase; an advantage that manifests as efficiency (in terms of effort) compared to the other security model in the same phase. DtP is efficient at identifying malicious sources, while DtD is efficient at actioning intelligence.

1.2 Research Question and Goals

This research attempts to answer the question of whether or not a hybrid security control is able to improve the information security posture of a network by denying malicious sources access to network assets. Such a security control is a hybrid because it is based on a combination of the most efficient phases of the DtD and DtP security models. The goal

is to build a physical security control that adheres to the principles of this hybrid security model; a model that uses the Discovery phase of the DtP model and the Action phase of the DtD model. The model will then be empirically tested by deploying the physical security control. Once the security control has been tested and the model verified, the possibility of extending its usefulness will be explored by expanding the model into multiple nodes. The nodes will pool their findings with the aim of further improving the security posture of the network by increasing the internet surface area that they cover.

The research also tries to expand on the potential usefulness of the system by distributing the nodes to different geographical regions. The intention is to increase the internet surface area that the control is attached to, and to diversify the sources of information. The data sets gathered from these distributed nodes will also be analysed and may discredit the use of source IP address-based blacklists that distribute their findings outside the region in which they were collected. There may be, however, some value found in attributing the source IP addresses into regions, and then constructing analysis systems based on the cumulative results per country.

1.3 Limitations and Scope

Certain limitations did impact this research, the most notable being a lack of IPv6 testing. The security control that was built to act as an empirical testing tool for the new security model was not able to be placed on networks that generated a sufficient amount of production IPv6 traffic to accurately draw any conclusion. The testing was also limited to only TCP/IP traffic (no ICMP or UDP traffic) because the connection state built into the protocol was needed to prevent possible corruption caused by traffic spoofing. The scope of this research is thus limited to IPv4.

The system also needs to be relevant to enterprise deployment, meaning that the scope would exclude enhancements that were unrelated or unrealistic in terms of a typical enterprise network.

1.4 Organisation of Thesis

The remainder of this document is structured as follows:

- Chapter 2 discusses the literature survey that was undertaken in the field of honeypots and how they differ from more traditional security controls, and the security models that dictate their construction. It introduces the concepts that are discussed throughout this thesis and serves as an introduction to the idea of a honeypot.
- Chapter 3 deconstructs the security models that influence the workings of some of the traditional security controls and honeypots, and proposes a formula that can be used to cost the security model's two phases. With this information a new security model is put forward and a technical specification to conform to it is designed, and which will become the basis of testing the new security model.
- Chapter 4 analyses the data that was collected by Amber, the software system defined in chapter 3 and details the findings as well as certain strategies that can be used to improve signature-based controls with the type of data that was collected.
- Chapter 5 outlines enhancements that would benefit future deployments of the Amber ecosystem but that were out of the scope for the research goals presented here. The research is also concluded in this chapter.

2 Background and Related Concepts

At the core of this research is the unique way in which a honeypot is able to contribute to the information security posture of a network, but there are many differences within the world of honeypots. This chapter serves as an introduction to the terms and concepts that frequent the field of honeypots as well as a discussion on the security models that group together traditional signature-based security controls and honeypots. The remainder of the chapter will explore the following topics:

Section 2.1 looks at the history of honeypots and how the concept has evolved over the years.

Section 2.2 expands on how honeypots are classified, in terms of how much functionality they are able to expose to potential attackers, and the honeypots' deployment goal.

Section 2.3 explains the differences in honeypot deployments, be it server or client side.

Section 2.4 takes a look at some of the less traditional honeypots and elaborates on what aspects make them unique.

Section 2.5 lists the potential advantages of deploying a honeypot.

Section 2.6 lists the potential disadvantages of deploying a honeypot

Section 2.7 tackles the problem of exposing the environment to certain legal and ethical liabilities.

Section 2.8 formally defines the two security models that make up signature-based security controls and honeypots.

Section 2.9 serves as a summary for the chapter.

2.1 History of Honeypots

Antivirus, firewalls, intrusion detection/prevention systems and email filters all share a common trait in that they implement their information security enhancements by analysing the information that they are presented with.

According to Cohen (1989) antivirus software uses signature repositories to compare executed code to known bad snippets of code, labelling them as viruses. Firewalls, as noted by Ioannidis et al. (2000) traditionally define what ports and services are allowed to access a network node and block everything else. Intrusion Prevention Systems (IPS) use the same principle as antivirus, inspecting network traffic and attempt to match known bad sequences to incoming traffic as detailed by Heady et al. (1990). Michelakis et al. (2004) describes email filters as making a decision on whether an email is spam based on numerous factors such as destination, source and content.

Each one of the above basic information security controls has a list of predefined patterns or actions that have been classified as malicious, and when they detect them they take action based on the severity of the identified threat, a process that is detailed by Oberheide et al. (2008). The process of building a behavioural database is intensive and prone to both false positives and negatives, as explored by Cohen (1989).

A honeypot, however, is an information security system that has no productive business use, because it occupies non-production IP address space, such as an unused IP address. Based on this Spitzner (2002); Provos (2003) theorise that any interaction with the honeypot is automatically suspicious, regardless of the behavioural traits of the interaction.

One of the earliest examples of a honeypot was a chroot jail that was built on a production system by Cheswick (1992) who worked at AT&T Laboratories. In 1991, a cracker attempted to use a well known sendmail DEBUG vulnerability to download a copy of the server's passwd file (a file that contains usernames and encrypted passwords) for a server that Cheswick administered (Cheswick, 1992). The server had been set up previously to expose a set of services that were not connected to production functions and which logged all commands that they processed. The log files could be analysed later, and, in this early case, required Cheswick to maintain the interaction with the cracker by manually intervening and emailing the cracker a bogus passwd file. After many months of interactions, Cheswick was able to learn a great deal about the cracker, enough to be able to recognise his or her typing style when logged into other servers on the network. This kind of engagement also led to these concepts of luring malicious parties onto disposable systems being introduced into popular fiction, by Stoll (1995) in his book The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage.

Network Associate's Cybercop Sting (PR Newswire, 1998), released 14 July 1998, is the earliest version of a commercial honeypot, originally referred to as a decoy system. It offered consumers a way to audit and monitor intruders before legitimate data and data systems were at risk. Prior to that, Cohen (2012) at Fred Cohen & Associates released a package called The Deception Toolkit. It provided an open source alternative to the commercially licensed Cybercop Sting and enabled the open source community to experiment both with the idea of allowing intruders to compromise assets and with using those interactions as a way of learning how to defend against them. Cohen's toolkit formed a basis from which the community could start actively researching the field of honeypots as his system was a collection of Perl scripts. One such researcher was Lance Spitzner (Spitzner, 1999), founder of the Honeynet Project, and author of Honeypots: Tracking Hackers.

In his book, Spitzner (2002) outlines the foundation for modern honeypots. It contains real world examples of why they are useful, as well as the core concepts of honeypots: occupying unused and unproductive internet space, with the intent of luring and learning about the actions of malicious attackers. In his paper, To Build a Honeypot, Spitzner (1999) details the basics of building a server that can be used to record the steps and actions taken by a malicious intruder. The server (or honeypot) would be separated from the valued network so that it could safely sustain damage and report back on how that damage was inflicted. To Build a Honeypot led Spitzner and co-author Niels Provos to create a novel and simple way of deploying a honeypot, namely through Honeyd (Provos, 2003), a virtual honeypot daemon. Honeyd can simulate a TCP stack so as to further fool would be attackers, and can be configured with a number of templates which mirror certain stereotypical production servers. Spitzner and Provos' research on the potential of harnessing unused space on networks as an additional method of monitoring unidentified intruders laid the foundation for numerous future works on the subject, due to the authors' decision to build open source systems.

2.2 Honeypot Classifications

Honeypots can be classified according to their goals and the manner in which they attempt to collect information. At the highest level, honeypot goals can be seen as being either

detection driven or prevention driven. A honeypot can be further classified as a low, medium or high interaction honeypot, based on how it collects and controls data, which is another core function. Zhang, Zhou, Qin, and Liu (2003) noted that a honeypot has two basic functions: data control and data capture. Data control relates to the honeypot's ability to expose a service or port that a potential attack can connect to, while data capture relates to the recording and archiving of inbound and outbound data, for the purpose of studying the full interaction.

2.2.1 Prevention Honeypots

Prevention honeypots attempt to prevent an attacker from compromising production systems, by luring them to focus on the honeypot instead of the production systems. While valuable, they are not well suited to automated worms because these grades of attacks do not incur a time cost, as explored by Zhang et al. (2003). This is not to say that prevention honeypots cannot be used against worms, as shown by La Brea (Chen, Gao, and Kwiat, 2003) and the Code Red worm, a worm that spread with incredible speed as noted by Moore, Shannon, et al. (2002);Chen et al. (2003). In this case, the prevention honeypot was designed to slow down the spread of Code Red by filling unused IP addresses on the network with services that incrementally fed the worm's attacks.

2.2.2 Detection Honeypots

Detection honeypots are similar to intrusion detection systems (IDS) in that they both issue an alert when an attack occurs, but while IDS systems rely on signatures to trigger an alert, a detection honeypot generates alerts based on system activities. Research done by Kuwatly, Sraj, and Masri (2004) shows that this subtle nuance enables detection honeypots to detect unknown attacks, or attacks which have no signature. These honeypots are deployed as a method to augment the creation of IDS signatures, as shown by Kreibich and Crowcroft (2004) in the form of the Honeycomb project.

2.2.3 High Interaction Honeypots

Honeypots can be further split into high and low interaction honeypots as originally stated by Provos and later refined by Alata, Nicomette, Kaâniche, Dacier, and Herrb (2006). Both high and low interaction honeypots share a trait in that they both expose systems with services in an effort to entice an attacker. The difference, however, lies in the fact that, in a high interaction honeypot, the systems are fully functional and the exposed services are real, whereas in a low interaction honeypot, the above are both fake. This means that any attacker who connects to a high interaction honeypot will be faced with a complete service and will be able to further his or her attack beyond just establishing a connection. The increased interaction between the attacker and the honeypot results in an increase in the quantity and quality of information that the honeypot is able to record, compared to a low interaction honeypot. An example of such a honeypot is Sebek and its evolution Qebek, as explored by Alata et al. (2006);Balas and Viecco (2005); McCarty (2003), both of which aim to capture as much information as possible regarding the connections that access it.

2.2.4 Low Interaction Honeypots

Baecher, Koetter, Holz, Dornseif, and Freiling (2006) states that low interaction honeypots still occupy an unused asset and expose services which are enticing to an attacker (either because they are known as being vulnerable, or are susceptible to authentication brute force attacks) and log the data about the initial communication. The services that the low interaction honeypot exposes are not complete or fully functional and only serve to entice an attacker to establish a connection. The honeypot La Brea, as mentioned earlier, is a low interaction honeypot. It does not mimic functional copies of services and its only goal is to keep the connection with the worm open for as long as possible.

Based on the work by Alata et al. (2006) it would seem that high interaction honeypots are superior to low interaction variants. However, while it is true that there is the potential for high quality information gathering from high interaction honeypots, this is offset against the fact that the honeypot itself needs to be more complex and the information gathered requires more in-depth analysis. Certain honeypots do not need this level of data

interrogation to achieve their goals. Dionaea, a newer version of the work originally done by Baecher et al. (2006), was written by PhiBo (2013) and is one such low interaction honeypot. It attempts to capture a copy of a malware payload that is being launched against a vulnerable emulated service hosted by Dionaea. The honeypot is only concerned with gaining a copy of the payload, so it is able to use emulated services to entice an attacker to launch the payload. An emulated service will not be able to function beyond the payload being launched against it, but as the payload is already captured at this stage, the honeypot does not benefit from any increased complexity or an increase in the life of any service interaction.

2.2.5 Medium Interaction Honeypots

Sitting in between high and low interaction are the so-called medium interaction honeypots. As outlined in work done by Wicherski (2006), these types of honeypots contain aspects of high interaction types in that they do interact with payloads that are sent to the services that are being emulated, but not to the degree of a complete service. The medium interaction honeypot uses a virtual layer to emulate the services. Medium interaction honeypots have been successful in collecting botnet malware that is downloaded after a machine is compromised. As Wicherski notes, the services of a medium interaction honeypot are able to mimic a vulnerable system, accept the vulnerability, analyse it and retrieve the botnet malware. An example of this is Nepenthes, which is a malware collection agent developed by Baecher et al. (2006), and which is able to following staging payloads and download complete botnet malware.

2.3 Honeypot Deployments: Server and Client-side

Traditionally honeypots were deployed as server side applications that waited for incoming connections. With an increase in the prevalence of client-side attacks in the form of drive-by-download (Cova, Kruegel, and Vigna, 2010; Rieck, Krueger, and Dewald, 2010; Egele, Kirda, and Kruegel, 2009) and watering-hole attacks (Net Security, 2013) both of which rely on users connecting to a server that is hosting malicious content, honeypot deployment

strategies have also expanded to include client-side honeypots. Client-side honeypots were first proposed by Spitzner (2002) and later employed by Göbel and Dewald (2011) as a way of identifying malicious servers that were trying to exploit vulnerabilities on connected clients. As stated by Spitzner (2002) client-side honeypots differ from server-side honeypots in that they are installed on a machine that acts as a client, and are configured in such a way to lure malicious servers into attacking them. This breed of honeypot crawls the internet looking for malicious servers by analysing the interactions between the honeypot and the server, flagging interactions that the honeypot sees as being malicious. The Honeynet Project's HoneyC (Seifert, Welch, Komisarczuk, et al., 2007) and more recently work by Nazario (2009) in the form of a virtual client-side honeypot PhoneyC, are both examples of client-side honeypots. As Seifert, Welch, Komisarczuk, et al. (2007) details, the strength of a client-size honeypot is that it has the ability to collect information from servers hosting malware, and how they are targeting clients.

2.4 Other Honeypot Systems

There are some honeypots that fulfil the basic requirements of a dictionary definition, but which have goals that are not aligned to those of traditional honeypots. Others collect and control data in a manner that is significantly different to that of the honeypots listed in sections 2.2 and 2.3. Some of these are discussed below.

2.4.1 Honeytokens

The term honeytoken was coined by de Barros (2003) but was popularised by Spitzner (2003) in his paper Honeypots: Catching the Insider Threat. He defined a honeytoken as "a digital or information system resource whose value lies in the unauthorized use of that resource". A honeytoken is therefore any digital information that has no real value, but which might be seen as having value to a malicious entity. An example would be an administrative username and password to a server that does not exist or bogus credit card information. The lack of utility means that at no point should any entity in a network be referencing or using the information, and if it is, it can be deduced that it is doing so

for nefarious reasons. An example of a deployed honeytoken would be a table in a SQL database that is named `director_salaries` and which is populated with dummy names and figures. Access to that table could indicate a user that is trawling for information and could be someone that is a potential insider risk. Another example of a honeytoken is a fake credit card number that is injected into a database of legitimate credit card numbers. If a payment processing system detects that the fake credit card number is being used, it can be assumed that there is a high possibility that the database has been compromised.

2.4.2 Artillery

Artillery[1] is another pseudo-honeypot that deviates from the traditional honeypot goal structure in that it focuses on instant-prevention. It was created by Dave Kennedy (Kennedy, 2012) and, unlike a traditional honeypot, it is installed as an agent on a productive server (namely, a server with business utility). The Artillery agent will then open a group of ports that the server is not using and wait for connections on those ports. If a connection is detected, all future attempts from that IP address to that service are automatically blocked by Artillery. It is this automated defensive action that distinguishes Artillery from other honeypots.

2.5 Advantages of Honeypot Deployments

Honeypots function in a distinctly different manner to other security controls, which leads them to have a certain set of unique advantages. If one is deployed as a detection honeypot (opposed to a prevention one), then the insight that the honeypot is able to produce is different to that of firewalls and antivirus systems. A detection honeypot can intercept copies of malware that have not yet been identified as malicious by signature-based systems, and can then safely distribute copies to those systems. This increases the speed at which those systems can update their definitions (Kuwatly et al., 2004). They also allow researchers to gain copies of worms, which was used by Moore et al. (2002) to dissect source code in order to learn what methods worms use to spread. Honeypots are

[1] https://www.trustedsec.com/downloads/artillery/

also relatively cheap to deploy because, traditionally, they do not need to be geared to function at multiple gigabits of throughput. Since the internet space that they occupy is not used for any productive purpose, traffic interacting with the honeypot is limited to the non-productive variety and is a fraction of the traffic that an inline firewall would face (Akkaya and Thalgott, 2012).

There are also a certain grade of attacks that simply do not affect honeypots, specifically denial of service attacks (Needham, 1993), distributed or not. Inline devices are limited by the amount of data they can inspect based on the hardware they are built on and the technology available. Firewalls and IDS applications are limited to the amount of simultaneous connections that they can process, a trait that has left them susceptible to SYN flooding. SYN flooding is a process where an attack fills up all available SYN connections, which leaves the firewall unable to process any further connections as detailed by Schuba et al. (1997). A firewall will traditionally fail-close in such a scenario, thus blocking all connections, but according to Garfinkel, Pfaff, Chow, Rosenblum, and Boneh (2003); Iheagwara, Awan, Acar, and Miller (2006); Sheth and Thakker (2013); Yang, Song, and Shen (2001) there are some IDS systems that will fail-open in such a scenario, allowing all connections to effectively bypass it. This sort of attack will not bypass a honeypot because it does not need to fail-open to allow traffic to pass through, because it is not an inline device. A denial of service attack against a honeypot will more than likely alert the network security administrators that the network is under attack.

Honeypots also require very little configuration to deploy because they do not require any research-driven signatures to add value. Projects such as Honeywall CDROM (Chamales, 2004) allow users to boot a fully functional honeypot ecosystem requiring only basic network configuration. Signature-based security controls need only be customised for the network on which they are deployed. According to Huang, Jasper, and Wicks (1999) certain signatures may need to be disabled or modified on intrusion detection systems to minimise the occurrence of false positives, while firewalls may require a ruleset to be written for the specific network that they are deployed on (Chapman and Zwicky, 1995). Antivirus systems may also require an updated signature file to be effective, which is often an additional service available from the majority of antivirus vendors for a fee (Gashi, Stankovic, Leita, and Thonnard, 2009).

2.6 Disadvantages of Honeypot Deployments

A honeypot relies on its ability to convince a potential attacker that it is a legitimate system or network or that it contains valuable information. This deception becomes a critical factor and can become a disadvantage of the honeypot. If an attacker is able to learn (or suspect) that a certain system is a honeypot, he or she could use the system against the network that he or she wants to attack. Krawetz (2004) shows that anti-honeypot technologies exist, and that there are already ways to fingerprint systems running Honeyd. If an attacker is able to identify that he or she is talking to a honeypot, they could input incorrectly crafted information to misdirect those analysing the honeypot data. Krawetz identifies using machines that were compromised earlier to launch attacks, which would obfuscate the real location of attacks. Research has also been conducted by Zou and Cunningham (2006) around botnets that are honeypot aware, and that will not deploy their final payload to a machine that is identified as a honeypot. While this does not directly harm the network, it does negate the potential value that a honeypot can derive.

As mentioned in section 2.5 (Advantages of Honeypot Deployments), honeypots are resilient to network-level denial of service attacks because they are not designed to pass traffic. While this is true, they are vulnerable to analysis manipulation or back-office denial of service attacks. If an attacker has identified a system as a honeypot that collects malware payloads, which are presumably stored for later analysis, he or she could automatically flood the honeypot with thousands of mutating malware. This will put pressure on the back-office malware analysis, which corrupts any legitimate or viable threats that the honeypot may have recorded (Prasad, Reddy, and Karthik, 2012).

2.7 Liability and Ethical Concerns

Honeypots do introduce the potential for liability to the owner if not properly constructed. Since a honeypot is a vulnerable system, it is possible for an attacker to take control of it and start to use it as a stage machine to launch further attacks on external networks, particularly if the honeypot is able to communicate to the greater internet. In such a scenario the honeypot owner might be liable for the damage caused by the malware

that has infected the honeypot, according to Warren and Hutchinson (2007). It is for this reason that Provos (2003) outlined a mitigation architecture that would prevent a honeypot from communicating outside of its network. Unfortunately, some botnet variants have started to use this limitation as a way of preventing honeypots from capturing a copy of the botnet payload. Zou and Cunningham (2006) discuss a form of botnet that uses an external verification module to check whether a system is a honeypot or not. Before sending the final payload, the system is instructed to connect to a listening sensor that reports back to the botnet if the system successfully establishes the connection, at which point the botnet will send the final payload. The logic is that if a honeypot is being targeted, it would be unable to connect to a remote server, and the botnet would then not send (and thereby expose) the final payload. However, a honeynet could be configured to allow the honeypot external access for a short period of time after a connection, so as not to diminish the honeypot's ability to gather this class of botnet malware, but doing so would open the honeypot up to potential liability.

In the paper by Warren and Hutchinson (2007), the potential ethical concerns of using deception is also explored, mostly around the concept of whether deception is equitable to entrapment. Entrapment is defined by Williams (1957) as "a defense to a criminal act when a person is incited, induced, inveigled, or lured into the commision of a crime not contemplated by him, for the purpose of prosecuting him, by a law enforcement officer or his agent" which is very similar to how a honeypot presents itself. It is unlikely that the purpose of deploying a honeypot is to prosecute someone, but because honeypots are traditionally presented as extremely vulnerable targets, it could be viewed as unrealistic and that it amounts to luring an attacker. Some, such as Scottberg, Yurcik, and Doss (2002) argue that the information that is gathered by honeypots is invaluable because of the honeypots' ability to collect previously unknown samples of malware and attack techniques, and thus should be used regardless of the above concerns.

2.8 Security Models

Throughout this section, a distinction has been made between those security controls that use signature databases to detect known malicious activities, and those that occupy unused internet resources. This creates the opportunity to group these different controls

under separate security models, which allows for some degree of generalisation when dealing with a control in a certain security model. Grouping the controls into broader models also allows for a comparison based on a set of common criteria, as long as the models share a similar workflow. A comparison would highlight the specific nuances between the controls and the models on which they are derived.

2.8.1 Decision through Detection (DtD)

As mentioned earlier, antivirus, firewalls, IDS and email filters all implement their form of security through the use of snippets of known malicious code. This process of decision-making can be referred to as Decision through Detection (DtD), because it requires the information security control to base its decision on the behaviour of the information flow that it is presented. If the behaviour matches a schematic that is seen as being malicious, the security control is triggered. Triggering, in this case, relates to how the control takes action based on what it discovers.

Thus, there are two distinct phases to the DtD security model: a discovery phase (where the detection takes place) and the action phase (where, based on the detection of the previous phase, a decision is taken and appropriate action is implemented). A security control that is governed by the DtD model attempts to fulfil the discovery phase by matching information that it is presented with against a list of known malicious artifacts, such as an antivirus signature database or a firewall rule set (Cohen, 1989). The action phase of the DtD security model pivots off the findings of the discovery phase, and it is in this second phase that the control is given the opportunity to intervene on the flow of information based on how malicious the detection is (Oberheide et al., 2008). The key here is that the discovery phase feeds the action phase because the actions that a DtD security control can take are all dependent on an accurate detection in the Discovery phase.

2.8.2 Decision through Presence (DtP)

In contrast to the DtD security model, a honeypot provides its improved security posture by occupying information systems that have no productive business use. Using the same

two-phase approach as detailed in the DtD model, a honeypot achieves the discovery phase simply by waiting for any interaction on the unproductive information space that it is occupying. According to Spitzner (2002); Provos (2003) anything that communicates with it is automatically suspicious, regardless of the behavioural traits of the interaction as nothing of business consequence should be communicating with the honeypot. In the following action phase, the honeypot provides the captured information for further analysis. This can take the form of malware binary analysis or profiling the type of attacks that an attacker has used. By not needing to take into consideration the behaviour of the analysed data flows, honeypots use a different decision system based on the presence of information and not the behaviour. Decision through Presence (DtP) is unique to honeypots by virtue of its links to the surrounding information resources.

2.9 Summary

Honeypots have proven themselves as a novel security control mostly through their ability to further research initiatives. While some honeypots are more aggressive than others, they generally adhere to one common principle: occupying unused space for reasons that are not productive in terms of business functionality. This is in contrast to antivirus, IPS and other signature-based security controls which traditionally inspect data for known malicious patterns. A proposed security model was devised for the differing groups of security controls, models which outline the advantages and disadvantage of the security controls. The next chapter will further explore these security models, particularly the constructing and costing of each of the models, and a potential method of combining parts of the models into a hybrid model.

3 A Hybrid Security Model

The way in which honeypots implement their flavour of security is unique but research expensive. This chapter identifies the specific costs that are required to build a successful honeypot and how that compares to antivirus and firewall technologies. Through this, a discovery is made that would allow for the construction of a honeypot that is still functional but without the high research cost. The deployment was named Amber because it resembles a honeypot, but is less supple. The rest of the chapter is outlined as follows:

Section 3.1 provides an in-depth look at the pieces that are used to build the two different security models and offers a formula that estimates the pricing of each phase and model.

Section 3.2 introduces the idea of a hybrid security model and a conceptual overview of how a potential technical solution could be built.

Section 3.3 offers a detailed design specification and a technical explanation of the pieces that are needed to build the core Amber system as well as the ancillary systems.

Section 3.4 analyses a proposed method of increasing the functionality of Amber by adding additional distributed nodes.

Section 3.5 serves as a summary for the chapter.

3.1 Decision through Detection and Decision through Presence

Chapter 2 introduced the concept of two security models which cover signature-driven security controls, such as antivirus and IDS/IPS, as well as presence-driven honeypots. While different, the security modes so share a common structure in that they are made up of two phases, a Discovery phase and an Action phase.

3.1.1 Discovery Phase

The basic principle of this research is to construct a system that is able to combine the most useful pieces of the two security models, namely: Decision through Detection (DtD),

and Decision through Presence (DtP). While the drawbacks of antivirus and firewall's DtD security model have been mentioned previously in chapter 1 (Cohen, 1989), their advantages have outweighed their disadvantages, which is testament to their success over the years as shown by antivirus vendors such as McAfee (Gallagher, 2010).

While it does require a great deal of human capital and infrastructure to build signatures, pattern files and a detection engine to run them (Dickinson, 2005), the model has the advantage of scaling very well as more nodes are added. Any work or investment that is done on one of the nodes can also be applied to another similar nodes, thereby reducing the total cost of research as more nodes are added. The only cost associated with an increase in nodes is with regards to the cost of delivery. This incorporates the cost to deliver the signature, or threat intelligence to the node, which is used to drive the detection. This has become relatively small with the cost and speed of data transfer over the internet as noted by Ma et al. (2009). Therefore the cost of detection is the total cost of research that was required to generate a signature file that can be passed onto a node using the DtD security model to detect threats. Captured in the total research cost is the human and capital investment of identifying new threats, analysing them, building a way for the security control to detect the threat and applying measures of quality control. Formula 3.1, 3.2, 3.3 and 3.4 express both phases of the DtD and DtP security models.

$$Cost : Detect = (TCoR/n) + (DC * n) \tag{3.1}$$

Where:

$TCoR$	is Total Cost of Research
n	is the number of nodes
DC	is the cost of distribution

As Formula 3.1 shows, the Cost of Detection for a security control which uses the DtD security model is the Total Cost of Research (TCoR) divided by the amount of nodes that the research is aiming to cover. Added to this is the cost of distributing the detection research to each individual node or security control.

This cost can become large as more security controls are added to what the research is attempting to cover, or if a certain degree of critical mass is required for the product to be useful. Antivirus, for example, is only effective if the signature file covers enough malware to offer a workstation or desktop that has the security control installed a high

probability of not being infected. The amount of work that is required to fulfil this is significant G-Data, a European antivirus vendor, reported that 1 017 208 new malware programs had been captured and incorporated into their signature file by the first half of 2010 (Willems, 2010). Supporting this scale of work is expensive in terms of human capital alone, as security firm McAfee's Annual Report for 2010 stated that the company employed in excess of 6 900 people that past year (McAfee, 2010).

In order for a detection-based security control to gain critical mass in terms of costs, the amount of nodes (n) that rely on the research to fulfil their function needs to be increased. Increasing the nodes amortises the Total Cost of Research over each node, reducing the cost of each node. It is this scaling that has facilitated the commoditization of DtD products, and is the reason for their commercial success (Gallagher, 2010), but it is also a limitation on the kind of security control that can be deployed using the DtD mode. The large expense of research can be a barrier to implementing a security control based on this model if the amount of nodes that would use the research is not sufficient to justify the Total Cost of Research. Distributing the cost of research amongst all the users of the DtD security model also means that each individual node receives a certain degree of improvement to its security posture, without having to assign research and development resources to it.

In contrast to this, systems that make use of the DtP security model have a research cost of zero, because the evaluation is simply that of existence, which doesn't require a complex detection signature.

$$Cost : Presence = if(I) \tag{3.2}$$

Where:
I is Information

As Formula 3.2 shows, the costs of Presence is defined as if there is information (I) present on a predetermined medium of communication. Unlike DtD's Detection cost, there is no research cost tied to find the presence of information, to the point where the Cost of Presence is near zero. This is true as long as the security control that is implementing the DtP model is the only system that is occupying an accessible information technology asset. If the security control is the only system using an information asset, there is no need

to try and detect what is a productive communication and what is possibly malicious. Presence becomes a boolean measurement, in that presence is either present or absence.

An example would be a file server that hosts documents on a network that is frequently accessed by a business user for legitimate purposes, and on which there is a remote code vulnerability in the file sharing protocol that causes a buffer overflow. The buffer overflow is triggered if an attacker sends a specifically crafted query to the file server. To prevent this from happening, a host-based intrusion detection system has been installed on the server in the form of an agent. The agent monitors all the queries that come to the file server, and attempts to match them to a list of known malicious queries. The security control is cohabitating the information asset (the IP address of the file server) with a productive asset (the file server itself) and the security control needs to detect what of the information that is being sent to the file server is malicious and what is legitimate. If the host intrusion detection system was installed on a system that had no productive use (such as the file server) then it would not need to discern between productive and possible malicious traffic because none of the traffic would be to a productive asset. In this example the cost of detecting malicious traffic is near zero because the detection criteria is whether there is any traffic. While the cost of configuring presence is zero, deploying a security control based on the DtP model requires the control to exist in isolation from other productive information technology resources - a contrast to that of detection-based systems.

As far as the Discovery phase goes, controls based on the DtP security model are less costly compared to security controls based on the the DtD security model, because there is no research cost involved with deploying a DtP system.

3.1.2 Action Phase

DtP loses its cost advantage in the second phase, named the Action phase. This is because the associated Discovery phase is based only on whether something has a presence, and not the nature of that presence. It is unknown at this stage, in a DtP model, if the information that is being seen is benign or malicious. This is in contrast to the DtD model which has the ability to define how malicious or benign the information is that is

being presented to it in the Discovery phase. DtD Discovery is based on known malicious code which is discovered through prior research. DtP security controls do not have that level of information in the Discovery phase, so the burden then falls on the Action phase to determine how to proceed with the data.

In the case of honeypots, the system owner would deploy the honeypot to gather information on possible new attacks, and use it as a research basis to develop a counter-attack or to modify existing defences (Bernardo, João, Anderson, Nascimento, Amaral, and Timóteo de Sousa Júnior, 2011). The value of this kind of action is that the output is more significant to the organisation housing the honeypot because it is an information stream that is actively targeting their systems, instead of relying on what threat intelligence has been pushed to the organisation via antivirus signature files and intrusion prevention system signature updates. Threat intelligence that is gathered from the honeypots relates directly to possible attacks that are actively targeting the organisation, because the information was collected on the organization's network.

It can be argued that the information gathered in this way is of a higher quality because its relation to the organisation is much more direct compared to a signature that was generated for multiple environments. But by reducing the amount of parties that can consume the threat intelligence, the total cost of generating that threat intelligence increases, because there are less parties over which the total costs can be divided. The result is a DtP cost equation that can be very costly.

$$Cost : Action = (I * threshold) * RCperI * n \tag{3.3}$$

Where:

I	is the amount of incidents collected
$threshold$	is the average incident analysis rate
$RCperI$	is the research cost per incident
n	is the number of nodes

Research by Huffaker, Plummer, Moore, and Claffy (2002); Yegneswaran, Barford, and Ullrich (2003) shows that a threshold needs to be decided on, which will either mark information that needs to be researched further, or information that will be discarded as internet noise. The threshold is a rolled-up number that equals the average amount of

information (I) that is marked for further research, as a percentage of the total amount of data that the DtP security model captures. If on average, the DtP security model passes 6% of the data that it captures for further research, then threshold number will be expressed as 0.06. The higher the threshold the more data is passed for further research, acting as a filter that removes information that does not impact the Action phase costing.

This number is then multiplied by how much it would cost to fully research a piece of information that required additional investigation. This cost includes the time and human capital expended as well as information technology assets such as workstations and bandwidth. The resulting value is the cost of researching all potential useful information from a single DtP security model node, which then needs to be multiplied by the number of nodes. Multiplying by every node that is collecting data is done to reach a value that is comparable to the total research cost expressed in the DtD model, where the ability to scale across a near infinite amount of nodes is an intrinsic advantage.

A DtD security model's Action phase benefits from the higher cost of the earlier Discovery phase, because the signature that was used to detect the malicious information has already decided on its nature. A detection against a database of known malicious activities implies that whatever is detected, is also malicious, and the Action phase can execute the information stream accordingly (in the case of antivirus the agent can quarantine the file). But using centralised research does have an additional, intrinsic cost associated with it. As certain requirements need to be generalised for the signature to be significant to as many nodes as possible, consumers of centralised research pay an additional fee for the lack of specialisation. This fee materialises as an increased rate of false positives (FPRate), and is levied on each deployed node (n). The cost associated with FPRate is that of repairing any damage done by signatures that take action against legitimate traffic, such as retrieving quarantined information or restoring from backups.

$$Cost : Action = n * FPRate \tag{3.4}$$

Where:

$FPRate$	is the false positive rate
n	is the amount of nodes

The only cost that is linked to DtD Action phase is the cost of correcting for false positives,

for all nodes.

3.1.3 Costing Example of DtD Security Model

Extending the DtD equation to a security control like antivirus highlights the ease at which security controls that are designed with this security model scale. Assume that it takes an analyst on average four hours to fully research a potential threat and produce a signature that can detect it, and that the average analyst has a monthly cost to company of R40 000 per month (ITweb, 2013) or an hourly cost to company of R227. If 200 threats need to be analysed everyday, a small amount compared to the 1 017 208 examples found in the first half of 2010 by G-Data (Willems, 2010), then the total cost to research those threats would be R181 600. Assuming that a medium sized organisation has 1 000 devices on the network, and applying that to the DtD Discovery costing Formula 3.1, we see the cost benefit of scaling research.

$$Cost : Detect = (TCoR/n) + (DC * n)$$

= 181 600/n + (DC * n) *[TCoR = 181 600]*
= 181 600/n + (0 * n) *[Assume DC = 0]*
= 181 600 / 1 000 *[Assume n=1000]*
= 181.6

If the research is distributed to 1 node, the Cost of Discovery is R181 600, but if the research is distributed to 1 000 nodes then the Cost of Discovery per node is R181.60.

To complete the total cost of the security model, the Action phase also needs to be included as per Formula 3.4. The Action phase has no cost that is linked to the research cost or the number of nodes that is included, but by increasing the amount of nodes that the research can cover, the research needs to be made more generic so that it is still relevant to a wider group of nodes. By relaxing the specific criteria of the research, the probability of a false positive occurring within the control is increased. It is assumed then that the false positive rate is optimised in terms of the business owning the security control, and set to a point where the maximum amount of nodes can be attached to the research. A false positive is a cost that must be borne by the vendor in terms of the security control's effective coverage and the market sentiment around any failings in that coverage. Due

to this it is assumed that the business will decide on how many nodes can be attached without impacting the business' continuity, meaning that the Cost of Action for a DtD model sets a theoretical maximum amount of nodes that can be attached to a piece of research. As long as the amount of nodes is below this level, the Cost of Action is a constant.

$$Cost : Action = n * FPRate$$

$=$ Optimised *[Assume FPRate is optimised as described ealier and is a constant]*

The DtD equation set shows that the full extent of the cost associated with a DtD security model is applied in the Discovery phase.

3.1.4 Costing Example of DtP Security Model

If we compare the above cost to the DtP equation as based on a honeypot, the differences in phase costing are apparent, as well as the scaling limitations of the traditional honeypot. As Formula 3.2 shows there is no cost associated with the Discovery phase. The honeypot is also already using unproductive information assets, which have no usage cost, and the cost of listening for anything without complex detection tools is also zero.

$$Cost : Presence = if(i)$$

$= 0$ *[Assume 0 as no complex detection is needed]*

The DtP Discovery action shows that there is no cost related to completing this phase and that there are no limitations on scaling. This changes severely when the model is completed by adding the Action phase costing, as per Formula 3.3. If we assume that a single node is able to log 200 incidents per day, and on average 1% of those incidents are analysed. Assume that it takes an analyst on average four hours to analyse each incident and that the average analyst has a monthly cost to company of R40 000 per month or an hourly cost to company of R227. Finally it is assumed that there are four nodes deployed.

The calculation shows that in order to action intelligence that is gathered by a honeypot in the Discovery phase, R7 264 needs to be spent per day. The cost is directly linked to

$$Cost : Action = (I * threshold) * RCperI * n$$

= (200 * threshold) * RCperI * n *[200 incidents logged]*
= (200 * 0.01) * RCperI * n *[1% of incidents are analysed]*
= (200 * 0.01) * (227*4) * n *[R2790 per hour, 4 hours needed]*
= (200 * 0.01) * (227*4) * 4 *[4 nodes deployed]*
= 2 * 908 * 4
= 1816 * 4
= 7264

the number of nodes that are added (each node adds another R1 816 to the daily total cost) meaning that there is no opportunity for economies of scale by adding more nodes. The diference between the two models can be seen in Figure 3.1.

DtP then enjoys an economical Discovery phase, but is penalised in the Action phase.

3.1.5 Model and Phase Summary

The above formulas and examples show that both security models have competitive advantages, defined in terms of cost, in different phases. The DtD security model has a cost advantage in the Action phase because all of the work has already been done in the earlier Discovery phase. The Action phase simply processes the decisions that the Discovery phase has already made. DtP is superior in the Discovery phase due to its simplistic criteria - only relying on the presence of information to pass it to the DtP Action phase.

Both DtP and DtD security models aim to improve the security of the segment that they protect, but manage to do so using two different methods with different competitive advantages (as measured by cost). It therefore stands to reason that a security model that followed the rules of each model's cost-advantage phase, and still managed to improve security, could be considered a more efficient security model, when compared to the DtP and DtD models discussed previously.

To help minimise the costs it would then be preferable for a system to utilise a hybrid security model, such as one shown in Figure 3.2, that uses the Discovery phase from the DtP security model and the Action phase from DtD security model, where possible.

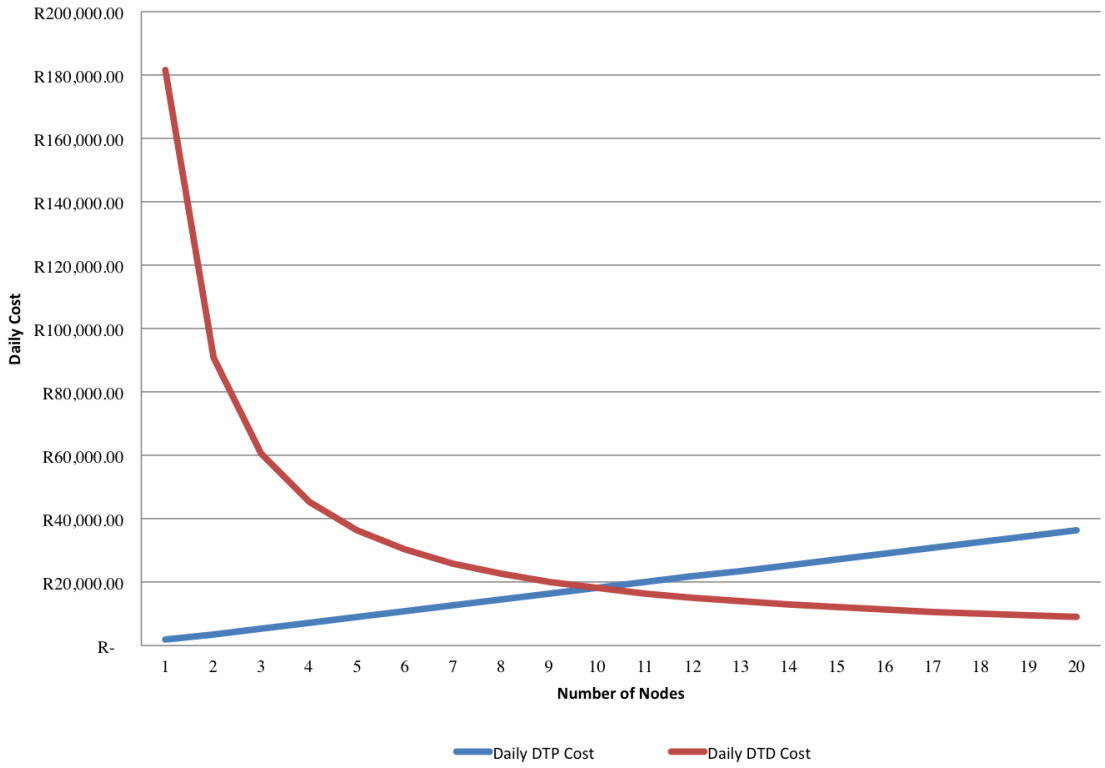

Figure 3.1: Daily cost comparison between DtP and DtD

3.2 Amber: a Hybrid Security Model Implementation

A technical proof of concept was devised that would aim to build a functional system that would act in accordance of a hybrid security model, to test if a system built to satisfy those rules would be able to function in a real world environment. Following the security models explored previously, the hybrid system would need to employ a Discovery phase that is similar to the DtP security model approach (because of the low cost associated with it) but cannot use the DtP Action phase because of the prohibitive costs. Instead the Action phase would need to mimic that of the DtD security model, where action is taken based on the findings of the Discovery phase. Even though the DtD security model has a cost advantage during the Action phase, it does still have an intrinsic cost associated to it in the form of resolving incidents of false positives. Any system based on this Action phase needs to reduce false positives to near zero if it wishes to conserve costs. Lastly, the system would still need to improve the overall security posture of the environment that

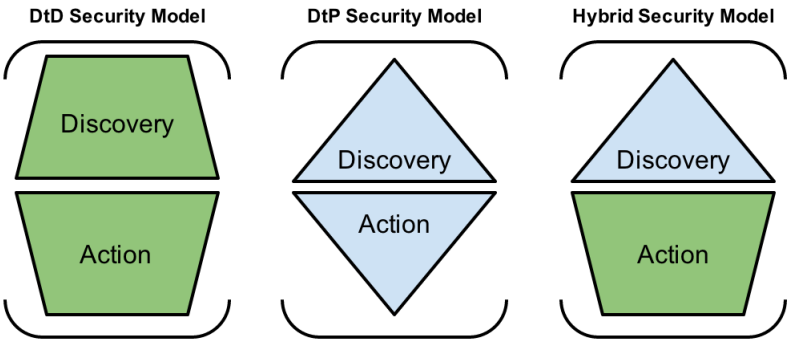

Figure 3.2: The hybrid model

it is connected to, for it to be classified as a valid security control.

Taking this into consideration, The Critical Success Factors (CSF) for the technical system would be as follows:

1. Zero-interactive system (reducing the research cost for both Discovery and Action phase to near zero)
2. Zero false positive rate
3. Improves security posture of the environment

If these can be satisfied then the system would fulfill the requirements of being based on a cost-effective hybrid model, and still serve as a security control. The system was given the name Amber because while it is similar to a honeypot, the low interaction decisions mean that it is less supple than the traditional honeypot.

3.2.1 Zero-Interaction

In order to fulfil the zero-interaction CSF, Amber needs to implement the DtP security model's Discovery phase, which requires the system to occupy unproductive information space. A suitable piece of information space was needed that served productive services to the internet, but that also had unallocated (and thus unproductive) space. A /24 subnet

which housed a group of web servers, each housing multiple websites was identified as the segment to which Amber would play custodian. A web hosting subnet was chosen due to the low likelihood of any accidental neighbouring traffic, as 80/tcp and 443/tcp connections are mostly accessed via their DNS name and not directly through the IP address (Cherkasova, 2000). Figure 3.3 illustrates how Amber is related to the websites because of the close proximity of its IP address, but because websites are rarely referenced via IP address, the chance of it being accessed by someone trying to legitimately access the web servers is low. To maintain the zero-interaction CSF, a suitable Action phase had to be chosen that would lock into the DtP security model's presence-based Discovery phase, but that did not suffer from the high research costs of its corresponding Action phase.

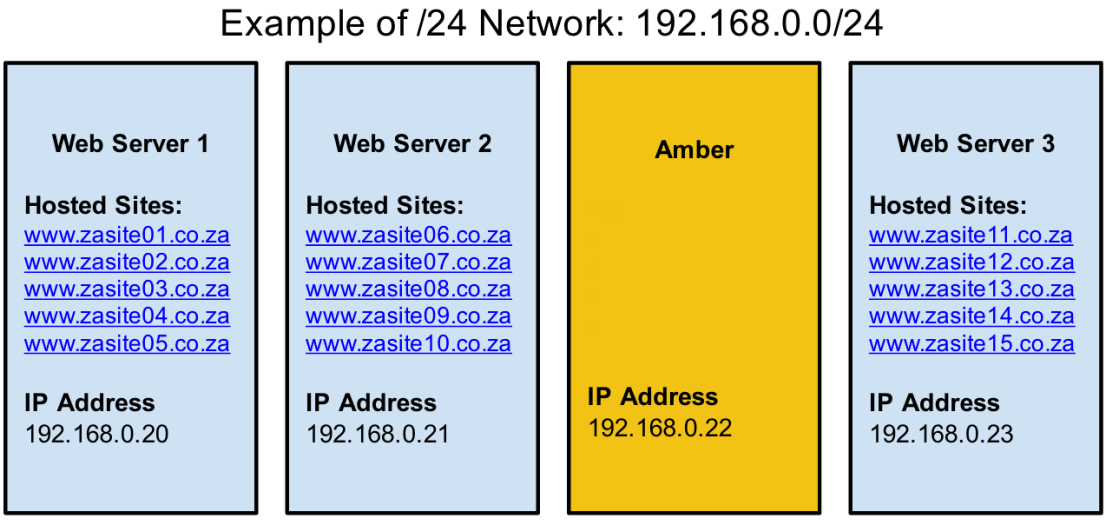

Figure 3.3: Typical Amber deployment

The DtD security model Action plan was also not suitable because it relied too heavily on research done during its own Discovery phase, which is then not available as the DtP simple presence-based Discovery phase was utilised. But it is possible to use the core idea of the DtD security model's Action phase: leveraging off the work done in the previous phase. By identifying any traits during the DtP Discovery phase, it was theorised that the cornerstone of a presence-based Discovery phase (namely the idea that nothing should be interacting with the system as it resides on a non-productive information space) could be carried over into the Action phase.

If information entered the Amber system while it resided on a piece of non-productive information space, the information itself could be judged as also being non-productive. Furthermore, the source of the non-productive interaction can be labelled as a non-productive source because there is no reason for anything to be interacting with this system. Based on that label, the Action phase can terminate any further interaction with the non-productive source. This theory allows Amber to use the rapid (and resource inexpensive) Discovery phase reasoning of the DtP security model in both phases, and links them together by leveraging the theory of the DtD security model. That is to say, it creates a hybrid security model, using the least-cost phases of the DtP and DtD security models.

3.2.2 Near Zero False Positives

The second CSF is a symptom of generalised classification criteria – a part of centralised research. As noted by Slaughter, Harter, and Krishnan (1998), false positives prevent a system from being automated, which increases the interaction rate. Systems that use the DtD security model are prone to it because the detection mechanisms used during the Discovery phase are based on a set of generalized criteria that try to suit as many nodes as possible, which has led to scenarios where critical system files were erroneously labelled as malicious (Keizer, 2007; Microsoft, 2010; Leyden, 2010). As more generalisations are made regarding the detection criteria, the possibility of something being detected and flagged as malicious also increases. As the number of inclusive nodes (the amount of nodes that are able to subscribe to the centralised research) increases so too does the degree of required generalisation, and with it the false positive rate. Given that the Action phase depends heavily on the research done during the Discovery phase, actions taken in the Action phase are also prone to false positives caused by general detection criteria.

As stated earlier, systems that use the DtP model are resilient to false positives because the Discovery phase is tailored to look at information that enters zones that have no productive use, and that are specific to the environment that they are attached to (Zhang et al., 2003). It combats false positives because the information needs to be fully analysed as part of the Action phase, meaning every context requires manual intervention. Amber discards the confidence built by analysing every context, but because the Action is based on the false positive resilient presence-based Discovery phase, any actions taken in the

Action phase would be based on information that had no productive use, making the probability of false positive extremely uncommon. To put it another way: given that all interactions during the DtP Discovery phase are unwarranted, it is unlikely for any action to return as a false positive in the Action phase, i.e. as non-malicious.

3.2.3 Improves the Security Posture of the Environment

The system is still an information security system, which means that through all its zero-interaction optimizations it still needs to improve the security posture of the environment that it is associated with. Without satisfying this requirement, the system would be useless. As a standalone unit, Amber is not able to take action against anything that passes through the Action phase, because it does not use a productive internet address space. It is also not positioned in such a way that would allow it to take action on behalf of a productive asset, such as an inline configuration. As Amber is not attached inline or in front of, and does not share internet space with any productive entity, any action that Amber takes directly on the information that it has seen, would only improve the security posture of the unproductive internet space on which Amber is residing. To overcome this Amber would need to rely on an external device that is situated in front of a productive portion of the segment, and that could take action on Amber's behalf. Amber would be able to log the source IP addresses of systems that targeted it and pass those IPs in the form of a threat stream to the network enforcer (such as a firewall). The firewall would then be able to drop packets that originated from the source IP addresses, and that attempted to connect to any other systems on the segment. This action would reduce the quantity of potentially malicious traffic entering a segment, and thereby increase the security posture of the environment that the firewall in conjunction with Amber was protecting. The IDS SNORT uses a similar approach as it also does not have the native ability to intervene against an information stream (Koziol, 2003).

3.3 Technical Design

Amber was built on top of an established operating system to save time and resources, and utilised that system's available network stack and commonly available packages wherever

possible. A combination of Perl, Python and Bash scripts were used to code the parts of Amber depending on where the strength and development speed of each stage lay. The project was broken down into its functional pieces, and developed as a collection of those pieces instead of as one large program, illustrated in Figure 3.4. This allowed for rapid development by prototyping each function, and then testing it in isolation before building the next piece of the system. The functions were dictated by what was needed to achieve the various goals within each of the phases of the hybrid model, namely Discovery and Action.

The Discovery phase requires some way to discover information that is trying to interact with Amber, and is able to record the source of the connecting entity, and satisfy the entity's presence. It also needs to have a way to ensure that the connecting host's presence is not being spoofed, as to force Amber to think that a benign host is trying to access it, and subsequently force Amber to take action against it. The Action phase requires a mechanism to send the source IP addresses to an upstream network enforcer, who can take action on Amber's behalf. Lastly, because Amber is designed to be unmanaged, there was a need for a module that would ensure that the different components of Amber were functioning correctly, and could restart them if that was not the case. This module was built as a watchdog script, and would check the integrety of the Amber components

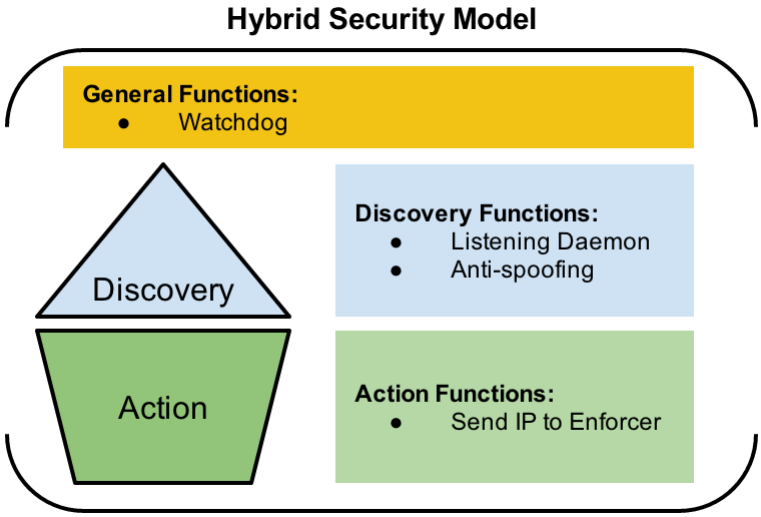

Figure 3.4: Hybrid security model components

3.3.1 Hardware

Initially Amber was deployed as a virtual machine on a Dell R210 running VMware ESXi 5.0. The decision to use virtual machines was taken because the hardware specification of the Dell R210 that was on hand (Xeon quadcore processor and 8 gigabyte of RAM and 4 gigabit ethernet ports of which 3 ports were used) was well beyond what was needed for Amber to function as a standalone deployment. Amber at one point successfully deployed on a virtual server that had a single core processor and 256Mb of RAM. The hypervisor also provided a way to employ the excess resources on other projects or ancillary systems.

3.3.2 Discovery Function: Listening Daemon

In order for Amber to adhere to a Discovery phase that is similar to that of a DtP security model, it would need the ability to open an arbitrary amount of IP ports and accept connections to those ports. The connection's source IP address then needs to be recorded for use in the Action phase.

This introduced a problem into Amber that made it vulnerable to spoofing attacks, which were originally discovered by Heberlein and Bishop (1996) and later expanded on by Templeton and Levitt (2003). A spoofing attack is executed when an attacker changes the source IP address of a crafted packet, effectively masking or spoofing the origin of the packet (Felten, Balfanz, Dean, and Wallach, 1997; Ferguson, 2000; Tanase, 2003). If an attacker knew the listening IP address of Amber, they could craft a packet with a source IP address that was not their own and send it to Amber. Amber would respond by classifying the attacker-chosen source IP address as a threat and streaming it to the network enforcers. This is similar to attacks highlighted by Hunker, Hutchinson, and Margulies (2008), and would allow an attacker to selectively deny a victim services that were protected by a network enforcer that subscribed to an Amber node's threat stream.

The first step in addressing possible abuse of this was to limit the scope of what Amber listened for to only TCP/IP instead of other IP protocols such as ICMP, UDP and GRE because TCP/IP, as noted by Stevens and Wright (1995) has built in sessioning, which helps defend against spoofing. The Linux kernel firewall, netfilter, is a stateful firewall

meaning it is able to keep track of connections as they interact with a system (Coss, Majette, and Sharp, 2000). It was used by Amber as a way to detect any incoming communications that successfully established a connection via the TCP/IP three-way handshake or the SYN, SYN-ACK, ACK process (Postel, 2003). If an incoming connection managed to establish a three-way handshake it would then indicate that the source IP address provided is the actual IP address from where the connection originated. While checking for the three-way TCP/IP handshake is a valid way to mark a connection as being established and not spoofed, an attacker would still be able to fool this method by guessing the correct SYN-ACK sequence number, as proposed by Bernstein (1997) in his paper on SYN Cookies.

Since then, Linux has by default enabled the proposed remedy for this, which are SYN cookies. But, in a paper by Kaminsky (2011), the author revisited the solution and noted that they are susceptible to a brute force attack due to the increase in possible internet throughput. It would take approximately eight million packets to successfully brute force a SYN cookie-protected system, which could take less than fifteen minutes on non-specialised hardware (assuming a packet rate of 100,000 packets per second).

More recently Lell (2013) discovered that the Linux kernel is in actual fact accepting 32 valid combinations for a single SYN cookie, due to the way in which the kernel encodes a counter and the maximum message size (MMS) value into the initial sequence number (ISN). This means that the work factor required to brute force a valid SYN cookie is effectively reduced by a factor of 32. Lell goes on to show that with a gigabit network connection over 300 000 packets per second can be generated, which would exhaust the amount of time needed to guess a valid, and now weakened, SYN cookie to eight minutes.

To address this, Amber incorporates the netfilter packet limit module, which limits the amount of packets entering the system (Welte, 2000). By instituting a limit of 800 ACK packets per second an attacker would have to spend 50 hours to exhaustively brute force all possible combinations and successfully guess a SYN Cookie. To further defend against brute forcing SYN Cookies, Amber's connection timeout was set to 600 seconds. An attacker could effectively launch a Denial of Service (DoS) attack against Amber because of the low packet per second limit, but because Amber is not attached to any productive internet space, a DoS attack directly against it would not harm the segment. In fact it

could be argued that it is slightly enhancing the network's security because a would-be attack is spending time targeting Amber instead of a productive asset.

Having established the constraints of what Amber would listen for, a network daemon was written in Perl and deployed on a virtual machine running Ubuntu 12.04 x64. The Perl script would spawn a set of common TCP/IP ports on the listening interface and record the source IP address of those connections.

The following three TCP/IP ports were chosen, as shown in Table 3.1

Table 3.1: Ports that Amber exposes

Port Number	**Description**
135	MS Remote Call Procedure
445	MS Active Directory
3389	MS Remote Desktop Protocol

TCP/IP port 445 and port 135 were chosen because they have been widely used by malware such as worms (Shin and Gu, 2010; Zhang et al., 2010; Irwin, 2013a). The service running on TCP/IP port 3389, Microsoft Remote Desktop, was chosen because a remote code execution vulnerability had recently been discovered by Auriemma (2012); Microsoft (2012).

3.3.3 Action Phase: Send IP to Enforcer

Once the listening daemon has validated a source IP address, the Discovery phase ends and the Action phase starts. In order to implement the findings of the Discovery phase the IP address needs to be recorded and passed on to a network enforcer. This resulted in another constraint on the system, because it was being deployed in an environment that was 'live' on the internet, and used to service client requests. This meant that the environment could be seen as a Production environment, which meant that any action that could possibly cause an interruption to clients trying to access their system, could not be followed through. This constraint meant that Amber was unable to take action against IP addresses in the way previously thought (passing the IP to a network enforcer where it could be denied further access to the network). Testing could also not be done

in a test environment because without access to a segment that was actively interacting with the internet, would not reflect a real world scenario.

A work around was needed that would simulate a complete interaction with Amber and that could measure the effect that blocking the source IP from the network would have, without physically interacting with a network enforcer. To address this a proxy test was engineered that would cause no interruption to the production network but would still be able to validate Amber's ability to improve the security posture of the network segment. This would rely on a packet capturing device that would be set up at the perimeter of the network segment that Amber was housed in, that would normally be behind the proposed network enforcer, and would capture all the network traffic entering the system over a certain period of time to offline pcap files. Amber would then log all the source IP addresses of connections that it recorded for the same time period, which could then be compared to the full pcap traffic dumps, to check if an IP address that Amber logged was later found trying to access another asset on the network, as per Figure 3.5.

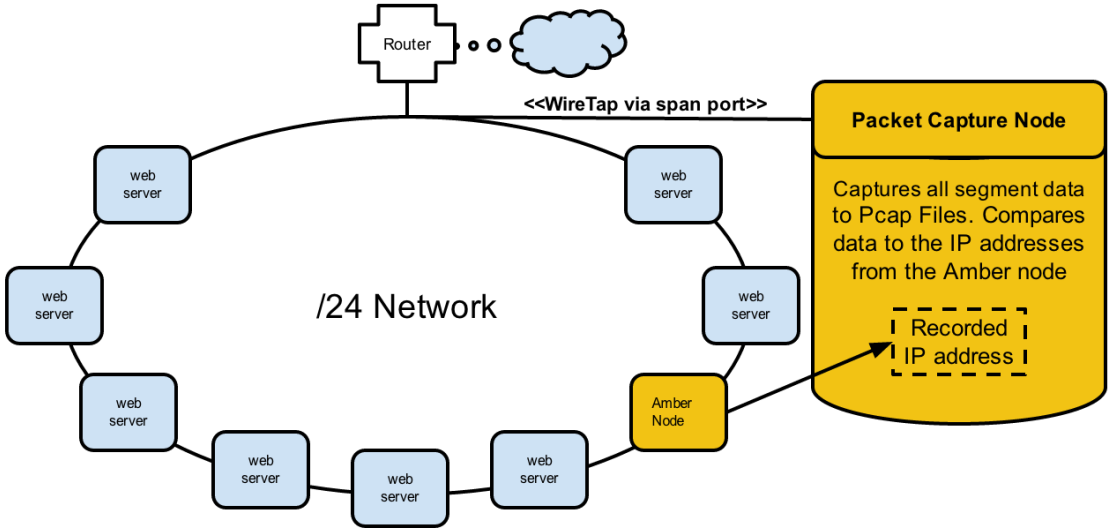

Figure 3.5: Malicious activity verified via wiretap

If this was the case, then it could be stated that if Amber had been fully functional and connected to a network enforcer, the logged source IP address would have been added to a DROP rule on the network enforcer, which would have stopped it from being able to

interact with the other system on the network, thereby increasing the security posture of the network segment.

The packet capturing system was deployed as another virtual machine on the VMware ESXi 5.0 server and assigned two network interfaces. The first was connected to the management network so that it was reachable, while the second was attached to a duplicate stream of all the traffic entering the shared web hosting network. This was accomplished by spanning or mirroring (Phaal, 2007) the trunk port on the switch that led to the external facing port on the network firewall, to another port on the same switch. This port was then connected to an open interface on the VMware ESXi 5.0 machine. The ESXi 5.0 vSwitch is able to split the different VLAN traffic that a physical interface sees, to different virtual ports. The VLAN (McPherson and Dykes, 2001) that consisted of the shared web hosting traffic was attached to one of the virtual ports, and to the secondary interface on the packet capturing system. This allowed the packet capturing system the ability to see all traffic entering and exiting the network segment, but unable to affect it in anyway.

The packet capturing system was also built on Ubuntu 12.04 x64, and used the tcpdump binary to capture all incoming traffic to the segment via a mirrored uplink port. A truncated version of the packet is captured because an analysis of the complete packet contents is not needed, since only the source and destination IP address needed to be checked. As detailed in chapter 2, Amber and the packet capturing system, would be active for 24 hours during which time the packet capturing system would record all the traffic entering the segment. At the end of the 24 hour period, Amber's daemon would have logged all the validated source IP addresses locally in a flat file, and this list would be exported to the packet capturing system where it would search through its packet captures over the same time period. For each IP address logged by Amber, the packet capturing system would check if the same IP address attempted to target another host on the segment, an action that would not have been possible if Amber were able to stream the source IP addresses that it captured to an upstream network enforcer.

3.3.4 Ancillary Systems: Packet Capturing System

The packet capturing system became an essential part of the Amber testing process because it allowed Amber to be tested on a production network with numerous websites, but it also increased the complexity of the test suite. Complete packet captures for all of the traffic that entered the entire segment made it possible to do offline analysis on the traffic, but also meant that the data would need to be stored until it was analysed.

The network segment that Amber was housed in would, on average, generate 300 Gb worth of captured pcap files on the packet capturing system over a 12 hour period, with network traffic spiking to 30MB/s during peak usage times. This created a burden on storage which was limited to 1TB on the packet capturing system. This meant that the maximum amount of data that could be housed at any one time was 980 Gb, or approximately 36 hours worth of packet captures. Storage is only half of the problem of offline analysis, as the files still need to be analysed, which also added a great deal of time needed to complete an analysis window. Because storage was only necessary for testing purposes it would not be a limitation on a production instance of Amber.

Initially the command line implementation of Wireshark (a binary called tshark) was used to analyse the pcap files (Merino, 2013). The captured packets were analysed for all instances of a source IP address that the Amber node had logged, and would return a list of timestamps when that IP address had also tried to access other assets on the same network. All instances that occurred after the timestamp that Amber had originally logged the source IP address would indicate instances that would not have occurred if Amber had informed an upstream network enforcer of the source IP address. This total would form an effectiveness rating for Amber; measuring how many additional connections and potential malicious information exchanges could have been prevented after Amber had logged that source IP address.

This method ended up being very computationally expensive. To analyse a total of 5.5 Gb of packet captures, split into eleven 500 Mb pcap files, for a single logged source IP address, took on average 5 minutes and 4 seconds. This meant that it would take the the offline packet capture system just under 5 hours to process a 12 hour window of traffic for 1 source IP address. The Amber node was recording between 10 and 20 source IP addresses

per 12 hours, which meant that it would take between 10 and 20 times what was needed to do a single source IP address, or 50 to 100 hours, to fully analyse a 12 hour window. While it was possible to generate results with this model, it was very unproductive and meant that there was no room to scale the architecture. Also, if an outlier day were to occur which caused Amber to log an exceptionally large amount of source IP address, the day's research would have to be discarded due to it being impractical to analyse.

In order for tshark to process a packet capture file, a filter needs to be constructed that tells tshark what to look for in the packet capture file. For example Figure 3.6 shows a tshark query that has been constructed to look for a single source IP address, in this example 8.8.8.8, in a pcap file ambercap.pcap. Line 1 of the tshark query sets ambercap.pcap as the target file. Lines 2-5 list the fields that are to be extracted if the condition is met. Line 6 sets the condition filter that needs to be matched to return a value.

```
1  tshark -nn -r ambercap.pcap -T fields -E separator=';' \
2      -e frame.time \
3      -e ip.src \
4      -e tcp.srcport \
5      -e tcp.dstport \
6      '(ip.src == 8.8.8.8)'
```

Figure 3.6: Tshark command

It was discovered though that if multiple filters were linked through an OR condition, the processing time that it took to complete the query did not increase proportionately to the number of source IP addresses added. Therefore instead of constructing a filter to look for a single source IP address, a filter could be constructed that linked many source IP addresses with the OR condition. This allowed the packet capture files to only be searched once, requesting all the IP addresses that had been logged by the node during that capture window, as shown in Figure 3.7 lines 6-10.

Table 3.2 and Figure 3.8 display further tests that were run to determine if this held true for n inputs, or if there came a point where this observation no longer held true. The input filter was tested with the following sizes: 1, 5, 10, 50, 100, 500, 1 000, 2 000 and 4 000 and ran against a pcap file that was 500mb in size. The Unix time command was used to ascertain how long it took to analyse the pcap file with each of the seven filter sizes, as described by (Das, 2006).

```
tshark -nn -r ambercap.pcap -T fields -E separator=';' \
    -e frame.time \
    -e ip.src \
    -e tcp.srcport \
    -e tcp.dstport \
    "ip.src == 114.80.157.114 \
    or ip.src == 123.200.148.102 \
    or ip.src == 141.212.121.10 \
    or ip.src == 95.34.22.1 \
    or ip.src == 33.54.1.2"
```

Figure 3.7: Tshark command - multiple conditions

Table 3.2: Processing times as source arguments increase

No. of Src IPs	Real Processing Time (seconds)	Seconds per IP
1	9.247	9.247
5	9.205	1.841
10	9.553	0.955
50	10.860	0.217
100	13.367	0.134
500	26.523	0.053
1000	39.158	0.039
2000	63.900	0.032
4000	112.741	0.028
5000	na	na

This testing does show that the processing time does increase as the number of filter inputs increases, but not proportionately while n $<$ 500. Increasing the number of inputs from 1 to 500, causes the processing time to increase from 9.247 seconds to 26.523 seconds. This is an acceptable increase in processing time since it is an increase of 2.868 times that of 1 input, but results in 500 times the number of inputs being analysed. Doubling the inputs from 500 to 1 000 increases processing time to 39.158 seconds, which is still an improvement as it still takes less time to analyse each IP as shown by how many seconds its takes to analyse each IP in Column 3. Even using 4 000 source IP addresses still manages to perform better than 2000 IP addresses, albeit only slightly, and anything over 5 000 IP addresses causes an 'Argument list too long' error to be triggered in the tshark binary. Based on this, as long as the argument list does not exceed 5 000 arguments the processing time is efficient to stack as many filter arguments into one query as is possible.

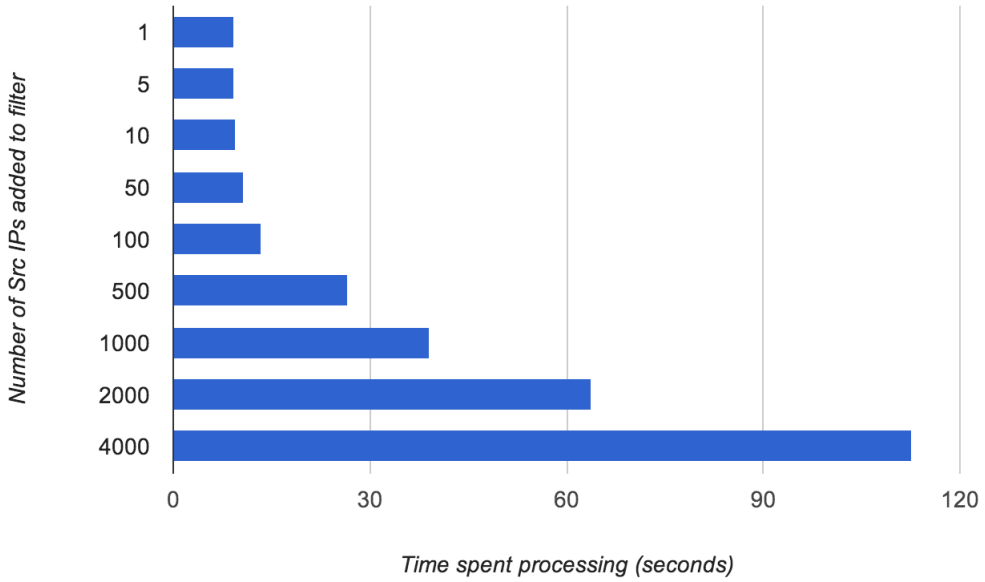

Figure 3.8: Processing time per IP (500mb pcap file)

Using this method of analysing the pcap files resulted in an effective disregard for the amount of IP addresses that needed to be analysed, because the average number of source IP addresses that needed to be analysed was between 10 and 20.

3.3.5 Further Gains with Parallel Processing

By using a stacked query as previously described, the time to process 12 hours worth of pcap files was not hampered by the amount of source IP addresses that were being analysed, but it would still take close to five hours to complete an analysis. This number was a function of the speed of the processor and, to a lesser degree, the input/output speed of the hard drive being used. While the tshark binary was analysing a file, the core of the CPU that it was tied to would report as being in 100% use, but the remaining cores of the quad core Xeon E3-1235, would be standing idle. This was because tshark is not a multithreading binary, meaning that it cannot use all available cores for a single query, which meant that the extra cores on the E3-1235 were underutilised when a pcap file was being analysed.

Because the pcap files are stored on the hard drive in 500Mb 'chunks' and a script is used to iteratively analyse each of the files, it was theorised that a form of parallel computing could be simulated by a control script. Spawning multiple queries against different pcap files, and shuffling those queries to different processing cores, meant that more of the available computing resources could be used to analyse the pcap warehouse. By increasing the resources available it should be possible to cut down the total time needed to analyse a 12 hour capture window.

The Python programming language has a parallel processing framework called pprocess, which exposes certain tools that make it possible to implement parallel programming within a Python script (Schiemenz and Igel, 2013). Instead of rewriting the query generator to use Python, and thereby make use of pprocess's multi-processing capabilities, it was decided that it would be more time efficient to rather reuse the shell scripts that had already been coded and tested, and use pprocess as a wrapper. Because Python can also launch shell scripts with the subprocess module, it was possible to write a Python script that would be able to take a shell script that needed to be applied to a directory of input files, and spawn multiple instances of the script while iterating through the input files.

Figure 3.9 shows a generic version of the script, one that could be applied to any shell script and directory of input files.

```
import subprocess
import os
import pprocess
import time
start = time.time()
def wshark(pcap):
    subprocess.call("script.sh {0} ".format(pcap), stdout = open( 'log', 'a'), shell=True)

directory_loc = "/warehouse"
file_list = []
for path, subdirs, files in os.walk(directory_loc):
    for name in files:
        file_list.append(os.path.join(path, name))
# Parallel computation:
nproc = 3       # maximum number of simultaneous processes desired
                # available cores -1 is preferable after testing
results = pprocess.Map(limit=nproc, reuse=1)
parallel_function = results.manage(pprocess.MakeReusable(wshark))
[parallel_function(args) for args in file_list];  # Start computing things
parallel_results = results[0:len(file_list)]
print 'It took', time.time()-start, 'seconds'
```

Figure 3.9: Python wrapper for parallel data analysis

The variable `nproc` tells the pprocess framework how many instances of the shell script it can handle in parallel, which in the above configuration is set to 4, which yielded the best performance. The machine that was doing the packet capture analysis was equipped with a single quad-core CPU, giving it four cores onto which it could schedule tasks without impacting performance. Therefore, if `nproc=4` it is maximised for the amount of available processing power.

Additional testing was done on all possible values between 1 and 8 for nproc, it was found that when nproc exceeded the number of usable cores on the hardware, no further improvement in processing time was examined. The empirical test placed seven items, into the analysis directory, which was made up of seven packet capture files of 500Mb each. The first test did not use the pprocess parallel scheduling capabilities and instead just tested the speed at which the framework would execute on a single processing unit. This was completed in 66.294 seconds or 9.471 seconds per individual file, which is very close to the results examined by using the script directly from the command shell.

After that the test was rerun with the nproc variable incremented and the results recorded. As Table 3.3 and Figure 3.10 show, by adding an additional processing core into the process, the time it takes to analyse the files is improved but not perfectly. The addition of another processor into which data can be analysed does not halve the amount of time needed to fully analyse the content. When moving from one to two processes the job is completed in 60.5% of time that the single processor needed, which is close to half of the time but not exactly. This disparity in performance gains can be attributed to the analysis of capture files not only relying on the CPU to be completed, but also on the hard drive. In order for tshark to start analysing data, that data needs to be read off of the disk and parsed to the binary, but while the binary can be assigned different files on different processing cores, the data files reside on one hard drive. Each request to read a file adds to the single hard drives load, and while the effect is not very pronounced when analysing two files in parallel at `nproc=2`, when another core is added in at `nproc=3`, there is a much smaller decrease in the amount of time needed.

The total time that is needed to analyse the seven files does continue to decrease, albeit at a slower rate at `nproc=4`, but when the script is configured to start assigning five concurrent instances of tshark, the time needed starts to increase compared to what was

needed to `nproc=4`. This means that nproc=4 is the turning point resulting in a processing time reduction of just over half, which led to an average analysis time of just under 6 hours for a 12 hour packet capture.

Table 3.3: Identifying optimum NPROC setting

NRPOC	1	2	3	4	5	6	7	8
Items	7	7	7	7	7	7	7	7
Total Time (s)	66.29	40.11	37.13	32.20	35.31	35.08	38.95	36.29
Per file (s)	9.47	5.73	5.305	4.60	5.04	5.01	5.56	5.18
Performamce (0 is better)	100	60.50	56.01	48.57	53.27	52.92	58.76	54.75

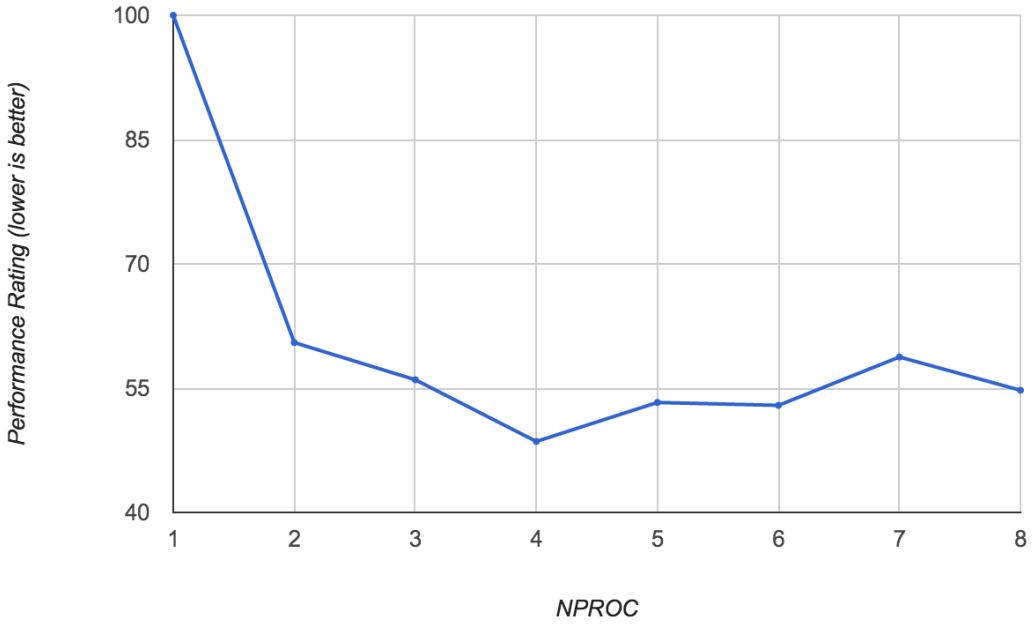

Figure 3.10: Identifying optimum NPROC setting

3.4 Implementing Distributed Intelligence

Sudaharan, Dhammalapathi, Rai, and Wijesekera (2005) notes that a honeypot's usefulness is measured by the amount of information that it collects, so increasing the amount of information that it is able to capture would increase its usefulness. An architecture

that would maximise the quantity of data that can be captured should then be implemented. More captured data should lead to an improvement in potential findings, as long as the additional data can be researched at the same rate and quality as before. Amber's DtP-based Discovery phase needed Internet space that had no other productive use, so increasing the amount of Amber nodes on a segment reduces the amount of usable Internet space for that segment. Unfortunately, due to limited available IP address space on a network segment this puts a limit on the number of nodes that can exist on a particular network segment. What is more detrimental to this architecture is that each additional Amber node also erodes the total protection that Amber grants the productive systems on the segment, because there are simply less productive systems. While it might be feasible to sacrifice a small amount of additional IP addresses onto which Amber nodes can be deployed, it would depend on the value that each node adds to the cluster.

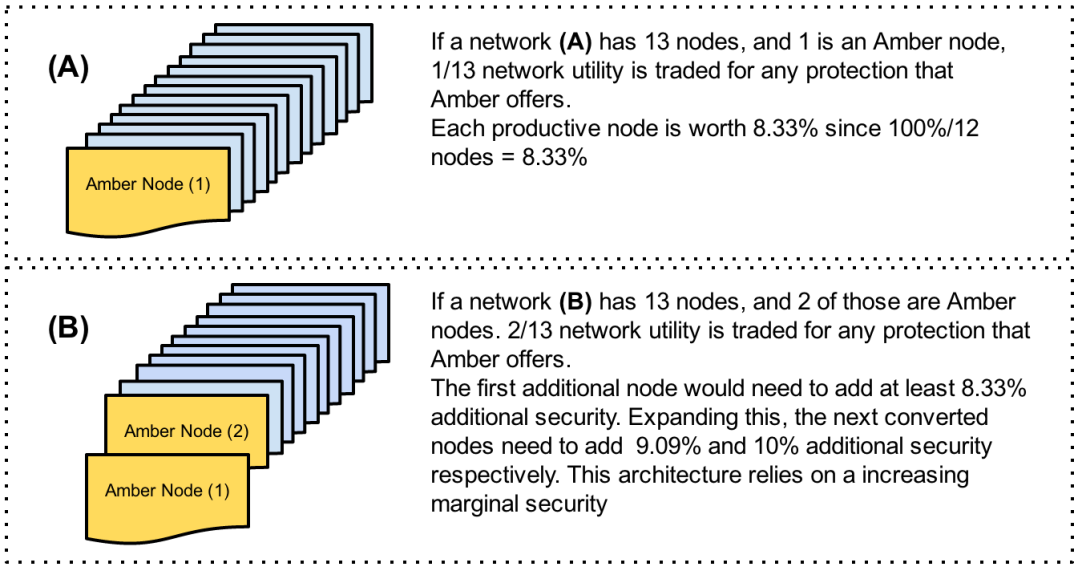

Figure 3.11: Network utility is lost for each node deployed

The architecture in Figure 3.11 is flawed because of its reliance on an increasing marginal security added per node. It was decided that the practice of deploying additional Amber nodes on the same segment was impractical. However, if the trade-off of productive to unproductive internet space could be offset, then any increase in the security posture could be seen as relatively beneficial, because it is not tied to a cost. It was also theorized that sampling internet space that was geographically removed from the first node, would

yield a list of potentially malicious source IP addresses that were significantly different for the first node's list.

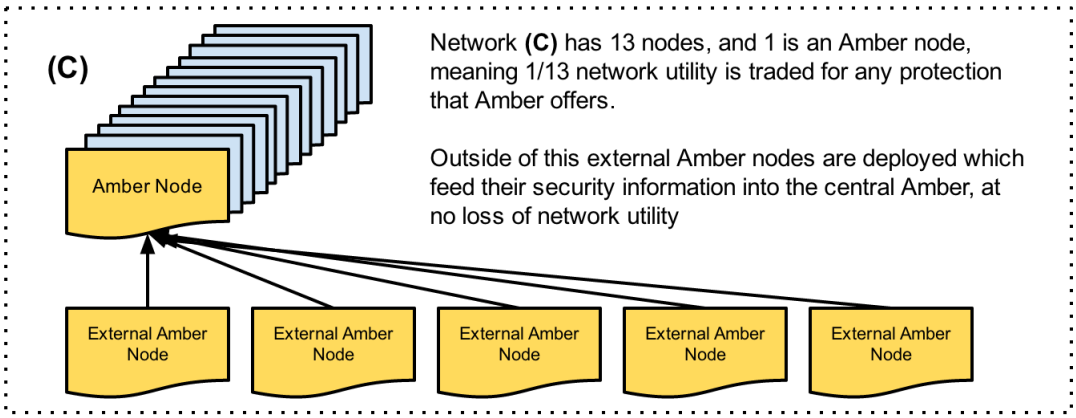

Figure 3.12: No network utility is lost for nodes deployed externally

A new architecture was ratified that would deploy additional Amber nodes on geographically distributed segments as shown in Figure 3.12. This would nullify the cost of converting productive internet space on the network segment that is being protected, and therefore any increase in security posture would be acceptable. Two additional Amber nodes were set up on geographical areas that were separate from the initial sensor. The original Amber node resided in South Africa, while the additional nodes were established in the United States and Germany. The locations were chosen based on the implementation costs of renting infrastructure, and because they are located in different continents and time zones. Dissimilar time zone and geographic location was important because it would create a deployment that would be able to test the speed at which a possible malicious source IP address was identified on different nodes. Figure 3.13 shows that while all three nodes had different time zones, the German (zone O) and South African (zone P) nodes were only separated by one zone, while the United States node (zone G) was eight zones away from the German node and nine zones away from the South African node. This would highlight if there was any difference in the data collected between time zones based on their distance.

Initial testing of the data that was being collected by the additional nodes revealed that there may be a limited number of correlating events between threats recorded in these different locations. Reviewing the threat streams generated by the United States and

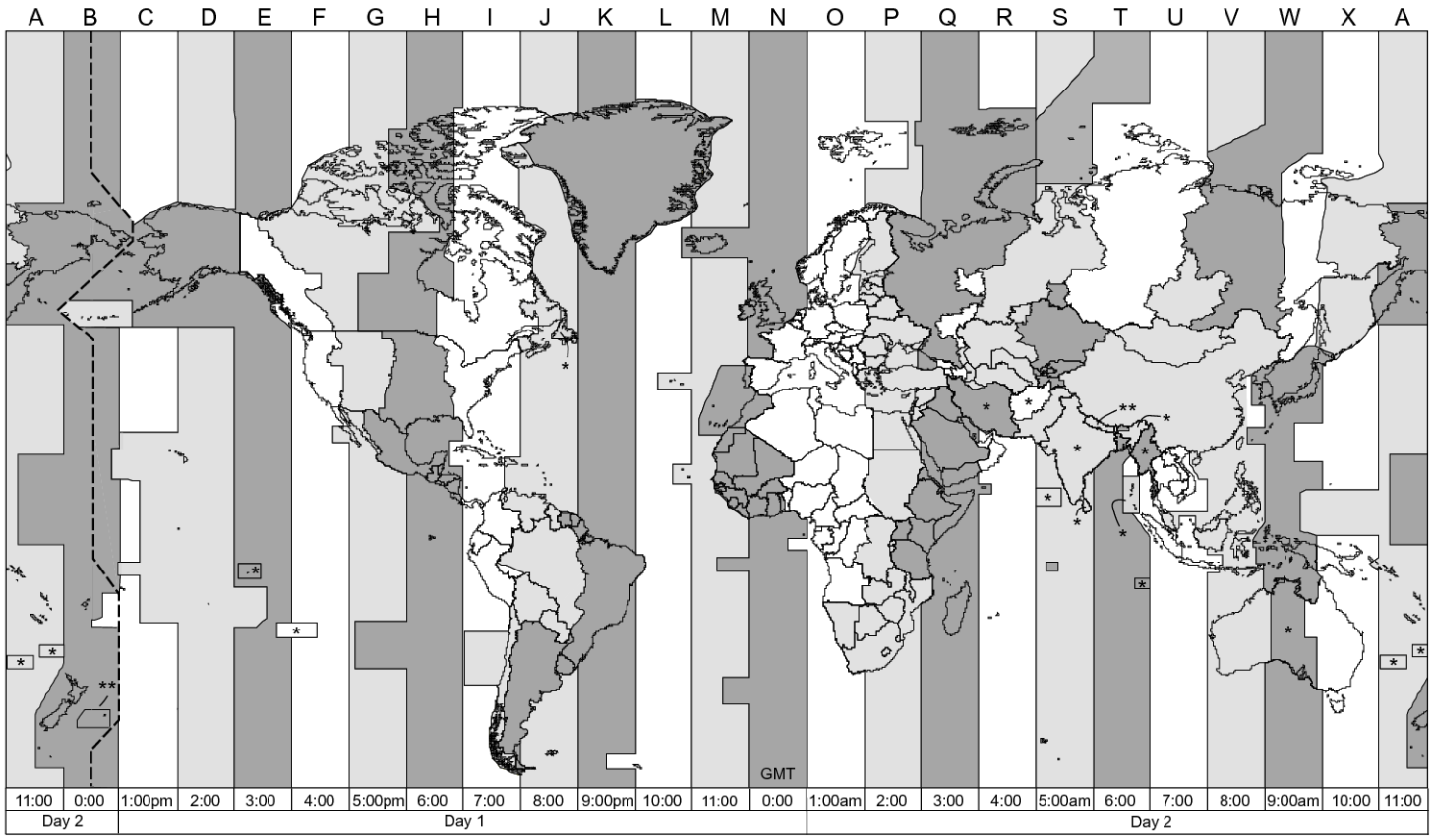

Figure 3.13: Time zone differences between distributed Amber nodes

Source: US Naval Observatory, 2002

German nodes showed that none of the validated source IP addresses that were being captured were appearing on the South African packet capturing node. A precursor hypothesis was devised: is there any correlation between attacks seen between the United States, German and South African geographical nodes? This hypothesis would act as a sanity check to the hypothesis of using distributed nodes to increase the security posture of a single segment. Testing the precursor hypothesis would also not require any additional resources or information because it would compare information that was already being captured. If the precursor hypothesis was false then the distributed node hypothesis would automatically be false, due to its reliance on a correlation between the regions to exist.

The networks that the United States and German nodes were deployed on were commercial hosting environments, which made it extremely difficult to monitor all the network segments that the distributed Amber nodes lived on. A similar test for correlation could be achieved though by comparing the source IP address streams generated by each node. Each node was built on the same basic stack as Amber because their purpose was identical to the original amber, except that the captured IP addresses would be analysed differently. A listening daemon would record any source IP address that communicated with it, relying on the same suite of anti-spoofing technology as the original Amber node. Once a source IP address was verified through connection sessions it was logged to a file, and appended to the list of IP address that the netfilter firewall dropped. Each list of verified IP addresses needed to be compared to the other nodes' lists which meant that a centralised system was also added to the architecture that could collect and correlate the IP information, as per Figure 3.14. Using a centralised node would also allow all interactions with the remote nodes to be orchestrated through one system, cutting down on the amount of time needed to collect information from the remote nodes. This architecture also makes it possible to scale the systems to many more nodes, as it removes the administrative work required to maintain and manage the systems.

The central node would need to be able to log into each of the remote nodes, stop the listening daemon, transfer the verified source IP address list and log files to to its archives, sanitise the remote node's environment and restart the listening daemon. To accomplish this, a trust relationship needed to be established between the central node and each of the remote nodes, through the distribution of public-keys for secure shell login as described

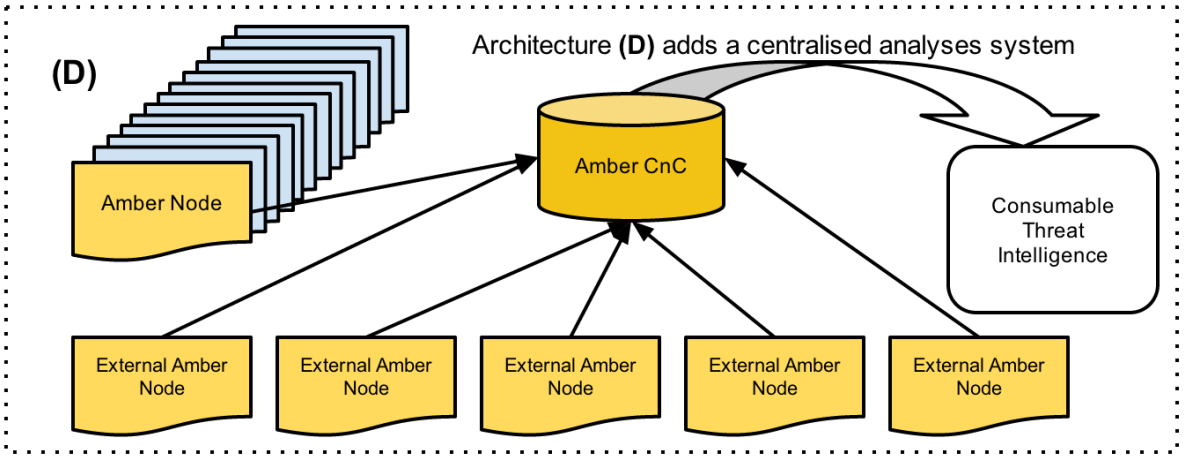

Figure 3.14: Amber architecture with centralised Command and Control system

by Ylonen (1996). This allows the centralised node to access the remote nodes without being prompted for user credentials.

This test only needs the source IP addresses to be stored, instead of full packet captures as with the single node architecture. This meant that the system was also able to run for longer periods of time without intervention. Further, because of the centralised node the interaction could be limited to non-human interaction as the centralised node would be used to pull and analyse the data on the node, and to reset its state. This required the remote nodes to be built robustly enough to function with no interaction or human intervention. Due to the modular nature of the remote nodes, it was possible to write a watchdog script that would check the current state of the node based on what processes were running, and correct any state that was not one of the three desired states: Listening, Maintenance or Disabled. The watchdog would poll certain aspects of the node and collect that data, which was compared to a matrix of possible state indicators. This allowed it to derive the running state of the node to which it compared the desired state via a configuration file switch and corrected, where needed, the systems out of sync. This enabled the nodes to collect data for more than eight months without any direct human interaction or administration.

3.4.1 Capturing Packets

All initial testing and pcap file processing optimisation was based on a 12 hour packet capture window because it was thought that any longer would result in an analysis time slot that would be too long to manage. In practice, this was not the case. Twelve-hour capturing windows required two human interactions with the system, per day cycle. It was not possible to automate these interactions because of the complexity involved. Actions needed to be taken on two separate servers:

1. Starting the test which consisted of sanitising the test environment (deleting all previous packet captures), activating the listening daemon on the Amber node and start capturing packets on the capturing node
2. After 12 hours the window is automatically closed and the data is ready to be analysed. The listening daemon node's collected IP addresses need to be transferred to the packet capturing node, and built into a single tshark query which can be run on the packet captures that were collected. The results are then pushed into a database as discussed in section 3.3

Assuming that the test is started at 06h00, capturing would run till 18h00 at which point it would need to be switched over to analysis mode. This had to be a manual mode because of the complexity in moving and formatting the data between the capturing and analysis phases. The analysis would run for 6 hours, finishing at midnight, and at 06h00 the next day the process can be repeated. Batch process was not possible because of the amount of hard drive space that was required to store multiple windows of captured data. Figure 3.15 shows that when a 12 hour window is used, the system is able to capture 12 hours worth of data every 24 hours, or 24 hours worth of data every 48 hours. This requires two interactions, one at 06h00 and another at 18h00, or four interactions for 48 hours.

Moving to a 24 hour capture window (Figure 3.16) doesn't increase the amount of capturing time when normalised and compared to the 12 hour window, but it does decrease the amount of interactions to two per 48 hours. It also synchronises all interactions to

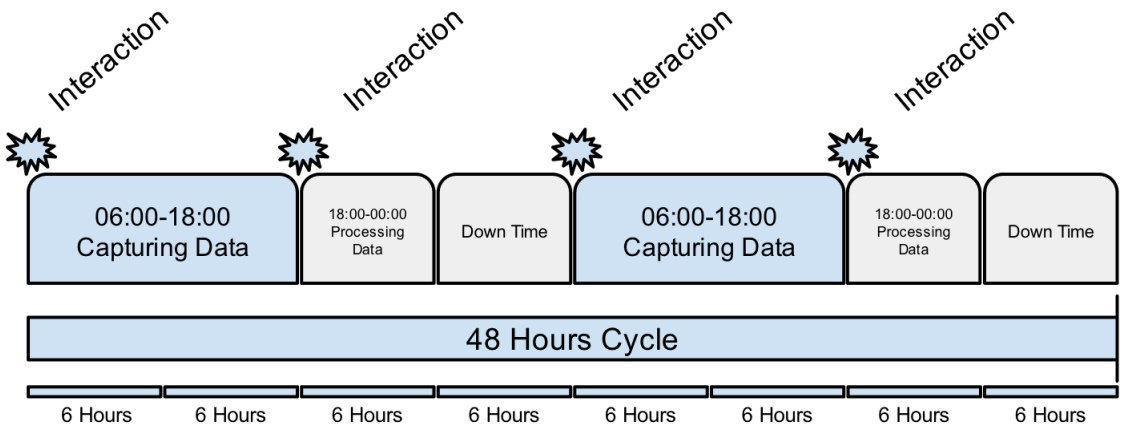

Figure 3.15: 12 hour capture window

occur at the same time every day. Assuming that the test is started at 06h00, capturing would run till 06h00 the following day, at which point it would need to be switched over to analysis mode, which would run for 12 hours and end at 18h00. The next day at 06h00 the process can be repeated.

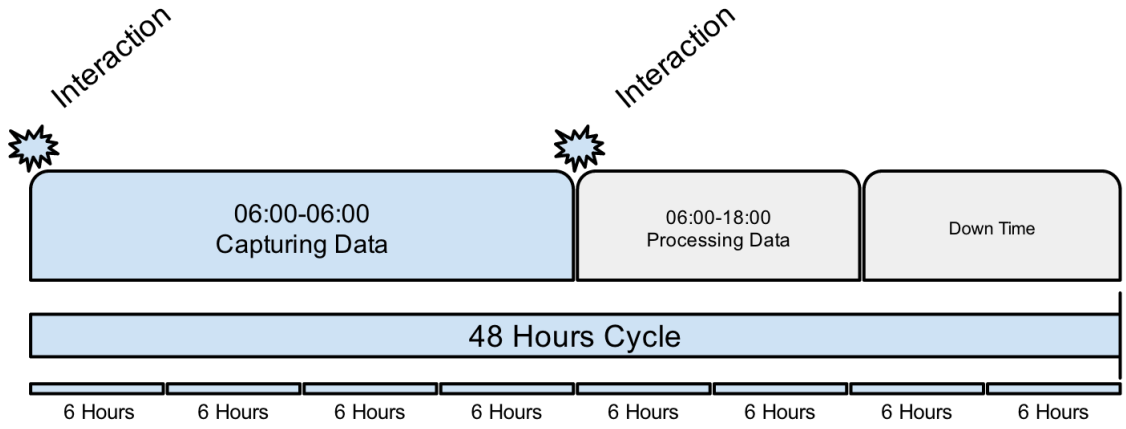

Figure 3.16: 24 hour capture window

The 48 hour cycle meant that the data that was captured would not be biased towards a certain time of day because capturing took place for an entire day and night cycle. Being able to schedule all interaction at the same time every day also reduced the amount of administrative overhead of monitoring the project. For these reasons the 48 hour capturing period was chosen.

3.4.2 Time Synchronization

A common and accurate time-keeping service became another integral part of the testing system, because packet arrival time would be compared between two systems (Kent and Souppaya, 2006; Tierney, Crowley, Gunter, Lee, and Thompson, 2001). If there was a discrepancy in the time between the system that was capturing the segment PCAP files and the Amber node, then the integrity of any derived conclusions would be questionable. To address this, a local network time protocol (NTP) server was used to synchronize the system time of all the systems that were part of the testing environment (Ylonen, 1996). The built-in UNIX NTPD service was used to automatically probe the common time server and alter the local time if it was not synchronized.

3.5 Summary

This chapter introduces the concept of a Discovery and Action phase as common to both detection-based and presence-based security models. Costing methods were produced that served as the backbone for selecting certain characteristics of the hybrid security model, combining the lowest cost phases of each of the two models. Once the model was defined, a technical design was proposed that would meet all the requirements of the hybrid model, which became the Amber system. The system was characterised by a listening daemon and a method of streaming IP addresses, as well as certain ancillary systems that aided Amber. The idea of expanding on the Amber deployment by increasing the number of nodes was explored and it was found that potential gains could be made if more nodes were deployed in different countries opposed to adding more nodes on the same network segment. The next chapter focuses on what findings were discovered when the data that the Amber nodes collected was analysed, as well as certain ways in which an Amber deployment could be used to aid traditional security controls.

4 Data Analysis

Introduction

The single and distributed Amber nodes collected a significant amount of data during the time that they were active. By looking at the data that was collected by the nodes, certain discoveries were made that produced three ways in which security controls could be improved by analysing the data collected by an Amber node on a network. The rest of the chapter is outlined as follows:

Section 4.1 investigates the testing process, and how it was structured.

Section 4.2 presents the data that the local Amber nodes collected.

Section 4.3 extends on Section 4.2 by adding data from two other nodes that were located in different regions. This data is profiled based on top 10 sources and the most prevalent TCP/IP ports used.

Section 4.4 takes the collected source IP address and maps them visually on a world map to show which remote locations were prone to attack, and which were not.

Section 4.5 focuses on the data and constructs three methods in which signature based security controls can be improved by transforming the data captured by a deployed Amber node.

Section 4.6 describes possible deployment strategies for an Amber system, covering a simple and complex network, through an extendable architecture.

Section 4.7 serves as a summary for the chapter.

4.1 Testing Process

Testing was broken down into two parts: single node and multiple node tests. This split exists because the actual testing for single and multiple nodes needed to be different,

due to differences in control over the environments. The single node was deployed on a network to which physical access was granted, meaning that out-of-band testing, wiretaps and high-speed backbone access was provided. This was not the case for the multiple node tests because the geographically distributed nodes were rented from Virtual Private Server (VPS) providers. This meant that shell access and at best an administrative dashboard was made available to the infrastructure during testing.

Choosing where the geographically distributed nodes would be placed (the two nodes outside of South Africa) was decided primarily based on cost, but was also related to hosting concentration. Low cost hosting attracts more demand for hosting, which would make the regions with low cost hosting more representative than fringe hosting providers. LowEndBox[2] is a website that collects VPS specials and publishes them, and it was used to obtain a node in the United States and Germany for under $2.00 per month each. These ultra-cheap nodes are normally limited in terms of CPU, RAM and available hard drive space, but because the Amber node deployment has a small footprint, these limitations did not influence the nodes ability to collect data.

The single node used a method of checking collected IP addresses against the rest of the network traffic with a physical wiretap, but since this level of control was not available on the VPS nodes, a substitute test needed to be devised. This took the form of a correlation test between the three nodes to establish linkage with regard to the IP addresses that they captured.

4.2 Test Results: Single Node with Segment Validation

The 24 hour long capturing test was run a total of ten times over a six-month period from August 2012 to January 2013. Amber's listening daemon would collect the following information from each source IP address that attempted to connect to it: the source IP address, the destination port and a timestamp of when the connection occurred. The data would be stored in a log file as shown in Figure 4.1.

At the end of the 24 hour capturing window, the log file was moved to the packet capturing node where it can be compared to the data that was captured for the entire network

[2]http://www.lowendbox.com

```
        1            2       3
|--------------|------------|--|
Jan  9 14:46:43;96.31.71.114;80
Jan  9 14:53:35;190.207.13.15;445          (1) Timestamp
Jan  9 14:56:37;113.162.232.186;445        (2) Source IP Address
Jan  9 15:11:47;91.197.21.83;445           (3) Destination Port
Jan  9 15:15:47;94.228.30.36;445
Jan  9 15:16:57;217.26.6.69;135
Jan  9 15:28:17;78.84.193.157;445
Jan  9 15:33:54;86.63.121.134;445
Jan  9 15:38:20;222.66.235.209;445
```

Figure 4.1: Amber connection request log file

```
                1                          2         3       4       5
|------------------------------|--------------|----|-----------|---|
Aug 21,  2012 00:53:55.262025000;178.131.62.139;4308;196.x.x.153;3389
Aug 21,  2012 00:56:51.222421000;58.122.108.12;5543;196.x.x.153;3389
Aug 21,  2012 01:10:09.882886000;41.134.152.62;33785;196.x.x.153;445
Aug 21,  2012 01:40:45.543238000;89.2.73.56;22331;196.x.x.153;445

 (1) Timestamp                      (2) Source IP Address
 (3) Source TCP/IP Port             (4) Destination IP Address
 (5) Destination TCP/IP Port
```

Figure 4.2: Amber transaction information log file

segment. Each of the source IPs that the listening daemon recorded would be strung together into a tshark query. This tool would then extract all instances where that IP address also attempted to connect to another system on the segment during the same period of time that Amber and the capturing system were active. This produced a data set that contained the timestamp, source IP address, source port, destination IP address and destination port. An extract of the resulting log file is given in Figure 4.2.

A script would then compare the timestamps from the combined log file against that of the IP transactions that the packet capturing node recorded, discarding those results that occurred before the Amber node logged an IP address. Figure 4.3 visualises this workflow. This would result in a report that only contains the source IP addresses that first attempted to access Amber, and then went on to also attempt to access other systems on the segment. IP addresses that appeared in the processed list would prove that Amber was able to improve the security posture of the segment, if it was attached to a network enforcer. In an environment where a network enforcer was attached to Amber, an IP that was recorded by Amber would not have been able to make further connections into the segment.

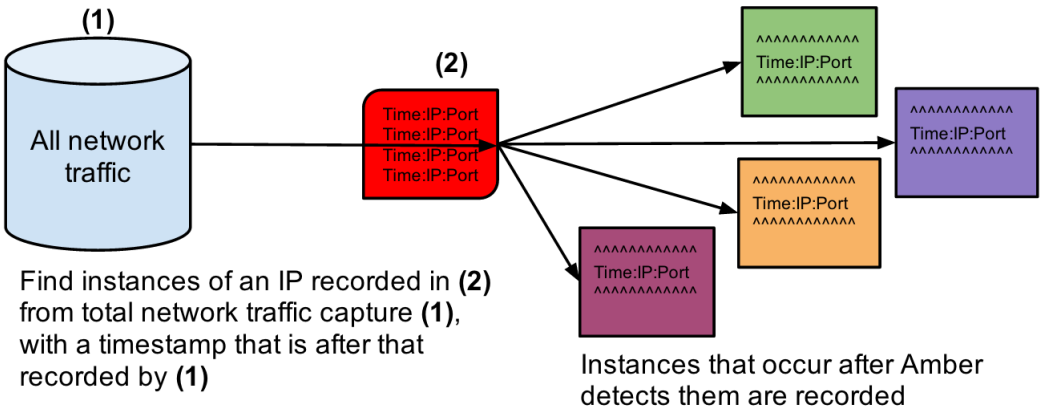

Figure 4.3: Network IP filtering according to Amber

Table 4.1: Node connection statistics for opened ports

Port	**Occurrences**	**Percentage**
135/tcp	50	8.45%
445/tcp	199	33.61%
3389/tcp	343	57.94%

Each source IP address that Amber validated and passed onto the packet sniffer was identified, on average, as accessing the rest of the segment 6.89 times. This number is calculated by counting the number of occurrences that an IP address is found to have connected to another host on the network, after Amber recorded a connection from that IP address. These numbers were then averaged to produce the 6.98 times result. Therefore, Amber was able to improve the segment security posture, as measured by the method described above. Of the ports that the listening daemon opened, the most popular was port 3389/tcp, or Microsoft Terminal Services, which saw 1.7 times more traffic than the next largest destination port as per Table 4.1.

Port 445/tcp has been associated with one of the most reliably exploitable vulnerabilities in the Windows operating system, MS08-067 (Micosoft, 2008), which was used by the Conficker worm outbreak in 2008, as explored by Fitzgibbon and Wood (2009). The Conficker worm (specifically the Conficker C variant) was a computer virus outbreak that was first discovered in November of 2008, and is known for the speed and veracity at which hosts were infected. One of the main components that Conficker C used to propagate was

the MS08-067 vulnerability because once a particular machine was infected, it tried to infect any adjacent machines. According to Porras, Saidi, and Yegneswaran (2009) it did this by scanning for hosts that were listening on TCP port 445. This has created a substantial increase in the volume of noise on the internet as detailed by Irwin (2013a,b) in the form of infected machines scanning for new potential hosts, by trying to establish connections to TCP/IP port 445.

The significant proportion of observed Windows Remote Desktop Protocol (TCP/IP 3389) traffic is interesting because historically this service has not been listed as a top noise generator as shown by Francois, Festor, et al. (2009). This can be attributed to it being difficult to weaponise the Remote Desktop Protocol. In the past there has only been one mainstream worm that has made use of the RDP protocol. As noted by Vizváry and Vykopal (2013), the Morto worm did employ the Remote Desktop Protocol as it attempted to exploit users, but it did so by checking for weak passwords by brute forcing username and password combinations. In order for a service exploit to be successfully deployed by a worm, the exploit needs to leverage predictability and allow for remote code execution because a worm relies on scripted propagation mechanics (Manna, Chen, and Ranka, 2010; Wei, Hussain, Mirkovic, and Ko, 2010). MS08-067 falls into this category, but according to the National Vulnerbaility Database (2013), which is maintained by the National Institute of Standards and Technology (NIST), there has not been a Remote Desktop Protocol exploit that allowed for remote code execution until March of 2012, when MS12-020 (Microsoft, 2012)or CVE-2012-002 was added. MS12-020 changed this and for the first time there was such a theoretical exploit, albeit one that had a low probability of successfully being leveraged as shown by Rapid7 researcher Sherwyn (2012). As of January 2013 there is still no proof of concept that is freely available and able to exploit MS12-020 without causing the server to crash into a non-responsive 'blue screen'.

The test data collected by the Amber node reveals though that the occurrence of Remote Desktop Protocol connections that were captured persisted throughout the test. This could mean that there is a working remote code execution proof of concept for MS12-020 that has not been made public, and that it is being used by a worm. This is consistent with other internet noise-related studies by Pang, Yegneswaran, Barford, Paxson, and Peterson (2004);Irwin (2013b). There is also a second possible explanation for the increase of TCP/IP port 3389 noise on the internet. Soon after MS12-020 was made public,

Kaminsky (2012) published the results of a scan against 300 million internet-connected hosts on TCP/IP port 3389. The purpose of the scan was to estimate the potential exposure of MS12-020 (Caldwell, 2012). Researchers have had difficulties building the remote code execution proof of concept for MS12-020, so a great deal of uncertainty exists around the exploit. This could lead more researchers to focus their attention on the exploit or the possible implications if the exploit was weaponised. The continuous Remote Desktop Protocol scans could be related to this sort of research; research that is being conducted into what the potential exposure of a successful MS12-020 remote code execution proof of concept would be.

4.3 Test Results: Expanding to Geographically Dispersed Nodes

Compared to the pilot test capture settings which used TCP/IP port 135, 445 and 3389, the inclusion of the remote nodes also added TCP/IP port 80 and 443 to the listening daemon. This decision was made to maximise the amount of data that the remote nodes collected by increasing the listening surface area, as the purpose of this test was to establish a link between the different nodes and the source IP addresses that accessed them. Any form of attribution had to be conducted on a country basis, as the translation from source IP address to location is not accurate to a specific location as depicted by Poese, Uhlig, Kaafar, Donnet, and Gueye (2011). Translating from source IP address to geo-location mostly only returned the capital city and country from which the IP originated. To facilitate a process of being able to draw a conclusion from the data, the dataset was normalised to only report the country that it belonged to. This means that all conclusions are drawn on country of attribution, but if more data was available it was stored and used to plot the geographical dispersion figures and maps.

4.3.1 Amber Node: South Africa

Over the nine months of data capture, the South African Amber node captured 2 868 IP addresses, while listening on five of the ports, which received traffic portions according to Table 4.2. The most popular TCP/IP port was TCP/IP 3389, or Microsoft's Remote

Table 4.2: South African node

Rank	Port	Occurrences	Percentage of total
1	3389	1436	50.07%
2	80	741	25.84%
3	445	414	14.43%
4	443	184	6.42%
5	135	93	3.24%
	Total	2868	100.00%

Table 4.3: Top 10 attributed countries for South African node

Rank	Country	Occurrence	Percentage
10	United Kingdom	51	3.81%
9	Korea, Republic of	51	3.81%
8	Spain	57	4.25%
7	South Africa	68	5.07%
6	Russian Federation	69	5.1%
5	Turkey	72	5.37%
4	Germany	122	9.10%
3	Brazil	149	11.12%
2	Argentina	241	17.98%
1	China	460	34.33%
	Total	1340	100.00%

Desktop Protocol (RDP), making up 50.07% of the traffic. TCP/IP port 80 was the second most popular port used by connections that were captured by the node, but was almost half of the total amount of TCP/IP port 3389 captured. MS08-067's spreading mechanism, TCP/IP port 445 was still relatively popular, and accounted for 14.43% of the traffic, making it the third most popular port.

Focusing on the country source of the connections that were captured, 104 distinct regions were recorded as being hosts for one or more connection. The data is skewed towards a handful of regions which made up the majority of the recorded connections. Similar work done by Yegneswaran et al. (2003) also showed this as 71% of their most frequent source IP addresses were traced back to one region, the United States. In the case of the South African node, the top 10 countries are shown in Table 4.3.

The Top 10 slice of the dataset contained 1 340 observations, which is 76.73% of the total

dataset. Of that subset, China accounted for 34.33% of the Top 10 data, followed by Argentina, Brazil and Germany. Table 4.4 shows the traffic that originated from China expanded into the five TCP/IP ports. China's traffic does align to what was observed in the total traffic, whereby TCP/IP port 3389 is the majority of the traffic by a large margin, followed by TCP/IP port 80. This is not the same compared to the second highest source region, Argentina, where 91.75% of traffic attributed to this region targeted TCP/IP port 80, also shown in Table 4.4. The data collected from connections that originated from Brazil had a bias towards TCP/IP port 80 and 3389 equally as depicted in Table 4.4. Traffic on those two ports accounted for 87.92% of the total traffic, while German attributed data was again in line with that seen from China, favouring TCP/IP port 3389 but to a much lesser extent.

4.3.2 Amber Node: Germany

Over the nine months of data capture, the German Amber node captured 1 155 IP addresses, while listening on five of the TCP/IP ports. The most popular TCP/IP port was different to that of the South African node, favouring TCP/IP port 445 which is MS08-067's propagation medium. TCP/IP port 445 constituted 47.88% of the total traffic analysed by the node, followed closely by TCP/IP port 3389 at 26.93% as shown in Table 4.5.

Focusing on the regional source of the connections that were captured, 88 distinct regions were recorded as being hosts for one or more connection, slightly higher than the South African node but not significantly. The data is skewed towards a handful of regions which made up the majority of the recorded connections. The Top 10 countries in the case of the German node are shown in Table 4.6.

The Top 10 slice of the dataset contained 520 observations, which is 45.02% of the total dataset. Of that subset, as with the South African node, China accounted for the single highest region source. 24.04% of the Top 10 data originated from China, tapering off to Germany, the Russian Federation, India and Brazil. Table 4.7 expands the traffic that originated from China into the five TCP/IP ports. The China-based traffic that was captured by the German node is quite different from that recorded by the South African

Table 4.4: Port profile for attributed connections to SA node

	China			**Argentina**			**Brazil**			**Germany**			Total Per Port
Port	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	
3389	326	70.87	66.26	12	4.98	2.44	68	45.64	13.82	86	70.49	17.48	492
80	81	17.61	20.88	222	92.12	57.22	63	42.28	16.24	22	18.03	5.67	388
135	22	4.78	78.57	-	-	-	4	2.68	14.29	2	1.64	7.14	28
445	16	3.48	41.03	6	2.49	15.38	13	8.72	33.33	4	3.28	10.26	39
443	15	3.26	60.00	1	0.41	4.00	1	0.67	4.00	8	6.56	32.00	25
Total	460	100.00	47.33	241	100.00	24.79	149	100.00	15.33	122	100.00	12.55	972
Sum % of Total (1340)	34.3			18.0			11.1			9.1			1340

Table 4.5: German node

Rank	Port	Occurrences	Percentage of total
1	445	553	47.88%
2	3389	311	26.93%
3	80	157	13.59%
4	135	94	8.14%
5	443	40	3.46%
	Total	1155	100.00%

Table 4.6: Top 10 attributed countries for German node

Rank	Country	Occurrence	Percentage
10	United Kingdom	26	5.00%
9	Turkey	30	5.77%
8	Netherlands	31	5.96%
7	Spain	35	6.73%
6	Taiwan	37	7.11%
5	Brazil	39	7.50%
4	India	53	10.19%
3	Russian Federation	59	11.35%
2	Germany	85	16.35%
1	China	125	24.04%
	Total	520	100.00%

node. In this set, connections originating from China had a equal bias towards TCP/IP port 3389 and 445. Germany was the second highest regional source of connections to the German node, detailed in Table 4.7. Interestingly enough it did also not follow the same connection profile that was recorded by the South African node, heavily favouring TCP/IP port 135 opposed to 3389 or 445. TCP/IP port 135, or Remote Procedure Call (RPC), is another common worm propagation port that was very successfully used by the Blaster worm in 2003 (Bailey, Cooke, Jahanian, Watson, and Nazario, 2005). RPC accounted for 40% of German connections. Connections that originated from the Russian Federation favoured both TCP/IP port 445 and TCP/IP port 3389, but with a bias towards port 445. The other ports that were monitored hardly recorded any results at all, as shown in Table 4.7. Connections that were recorded as originating from India, show a definite bias towards TCP/IP port 445, hardly recording any other ports whatsoever. The connections from Brazil also favoured TCP/IP port 445, making up 53.85% of the connection destination ports, but there was also a large portion of TCP/IP port 3389 recorded.

4.3.3 Amber Node: United States

Over the nine months of data capture, the United States Amber node captured the most observations, recording 10 695 IP addresses, whilst listening on the five TCP/IP ports. The most popular TCP/IP port was overwhelmingly TCP/IP port 445. Connections trying to access Microsoft's Remote Procedure Call (RPC) on port 445 accounted for 89.70% of the total traffic. This was followed by TCP/IP port 135, which had made up a minor portion of the traffic on the other three nodes.

Focusing on the regional source of the connections that were captured, 140 distinct regions were recorded as being hosts for one or more connections, significantly higher than both the South African and German nodes. As before, the data is skewed towards a handful of regions which made up the majority of the recorded connections, but where the other nodes had one region that dominated the distribution of connections, the United States node recorded three. Taiwan, the United States and Romania accounted for 45.42% of the top 10 traffic sources, as shown in Table 4.8.

Table 4.7: Port profile for attributed connections to German node

	China			**Germany**			**Russian Federation**			**India**			**Brazil**			Total Per Port
Port	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	
3389	64	51.20	53.78	16	19.75	13.45	20	33.90	16.81	5	9.43	4.20	14	35.90	11.76	119
445	50	40.00	32.26	6	7.41	3.87	32	54.24	20.65	46	86.79	29.68	21	53.85	13.55	155
80	9	7.20	26.47	19	23.46	55.88	3	5.08	8.82	2	3.77	5.88	1	2.56	2.94	34
135	1	0.80	2.56	34	41.98	87.18	1	1.69	2.56	-	-	-	3	7.69	7.69	39
443	1	0.80	10.00	6	7.41	60.00	3	5.08	30.00	-	-	-	-	-	-	10
Total	125	100.00	35.01	81	100.00	22.69	59	100.00	16.53	53	100.00	14.85	39	100.00	10.92	357
Sum % of Total (520)	24.0			15.6			11.3			10.2			7.5			520

Table 4.8: Top 10 attributed countries for the United States node

Rank	Country	Occurrence	Percentage
10	Venezuela	308	6.16%
9	Bulgaria	311	6.22%
8	Italy	313	6.25%
7	Japan	397	7.94%
6	China	422	8.44%
5	India	463	9.26%
4	Brazil	515	10.3%
3	Romania	694	13.87%
2	United States	776	15.52%
1	Taiwan	801	16.01%
	Total	5000	100.00%

The Top 10 slice of the dataset contained 5 000 observations, which is 46.75% of the total dataset. While Taiwan, the United States and Romania constituted the majority of the Top 10, they also made up a large portion of the total set. The combined connection total for the three regions equalled 21.23% of the total set of source IP addresses collected. Looking closely at each of the countries also revealed instances that were unique to this region, chiefly that the top attributed region caused a huge skew in the data. Table 4.9 shows that source IP addresses that connected from Taiwan and used TCP/IP port 445 accounted for 88.51%. The United States was the second highest source of traffic that the US node recorded, and as with the Taiwanese region, it also consisted mostly of TCP/IP port 445 connections. The distribution is not as aggressively skewed towards TCP/IP port 445 such as in the case of Taiwan, but it is very close accounting for 84.41% of the total connections. This trend continues through the top three source regions. The data that was captured by the US node and that was attributed back to Romania also showed a strong bias towards TCP/IP port 445. As Table 4.9 also shows 95.37% of the connections requested Microsoft's RPC protocol. The only other port recorded from Romania was TCP/IP port 135. The prevalence of TCP/IP port 445 is not limited to the top three regions, and can actually been seen across the top five regions, the last of which are Brazil and India. Both of these countries recorded distributions that are above 95% (97.86% for Brazil and 98.70% for India).

This trend can be seen in Table 4.10 when the total port distribution for the US node is

Table 4.9: Port profile for attributed connections to US node

	Taiwan			United States			Romania			Brazil			India			Total Per Port
Port	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	Conns	% of region total	% of port total	
445	709	88.51	27.84	655	84.41	25.72	679	97.84	26.66	504	97.86	19.79	457	98.70	17.94	2547
135	91	11.36	51.70	69	8.89	39.20	15	2.16	8.52	1	0.19	0.57	3	0.65	1.70	176
3389	1	0.12	2.13	37	4.77	78.72	-	-	-	8	1.55	17.02	1	0.22	2.13	47
443	-	-	-	2	0.26	100.00	-	-	-	-	-	-	-	-	-	2
80	-	-	-	13	1.68	76.47	-	-	-	2	0.39	11.76	2	0.43	11.76	17
Total	801	100.00	24.65	776	100.00	23.88	694	100.00	21.36	515	100.00	15.85	463	100.00	14.25	3249
Sum % of Total (10695)	7.5			7.3			6.5			4.8			4.3			10695

Table 4.10: United States node

Rank	Port	Occurrences	Percentage of total
1	445	9593	89.70%
2	135	877	8.20%
3	3389	186	1.74%
4	80	36	0.37%
5	443	3	0.02%
	Total	10695	100.00%

examined because 89.70% of all the data collected was attributed to TCP/IP port 445. No other node experienced such a high level of distribution bias.

4.4 Visualising Source IP addresses

The previous section made use of certain techniques, as shown by Padmanabhan and Subramanian (2001), that map an IP address to an estimated geolocation. The process is helpful because it can convert seemingly random IP address sequences into common locations, aiding analysis. In the previous section this made it possible to group the recorded attacks back to countries. The number of attacking countries for each node exceeded 100 entries which makes it difficult to view the relative attack propensity of the less prominent countries. To address this shortcoming each node's list of the source IP addresses was converted into their corresponding geolocation via the MaxMind (2013) GeoIP database. This transformed an IP address into a country, and depending on the data that was available, some of the IP addresses were translated into city and country. The data was then grouped together into a list of named locations that had tried to connect to the Amber instance, instead of IP addresses, which allowed for the above analysis to occur.

According to Kress and Van Leeuwen (2006) people relate better to visual representations than lists so it was decided that a visual representation of the data would overcome the difficulties of presenting a list of over 100 countries. A method of plotting the source locations on a map was chosen because it overlayed unknown information (the source countries) onto a piece of well-known information (a map of the world). Google has

```
        var map;
        function initialize() {
            var myOptions = {
                zoom: 2,
                disableDefaultUI: true,
                mapTypeId: google.maps.MapTypeId.HYBRID
            };
        var all = [
//BEGIN
["1", "Rondon, Brazil", "-23.474567", "-52.845022"],
["1", "Vaals, Netherlands", "50.770833", "6.018056"],
["1", "Dumfries, United States", "38.567620", "-77.328038"],
["92", "Russian Federation", "60.000000", "100.000000"],
["1", "Surgut, Russian Federation", "61.250000", "73.416667"],
["1", "Belgorod, Russian Federation", "50.610743", "36.580152"],
["1", "Toplita, Romania", "45.668967", "22.773993"],
["2", "Port-of-spain, Trinidad and Tobago", "10.666175", "-61.516571"],
["2", "Brno, Czech Republic", "49.195223", "16.607959"],
["1", "Utsunomiya, Japan", "36.565834", "139.883611"],
["3", "Belgrade, Serbia", "44.804006", "20.465126"],
["2", "Botosani, Romania", "47.750000", "26.666667"],
["1", "Gyoda, Japan", "36.133333", "139.450000"],
```

Figure 4.4: Google Maps API Javascript array

a simple mapping function that utilises the maps.google.com service to add markers at certain locations, which are parsed to the maps.google.com API via Javascript. The API does not accept named locations, instead requiring GPS coordinates to place markers. This meant that the list of source countries needed to be converted into a list of GPS coordinates before they could be parsed to the maps.google.com API. Geopy[3] is a Python library writen by Exogen (2010) that is able to do this mapping, by querying an online database for each IP address that it is parsed. As noted earlier, converting from an IP address to a location is not an exact process (Padmanabhan and Subramanian, 2001), and the same is true for converting from named location to GPS coordinates. This is so because a named location such as China, can have millions of GPS coordinates, which is also true even at the city level. Therefore when a location is parsed to the geopy library it returns an array of possible GPS coordinates which could be the named location. The assumption was made that the first coordinate pair would be used.

The list of coordinates was then taken and transformed into a Javascript array which the maps.google.com API understood, as shown in Figure 4.4. The file is constructed in

[3]https://code.google.com/p/geopy/

```
if (all[i][0]>=40) {
  var image = {
    url: 'images/red80.png'
  };
}
if (all[i][0]<40) {
  var image = {
    url: 'images/red40.png'
  };
}
if (all[i][0]<20) {
  var image = {
    url: 'images/red20.png'
  };
}
if (all[i][0]<6) {
  var image = {
    url: 'images/red10.png'
  };
}
```

Figure 4.5: Marker size scales with count variable

HTML, which calls the API when it is rendered by a browser. While this method did work, the maps.google.com API is not able to group multiple markers together if they fall on the same coordinate. This meant that a single marker could indicate one, ten or a thousand instances of attack. To remedy this some logic was built into the Javascript that rendered the HTML. Once the array was built, it would search for duplicate coordinates and keep track of how many times a duplicate occurred. The API allows for a custom image to be used as a marker instead of the default one, and this functionality was used to draw attention to coordinate pairs that represented a large amount of duplicates.

Figure 4.5 shows the use of custom images to denote a marker on the map that has more than 40 duplicates, less than 40 duplicates, less than 20 duplicates or less than 6 duplicates. The images that are assigned to each of those categories increase in size to highlight the increased amount of duplicates, from 10x10, 20x20, 40x40 and 80x80 pixels. A sample of the markers is shown in Figure 4.6.

The final HTML rendering, Figure 4.7, shows a world map and overlayed markers on areas of high attack that were recorded by the South African Amber node. From this view, certain quick conclusions can be made such as North Africa was not a large contributor

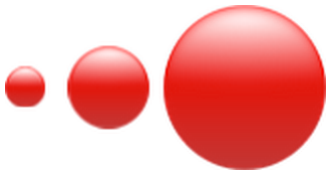

Figure 4.6: Marker sizes

Figure 4.7: World map visualisation of the SA node's captured data

to the threats that the node recorded. There is some infrastructural logic around this as North Africa and other third world regions are not endowed with personal computers and fast internet connections as stated by the World Bank Development Data Group (2012). One that is not that logical is Australia, which is an advanced country, but which caused few of the connections seen by the South African node. The United States, Europe and certain parts of Asia constituted the largest amount of attacks, which is expected based on the table summaries done earlier.

What the visual representation does allow is for a high resolution view of the attacking country, showing which regions in the country were more likely to be an attacker. By redrawing the world map in Figure 4.7 to only include a particular region such as Europe,

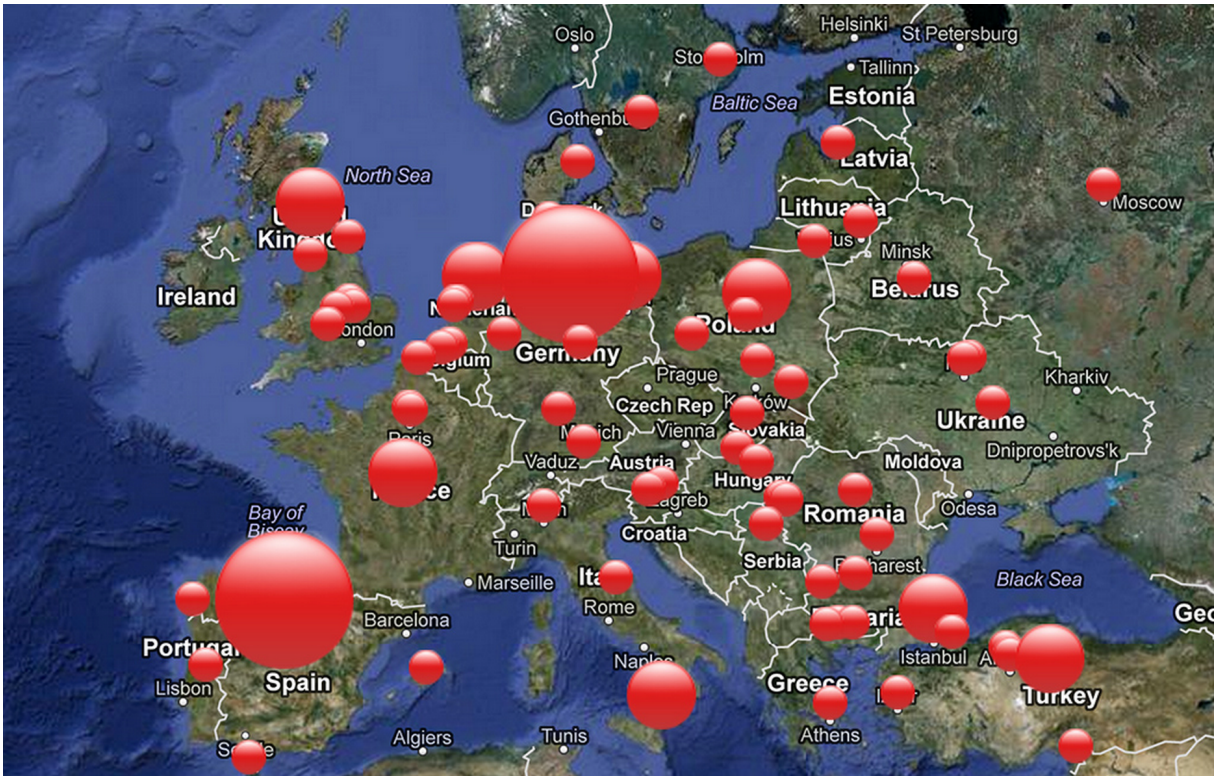

Figure 4.8: European map visualisation of the SA node's captured data

as per Figure 4.8 the map's increased resolution shows what areas in that region were particularly malicious. It shows that Germany and Spain were the greatest single sources in Europe.

A visual representation of each of the other nodes is also possible. Figure 4.9 shows the German node's map, which is particularly interesting because it quickly relays the difference in the amount of African-based attacks when compared to the South African node. The South African node experienced an increased number of attacks and from more African nations than the German node, which could indicate there is a high propensity to be targeted by regions that share the similar infrastructure such as undersea cables.

This finding is further supported by increasing the resolution of the German node's world map to show the same European region as Figure 4.8. The amount of German originating traffic as depicted in Figure 4.10 is significantly more than that of the South African node.

A world map generated by the United States node's attacker list surcomes to the same

Figure 4.9: World map visualisation of the German node's captured data

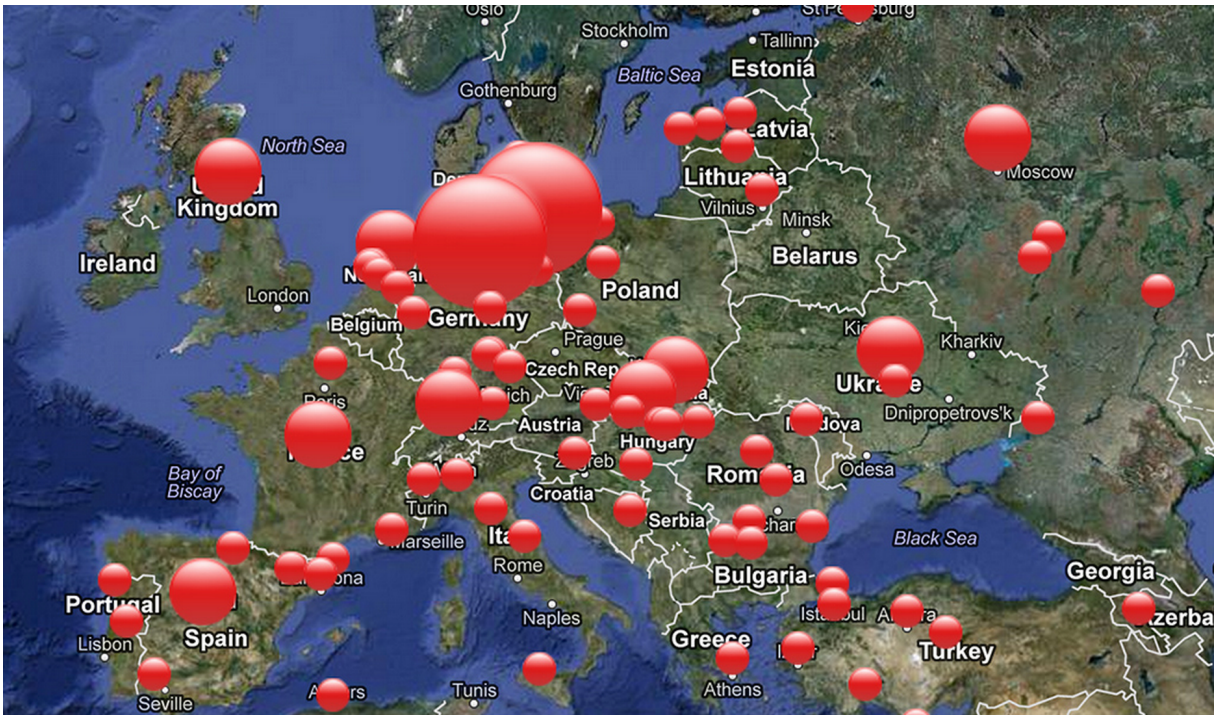

Figure 4.10: European map visualisation of the German node's captured data

Figure 4.11: World map visualisation of the US node's captured data

problem as the extensive list originally recorded. It is difficult to visually discern between regions in figure 4.11 because the markers cover most of the map's surface. What is distinguishable is that there is still a lower amount of attackers originating from the African region when compared to the rest of the attacker map. A visual investigation also shows that there were a greater number of attacks originating from Europe or Asia than from the United States. This is inconclusive because the size of the markers in Europe are so large that they obstruct meaningful analysis at this view.

Focusing on the European region of the United States world map in Figure 4.12, and comparing that to the United States from the world map, there is a clear sense of distribution throughout the region. Hungary, Slovakia, Romania and Bulgaria showed concentrations on certain locations as well as a distributed scattering of markers in that country.

These collections of maps show a snapshot of the historic threat landscape in terms of malicious connections that have been attributed to certain countries. Previous discussions have mentioned the shortfall in the attribution process of converting IP address to regions, which has the potential to nullify any form of analysis from this visual standpoint, but the changes in the threat map can be analysed. A change in the map, for example if there is a drastic increase in the amount of malicious connections being observed from one of

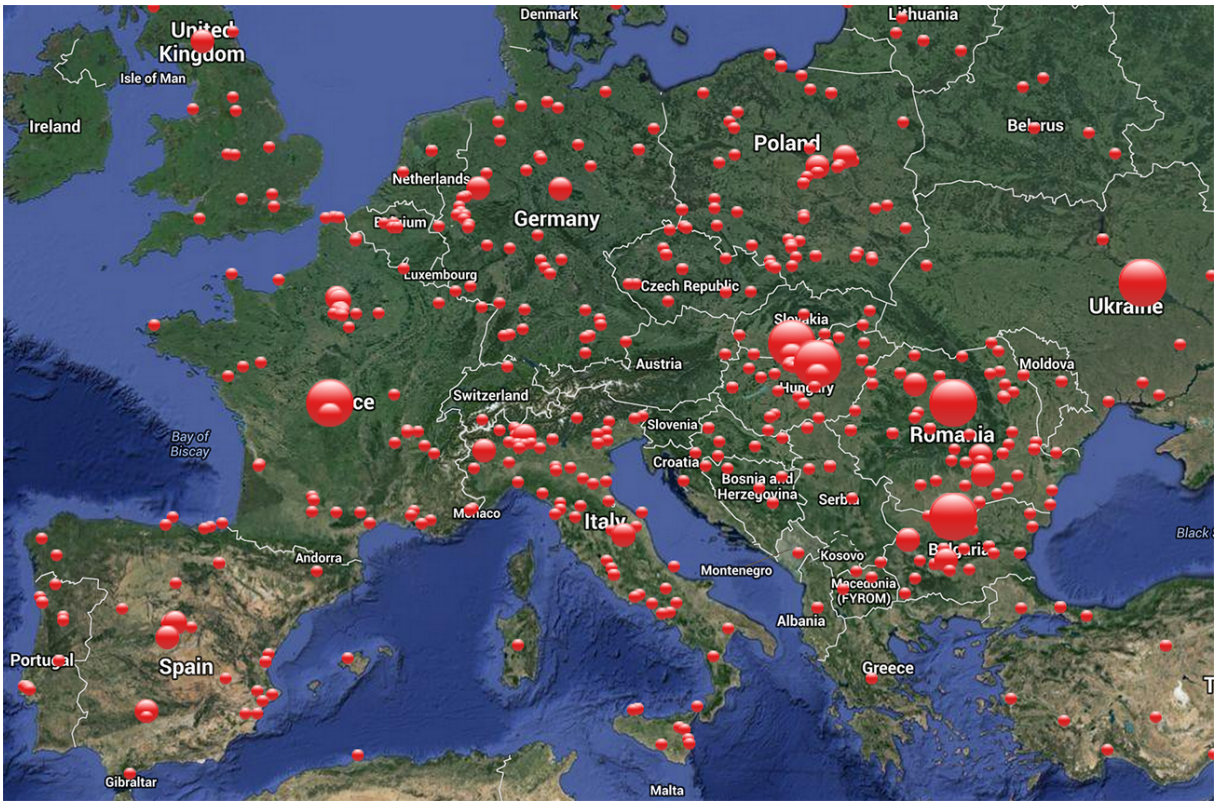

Figure 4.12: European map visualisation of the US node's captured data

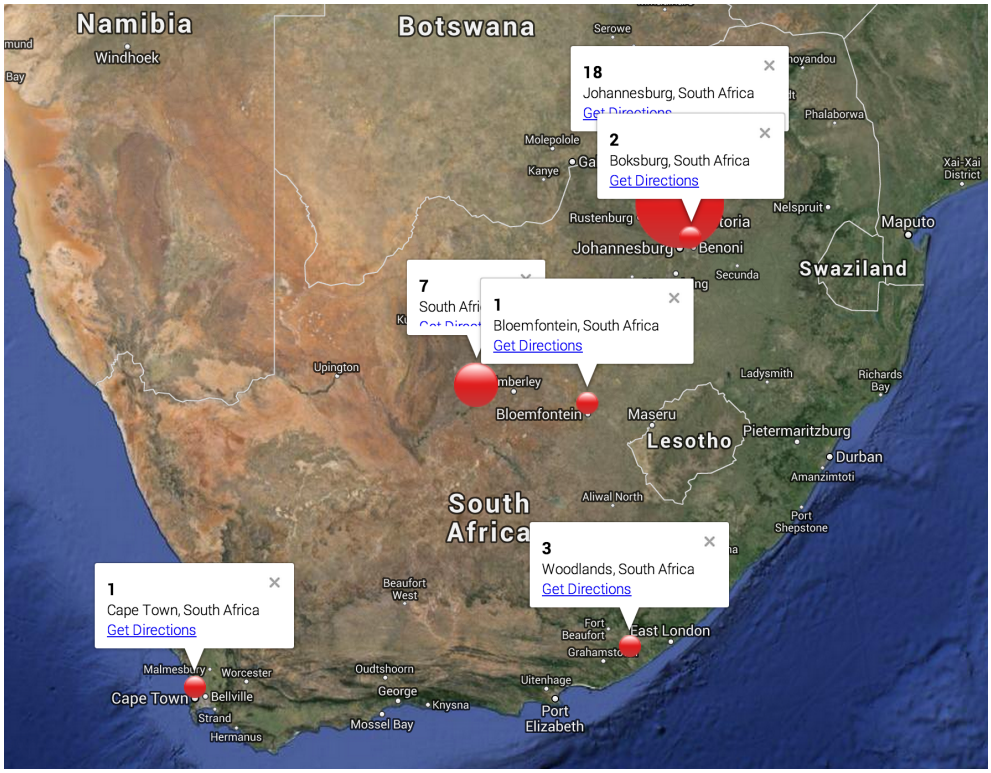

Figure 4.13: South African map visualisation of the ZA node's captured data

the less significant regions, such as the Democratic Republic of Congo, could be worth investigating. The specific profile of these connections could lead to a new threat that has emerged and is still publically unknown, or if there is a targeted attack underway against the network from that region.

The German and United States nodes reported very little atrribution towards South Africa, epecially comapred to the South African node, which shows a large concentration of connections on the world map view in Figure 4.7. Zooming into the South African portion, as shown in Figure 4.13 highlights that the majority of the connections were attributed to Johannesburg, which is also were the server was hosted. Based on this it would seem that a system faces a unique set of threats from systems that are phyiscally closer to it.

Table 4.11: Length of threat streams per Amber node

	South Africa	United States	Germany
South Africa	-	-	-
United States	15 (0.11%)	-	-
Germany	25 (0.62%)	43 (0.36%)	-
IP's Collected	2868	10695	1155
IP's linked to all three nodes: 4 (0.027%)			

4.5 Gains Through Distributed Intelligence

Testing the relevance of each geographically different threat stream on the data that was captured by the local, South African segment, would be achieved by looking for any common source IP addresses that were seen by the United States or German nodes and the South African node. As stated above, the three nodes measured the following successful connections as shown in Table 4.11. Each source IP address, from each of the nodes was compared to the other two nodes' IP address streams. A match would indicate that a particular IP address was discovered by both nodes, and showed that there is a correlation between the nodes.

The data in Table 4.11 shows that there was a very small set of IP addresses that were logged by two or more nodes. The German and United States based nodes logged 43 common source IP addresses, while the South African node only logged 25 IP addresses that also attempted a connection with the German node. The US and South African nodes recorded the lowest intersect of source IP addresses, finding only 15 IP addresses that were common between them. Only 4 IP addresses were common between all three of the nodes. Based on these findings it would seem that there is low value derived from distributing Amber nodes in geographically different locations, however longer sampling would be required to confirm.

While these results are on a minor scale, it does invite the question as to whether or not there is any value in applying threat countermeasures in one region based on intelligence collected in a different region. There are numerous commercial products that offer this service (Alme and Eardly, 2010) and while it was not possible to test this at the same scale as the large security vendors, it would seem that the ability of a small threat stream to

be useful outside of its region is extremely limited. As a substitute, the information that was gathered from the Amber nodes was cross-checked against the OpenBL.org project's 360-list day. OpenBL.org[4] is a project that records incidents of abuse from IP addresses for the purpose of blacklisting. The project tracks instances where source IP addresses attempted to brute force a service that was hosted by the OpenBL.org project. These IPs are stored in list files that span time periods of 30, 60, 90, 180 and 360 days, as well as meta lists that are service specific. Their 360-day list was the closest match to the time period in which the region Amber nodes were capturing source IP addresses. At the time of writing the OpenBL 360-day list contained 16 472 IP address, which when compared to the 14 718 IP address that the Amber nodes had collected, resulted in no common addresses. This further supports the very limited use of applying third party blacklists as a security control.

4.5.1 Weighted Scoring

According to the previous section's analysis, there is little value in building a distributed threat model that relies on only the IP addresses that it identifies as malicious, but if the data is categorised into countries, the way in which it can be analysed is changed. Looking at the data on a country level shows that certain countries are more prone to be attackers than others as certain regions seem to appear on more than one of the node's Top 10 attack lists.

We can determine which country is responsible for the most malicious traffic by building a weighting system by assigning each country in the Top 10 list a number between 1 and 10 depending on if they are the at the bottom or at the top of the list. Only the Top 10 countries were taken into consideration because it lessens the influence of outliers. These scores are then added up for all three lists, and displayed in Table 4.12. For example, Romania appears in position eight on the United States node's list where ten is the most dangerous country, but Romania does not appear on any of the other node's lists. Therefore it has a score of 8 (8+0+0=8). The Russian Federation on the other hand appears on two lists: spot eight on the German list and spot five on the United States list. This produces a score of thirteen (8+5+0=13).

[4]http://www.openbl.org

Table 4.12: Threats identified by the weighted scoring method (country and continent)

Country	**Score**	**Continent**
Venezuela	1	South America
United Kingdom	2	Europe
Bulgaria	2	Europe
Korea, Republic of	2	Asia
Italy	3	Europe
Netherlands	3	Europe
Japan	4	Asia
South Africa	4	Africa
Spain	7	Europe
Romania	8	Europe
Turkey	8	Europe
United States	9	North America
Argentina	9	South America
Russian Federation	13	Asia
India	13	Asia
Taiwan	15	Asia
Germany	16	Europe
Brazil	21	South America
China	25	Asia

Using the weighted list method, Table 4.12 was constructed, which highlights that China is the most prevalent region amongst the three nodes, scoring a total of 25. Based on this, China is the most dangerous region to which connections can be attributed. This is not a surprise since China has been labelled as a high source of malicious traffic by the press and industry experts ever since a report by Mandiant Central Intelligence, MCI (2013); Fidler (2013) claimed that their military was responsible for cyber attacks.

The weighted scoring system also highlights Brazil, Germany and Taiwan as regions that hosted a high number of malicious traffic sources, which is less obvious than China's inclusion in the list. This information is useful as it could be used to increase the weight of intrusion detection system signatures that match on a possible intrusion, that is also being launched from a region that has a high score. This is illustrated in the following example: A Snort IDS is placed in front of a Microsoft Windows server, which is configured with a signature that alerts when brute force login attempts are detected against the Remote Desktop Protocol. The signature does this by triggering an alert on five failed login attempts. It is plausible that a legitimate user could mistype his password five times, creating a false positive incident, but by also looking at the source of the connection and matching that against the weighted list findings these false positives can be given less priority. If the failed login attempts are also coming from Brazil, then more attention should be given to that incident than those which originated from the United Kingdom. Grouping the countries into their respective contenint shows that, while the weighted list is equally populated with European and Asian countries, Asia is almost 47% more dangerous as Europe and 700% more dangerous than the North American continent as scored by the sum of weights attributed to each contenint

This adds another dimension to incident evaluation as it offers signature evaluation systems a method of prioritising incidents that are attributed to countries that are more dangerous than others.

4.5.2 Profiling Countries

When factoring the source IP addresses in their respective originating regions, another avenue of potential threat analysis arises. Another use for distributed threat intelligence

outside of simply blacklisting source IP addresses is to template the different source regions. As mentioned above certain countries had a bias towards a certain protocol, and that bias was recorded by multiple nodes. Taiwan is the strongest example of this. The United States node listed Taiwan as hosting the most malicious connections that attempted to connect to it, and that those connections consisted mostly of TCP/IP port 445. Looking at the German node, Taiwan was not as prevalent on the Top 10 list when compared to the United States node (sixth position opposed to first), but the German node recorded that of the connections that originated from Taiwan 89.19% where TCP/IP port 445. The South African node tells a different story though. Taiwan didn't make its Top 10 list of malicious origins, and was ranked 14th, and also did not display a strong bias towards any particular port. The connections from Taiwan as recorded by the South African node was split between TCP/IP port 3389 (48.89%) and TCP/IP port 445 (37.78%).

This discovery altered the original hypothesis, in that it should only be applied to countries that showed a strong bias towards a certain TCP/IP port. By focusing on the countries that showed a port bias, the indicators generated by templating are only based on strong signals. This new hypothesis fitted the Taiwan example, where there was a bias for TCP/IP port 445 recorded by the United States and German node, but not in South Africa where there was no bias. It also fits with Brazilian originated traffic. Brazil appeared on the Top 10 list of all three nodes, but the TCP/IP port profiling showed that connections that originated from Brazil had no correlation between nodes. The German nodes profiled the connections originating from Brazil as having a minor bias towards TCP/IP port 445 (53.85%) with a strong complement of TCP/IP port 3389 (35.90%). The United States node instead showed an extremely strong correlation to TCP/IP port 445 with 97.86% of the connections using it, but the South African node saw an equal split between TCP/IP port 3389 (45.64%) and TCP/IP port 80 (42.28%).

This was not the case when the hypothesis was tested against the rest of the data set. China, the largest connection originator of the data set did not fit this hypothesis because all three nodes had recorded connection profiles that were all biased towards a certain port, but none of which overlapped. The South African node recorded a bias for TCP/IP port 3389 (70.87%) while the German node profiled connections from China as favoring TCP/IP port 3389 (51.20%) and TCP/IP port 445 (40.00%). The United States node on

the other hand recorded an overwhelming bias for TCP/IP port 445 (83.89%). This does not fit into the hypothesis of profiling country connections. China, Brazil and Taiwan are also the only countries that appear on all three of the node's Top 10 lists, which means that they are the only countries on to which this kind of analysis can easily be applied to, as those countries that to not appear on the Top 10 list did not generate enough connections from which to draw conclusions. The hypothesis of profiling a country's traffic based on bias data collected by the nodes was abandoned because of varying results that were recorded, and because the hypothesis was in danger of becoming tailored to the data set.

4.5.3 Local Bias

Profiling a region's outbound connections might not be possible due to the differences that the nodes recorded, but the fact that an originating region has a bias for a certain destination port is in itself interesting. Network scanners such as Unicorn Scan (Nomura, Watanabe, Tartakowskï, and Six, 2007), Nmap (Wolfgang, 2002; Yarochkin, 2013) and Zmap (Durumeric, Wustrow, and Halderman, 2013) will by default scan a large number of common ports, including those exposed by the three distributed Amber nodes. Because all of the ports would be triggered equally by a default scan, the emergence of a region that is biased towards a certain port when connecting to a node indicates that that port is specifically being targeted by connections that are attributed to that particular region. If an Amber node is deployed onto a segment, connections that it records are already relevant to the segment, which makes it possible to profile an attributed region specifically for the local segment.

For example, the South African node recorded connections from the Russian Federation as having a bias towards TCP/IP port 3389 (75.36%). Since this traffic is relevant to the local segment, this information could be used to improve scoring-based signature systems such as Snort, by increasing the score of all connections from the Russian Federation that try to connect to TCP/IP port 3389, which has become a recent threat as research by Irwin (2013b) has shown. This example overlaps a lot of the same intelligence that the weighted scoring system uncovered as the Russian Federation would have already been seen as a potential for malicious traffic. Because the scoring system is cumulative (additional intelligence increases the priority, and is not lost) this is not necessarily a bad

thing, but it does help identify new threats for countries that the weighted scoring system misses.

The South African node is unique in that the Republic of Korea only appears on its Top 10 connection lists. The weighted scoring system would therefore not add a higher priority to connections from the Republic of Korea. Using the Local Bias method, we see that connections from the Republic of Korea were biased towards TCP/IP port 3389. This intelligence can be applied to other security controls which make use of signatures, supplementing the scores of those signatures. Such a system could be linked to a local deployment of an Amber node which can update signature priority based on regional attribution and ports. For example, the priority of TCP/IP port 3389 connections from the Republic of Korea can be increased, focusing the time of expensive resources such as human security analysts on incidents that have a high potential of being malicious.

4.5.4 Combined Data Analysis

Earlier it was shown that there is little value in employing a remote node's IP list as a blacklist, but that value could be realised if the data was converted into regions and then analysed. This process of transforming data from its 'raw' form into data that is different but still related to the original data set allows for new directions to be pursued when analysing data. The ability to view three data sets that were collected in identical ways, but from different sources holds its own value.

This has shown that traffic that was attributed to the same region as a node that collected it had a higher probability of being malicious when compared to the remote nodes observations. The South African node was the only system that recorded a significant amount of South African attributed connects, and the German and United States nodes both recorded more localised traffic than their international counterparts did. The data profile in terms of what ports where attacked, as shown in the previous section, was also different for each region.

Based on this data, locally created security controls might have an advantage over international products when deployed in their country of origination. If the security control

was developed in a local region, then it stands to reason that it was fine tuned to a certain degree, for the threats and attacks that face that particular area. This would give such controls an intrinsic knowledge over the kinds of threats that face those regions. Overall this is inconclusive and requires more targeted investigating.

4.6 Enterprise Deployment

In order for this research to be relevant it should be deployable in an enterprise of varying sizes, and still add value to the overall network. This covers two forms of network layouts: small and complex. Small networks consist of a single segment, while complex networks are made up of many interlinked smaller segments, or small networks.

4.6.1 Small Networks

Amber was originally architected and built on a single segment network, which is the equivalent of a small company network. While the testing deployment required the use of a full packet capture system as well as the Amber node, a production system would only require a single Amber node on the segment. That Amber system could feed the data that it collects directly to the firewall or other security control, allowing the process to be productive while only introducing one system into the environment.

This architecture, shown in Figure 4.14, is the simplest and cheapest to deploy from a resourcing point of view. By deploying one system, the network segment would be able to generate a stream of IP addresses that are relevant to incoming attacks, which can be passed to a network enforcer such as a firewall, IPS or other security control. With this deployment it would also be possible to retrofit the Local Bias signature augmenting method because it does not require the use of geographically distributed nodes to make actionable intelligence. That intelligence would be sent to the IDS and / or IPS where it would be applied onto the control's signatures.

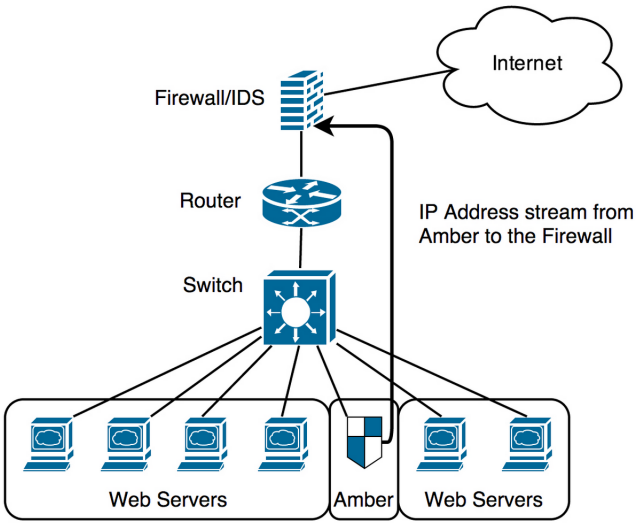

Figure 4.14: Amber architecture for a small network

4.6.2 Complex Networks

Scalability is a common problem faced by security controls, specifically those that are installed inline such as firewalls and IPS devices, because they are designed to inspect all data that enters or exits a network. If the amount of data that is flowing through network increases, then the device needs to be able to process the increased load. This impacts the architecture and design of these security controls when they are deployed in a small medium enterprise (SME) of 200 users, opposed to a large enterprise of 30 000 plus users. This is evident in the price differences of SME firewalls (rated to 100Mb/s) and Enterprise firewalls (10Gb/s plus). Amber does not have these limitations because it is not an inline device, and only needs a presence in a segment. Architecture is then done on a per segment basis, and can be scaled by duplicating the segment deployment to another segment. If a network comprises five segments, then the architecture would be to deploy five Amber nodes, one into each of the segments. One consideration is that once more than one segment is being studied, it would be beneficial to add an Amber command and control server to the architecture. The command and control server would sit in an internal, trusted segment and collect the connection information from each of the Amber nodes as shown in Figure 4.15. It would then combine the information and construct a joint threat stream which would be sent to all of the perimeter enforcers. Combining the threat lists

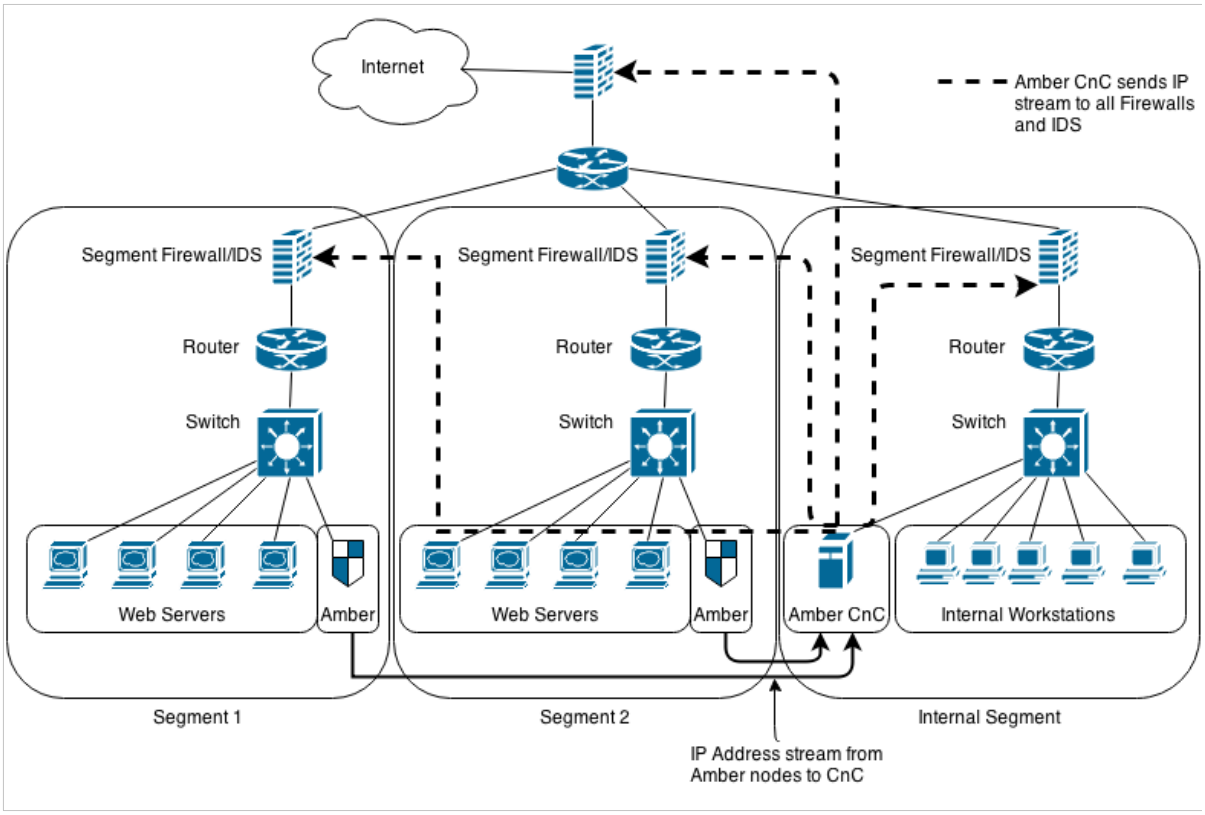

Figure 4.15: Amber architecure for a complex network with CnC server

is possible in this scenario because, and unlike the geographically distributed nodes, the nodes are intrinsically linked by common IP space ownership, and the geographic location of the segments (it is assumed that the segments are all housed in a single data centre).

A full architecture with multiple local nodes, a local command and control server and three or four geographically distributed nodes would still be very efficient in terms of cost and ongoing management/maintenance. The distributed nodes were able to run for months without any intervention due to watchdog scripts, meaning that once a local or geographical node is deployed very little interaction is needed. The command and control server would need more initial work as each node needs to be attached to its inventory. This requires the transfer of SSH keys and whether it is a local or geographical nodes because the type of data that can be pulled from the data stream is different for each. Then the command and control server would need to fetch the up-to-date IP address list

at a fixed interval, which can be optimised for coverage (increased frequency) or minimal network usage (decreased frequency). Analysis can be performed after each data pull or at certain times such as once a day or every 10 minutes depending on resourcing and the amount of IP addresses in the data stream. The actual analysis is fairly light on resource utilization, but the distribution of that intelligence to the network enforcers and intrusion detection systems could introduce congestion. Traditionally firewalls and intrusion detection systems need to reload their rule sets once a change has occurred, which takes time (Acharya et al., 2006). If the analysis resolution is set to high, then a bottleneck could occur at the rule set reload stage. This is very dependent on the particulars of the environment in which Amber is deployed, and would need to be tuned for each environment.

4.6.3 Command and Control

With multiple Amber nodes in separate segments, and an Amber command and control server it would be possible to also generate a weighted scoring list. The Amber nodes would send their data lists as per normal to the command and control server, which would run the weighted scoring analysis and send the results to the appropriate security controls, such as an IDS. It would also be possible to supplement the ecosystem with multiple geographically distributed nodes which report back into the command and control server. The data that the geographic nodes collect would be added to the command and control server and analysed as if it were any other node, except that it would only produce weighted scoring lists from that information.

The command and control server also centralises all the data that the Amber nodes collect, making it possible to generate reports for the entire Amber ecosystem from one location. These reports can take the form of security metrics or applying transforms on the data the Amber nodes collect. Possible transforms could be to do reverse lookups on the connecting IP addresses and analyse any correlation in the domains that are being used, or the method in which the reverse names were constructed. The command and control server forms the basis of these extended capabilities, but it also creates a central point of weakness for the Amber deployment. Because the Amber nodes have exchanged public keys with the command and control server, if an attacker is able to take control

of the central server they will also have full access to each of the Amber nodes and can destroy them by logging into them. This is not that severe because the value of the nodes is the data that it collects and if an attacker does manage to disable one of the Amber nodes little damage will be done to the production environment, as mentioned in section 3.3.2, Denial of Services attacks against an Amber node. What is more worrying is if an attacker starts to corrupt the data that the Amber nodes are feeding into the other security controls. A knowledgeable attacker could edit the data before it is analysed and, feeding information into the ecosystem that would lower the priority of their attack locations. This would lower the likelihood of future attacks being detected. To defend against this attack, the command and control server would be protected in the same way as any other centralised security system (firewall management stations, and antivirus signature deployment servers), such as only allowing certain IP addresses to log into the server and installing a local firewall.

4.7 Summary

This chapter takes the data that was collected by single and distributed Amber nodes and expresses it in terms of lists, showing the Top 10 attributed countries and which ports constituted the largest percentage of traffic. It also visualises the full data set via the maps.google.com API, through which a richer understanding of the countries that do not fall within the Top 10 was attained. The chapter reveals that once the data of the three distributed nodes was combined, very few of the IP addresses were repeat-findings between the nodes, which questions the value of commercial IP blacklists. Data analysis did yield three potential methods of transforming the data that the Amber nodes captured and applying them to signature-based security controls. The intelligence produced by the nodes for the security controls could aid the systems in focusing on incidents that are more likely to be malicious. Two of those methods could be integrated into signature-based systems, such as an IDS/IPS and firewall, while the third was found to be unsuccessful. The chapter ends by detailing how to deploy an Amber ecosystem into an enterprise network. Both simple and complex network environments are explored, as well as the low cost at which such an ecosystem could be deployed. The concept of a central command and control Amber server is expanded on, explaining the advantages

and potential weaknesses of using a central system, as well as ways in which it can be secured. The following chapter identifies possible future enhancements to the Amber ecosystem in terms of additional data transforms and techniques to improve the weighted scoring method by normalising certain aspects of the data as well as other factors that could influence the profile of the connections that an Amber node captures.

5 Conclusion

This research set out to prove that a security model that was built on the comparative strengths of two distinctly different, but related security models, could increase the security posture of a network segment. The two security models, Decision through Detection and Decision through Presence, both aim to improve the security posture of the assets that they are tasked to protect, but each manages to fulfil that custodianship with a different methodology and different associated costs. Despite their differences, each of the security models can be factorized into two distinct phases: the Discovery phase and the Action phase. Each model provides a notable cost advantage in one of the two phases, thus bestowing a differing competitive advantage to both.

In the traditional implementations, DtP is able to assess a threat in the Discovery phase at a relatively low cost, but falls short in the Action phase due to the amount of intensive research that is required. DtD on the other hand requires a large amount of research in the Discovery phase. While the research can be centralised and distributed to an infinite number of nodes, it does also increase the likelihood of false positives being detected as threats. Despite DtD's expensive Discovery phase, it has a superior Action phase as it leverages off the research completed in its Discovery phase. This asked the question of whether it would be possible to combine the Discovery phase of the DtP model and the Action phase of the DtD model, so as to capture the most successful pieces of each model and build a new model which is more efficient. It was decided to show the possibility of combining security models by building a system that would adhere to a hybrid model approach, and would then be empirically tested to discover if it would be able to improve the security posture of a network segment.

Using the DtP security model's Discovery phase and the DtD security model's Action phase methodologies, Amber was able to improve the security posture of the environment that it was connected to, as measured by its ability to detect a source IP that would later try to connect to multiple other hosts on the segment, informing a network enforcer of the IP address. Based on this it would be practical to use the hybrid model to build a new generation of security controls that adhere to the core concepts of the hybrid model; identifying unused electronic space, for which there is no productive use in accessing, and then connect it to an enforcer which controls access to information technology assets. It

was not practical to increase the amount of nodes on a single segment due to each node needing a piece of unused internet space, but it was theorized that multiple, geographically separate nodes could be interconnected through a command and control server. Each of the nodes would then send their individual IP lists to the command and control server, who would then distribute the list to all other nodes.

A preliminary correlation test was devised that tested whether there was any linkage between a source IP addresses logged between the multiple nodes, before a test was done to see if the security posture could be improved by increasing the nodes. The research yielded mixed results, showing that there was almost no linkage between source IP addresses logged by the different nodes. This meant that there was no advantage in extending Amber's source IP address stream to include that of multiple nodes across different geographical regions. This also questions the usefulness of initiatives that attempt to apply threat data from one region onto other regions. A secondary test, the 360-day OpenBlacklist IP list, was also compared to the source IP addresses that were captured by the geographically distributed Amber nodes. As with the previous test, there were no common source IP addresses found, but the data in chapter 4 does show that there is a distinct bias towards countries that have a well-developed information technology infrastructure.

The data that was captured by Amber did reveal two methods of potentially gathering threat intelligence. A method was devised that attributed each IP address to an originating country, and then built a list of top countries that hosted malicious traffic. Multiple nodes' lists were combined and weighted based on attack prevalence, which resulted in a weighted list of countries. The weights could then be applied to other signature-based controls to increase the priority of potential incidents from those regions. The second method was originally thought to be applied on information gathered by multiple nodes but was later only valid on single nodes. The local bias method once again transforms the data into their attributed countries and looks for a strong bias towards a certain destination port. Potential incidents that are attributed to such a country, and which are using the same TCP/IP port that it was biased towards, should have their priority increased. The architecture was also discussed and outlined a method that Amber can be inexpensively deployed and managed, and that it is able to scale, benefiting from economies of scale when more nodes are added along with a command and control server.

5.1 Research Goals

The goal of this research was to design a security control that adhered to the restrictions of a hybrid security model, as detailed in section 3.2. Further more section 3.2 also placed three Critical Success Factors (CSF) on the control that had to be met in order for it to be a viable security control. The three CSF were:

- Zero-interaction system
- Near zero false positive rate
- Improves the security posture of the environment

A theoretical hybrid model was successfully constructed by combining two known security models, DtP and DtD, by reusing the presence based Discovery phase of the DtP model and the relative information based Action phase of the DtD model. Section 3.3 showed that a technical implementation of this hybrid model, named Amber, could be constructed making it possible to test it against the three CSF to judge the models success. The factors were tested in section 4.2 and section 4.3 by analysing the data collected by Amber nodes.

While section 3.3.4 does layout a plan that required continual human interaction with the system, it was for the purpose of evaluating the security control's effectiveness and not a process that would need to be repeated when Amber was deployed. Section 4.3 speaks to this point, as the deployed international Amber nodes operated independently for many months. Therefore this research managed to completely fulfil the first CSF of being a zero-interaction system.

The second CSF is more difficult to test, because of the nature of presence decision making. Because the model only used a presence based Discovery phase to mark incoming connections as malicious, the probability of a false positive is small. As stated in section 3.1.1 DtP presence based Discovery phase systems are deployed on informational assets that have no productive use, so connections made to them also have no productive use. While it may be difficult to show with a high degree of certainty that the connections that Amber logged were malicious, what can be said with a high degree of certainty is

that the connections that Amber logged had no valid reason for making that connection. Therefore this research managed to fulfil the second CSF of operating at a near zero false positive rate.

Section 4.2 dissects the data captured by a single Amber node and finds that if the system was connected to an upstream network enforcer it would have prevented connections to potentially vulnerable network assets that resided on the same segment as the node. This means that if a completed Amber node was deployed on an environment, the overall security posture of the environment would have been improved by its presence. This fully satisfies the third CSF of improving the security posture of the network segment.

Based on the above findings, this research was successful in the goals initially set out in section 1.2.

5.2 Remarks on Research Findings

The process of factorising well known concepts or data into meta-concepts and metadata is a recurring theme throughout this work. It is a process that managed to yield new methods of securing informational assets, as well as identifying additional ways of supplementing signature based security controls. This process allows for complex analysis post data collection, by transforming simplistic data that has already been collected (such as source IP addresses), into complex metadata which is then analysed. There is of course a degree of data integrity that is lost when it is transformed into metadata. This is seen when IP addresses were attributed to regions of potential origin in section 4.4 where certain regions did not have accurate IP to GPS translation. But this loss of data integrity can be acceptable as long as the conclusions that are drawn from the metadata are interpreted with that in mind.

The weighted scoring method categorises attributed regions according to a level of malicious threat activity, but it then uses that conclusion to supplement signature-based systems by increasing the scores of connections that are also attributed to high-risk regions. It does not propose restricting all access from the highest risk region because there is not enough confidence in the transformed metadata to make such a heavy handed decision.

That being said the research does show that there are differences in how regions are attacked, which is interesting because it adds a layer of physical data to a realm (cyberspace) that is theoretically without borders. The research suggests that there are more factors to consider when deciding where to physically host a service than just network latency. It would seem that certain regions experience more attacks of a specific kind, compared to other regions, such as the network in which the United States node was placed and the amount of 445/tcp traffic that it recorded.

5.3 Future Enhancements

There are certain aspects of this project that could not be fully explored because they fell outside of the initial scope as detailed in section 1.3. Some of these ideas are explored in the rest of this chapter.

There are numerous data transforms that could still be applied to the data that the Amber nodes collect. One such example was touched on in section 4.5.4, where the connecting IP addresses could be converted into their reverse DNS records and analysed for any correlation in how the reverse records were constructed. This process of converting the captured data into a different format and then looking for patterns can be compounded, by applying the same process on the converted data. An example would be to then take the reverse DNS records and convert them into the domain registration data, and then mine that data for patterns. The ability to compound means that there could be an endless amount of data transforms that could be constructed. The Command and Control server acts as a framework for compound analysis as it already stores all the data from all the connected nodes.

5.3.1 Geographically Distributed Nodes Linked Through Commonalities

Many organisations have networks that span across geographical regions, linking their distributed networks over the internet. As it was shown in section 4.3, there is little value in applying a blacklist of IP addresses that was collected in a different region, but because these organisational networks are related through a common owner, it is

perceivable that there could be value in using regional blacklists. This would follow the complex network architecture as shown in section 4.6.2, with the command and control server. The command and control server would be able to first verify if there are any common IP addresses in the data that the distributed nodes capture, showing that the common linkage does improve the intelligence that the nodes capture, and then distribute the combined IP blacklist to all network enforcers, in all regions. This was deemed out of scope because it would require finding an organisation that would be willing to deploy multiple Amber nodes across their geographical footprint, and connect them to an Amber command and control system. A global enterprise was approached with the idea but was rejected due to the enterprise wanting to place restrictions on information disclosure from the deployment.

5.3.2 Impacts of Time Zone on the Connections

Section 3.4 mentioned that the three distributed nodes were placed in three different time zones so as to increase the differences between them, but the effect of different time zones was not fully quantified in this research. This could be further explored increasing the amount of nodes across different time zones, and specifically track the changes in the profile of the IP addresses that attempt connections to the nodes. Possible research goals would be to investigate if there is a correlation between the time zone difference of the attributed attack country and the victim node's country. Time zone analysis could also potentially show if attacking malware is installed on workstations that traditionally are only powered on during office hours. This was outside of the scope of this research because it would require deploying multiple Amber nodes, in all time zones which creates an Amber ecosystem that is not reflective of a typical enterprise.

5.3.3 Deeper Understanding of Attacker and Victim Bias

Section 4.5 shows that different geographical nodes were favoured as victims to certain attacking regions. While some of the regions such as China were consistently present in all of the nodes' data, other attributed countries, such as Taiwan, were only seen on a certain node. It was not possible to analyse why certain victim regions were attractive

targets for specific regions because it required a level of detail in the captured data that was simply not present because it was out of the original scope for the Amber ecosystem (only recording source IP addresses).

5.3.4 Basic Automated Reporting

A malicious host on a network is not only a danger to the victim of the attacking machine, but it is also a liability to the infrastructure hoster. The methods that malware use to spread, as noted by Garetto et al. (2003), are bandwidth intensive because searches for vulnerable machines are not targeted. Instead large blocks of IP addresses are scanned. It is therefore economically significant for datacentre providers to quickly dismantle systems that are hosting malware on their infrastructure. Serving this goal, Amber nodes could be deployed into the datacentres of participating hosting providers across the globe, and could report the IP addresses that are captured to an Amber command and control server. The command and control server would have a list of IP ranges that each of the datacentre providers own, and would inform them if any of the IPs that the distributed Amber nodes report belong to them. This would give infrastructure owners an automated method of locating malware on their infrastructure. As section 3.3 showed, an Amber node requires very little resources to be deployed, meaning it could easily be slotted into a datacentre in exchange for any information that pertains to that particular hosting provider's infrastructure. This form of Amber deployment is specialised and out of the scope for this research as it does not reflect a typical enterprise deployment.

5.3.5 Normalise Scoring Based on Socioeconomic Data

When the attributed locations of the sourse IP addresses were visualised in section 4.4, it showed that regions such as North Africa had almost no connections attributed back to it. The weighted scoring system would ignore regions such as Chad because of the low amount of connections that were attributed to it, compared to Germany from which many connections originated. This does not mean that Germany is more malicious than Chan, only that Germans have relatively more access to internet connected computers than Chad. Future initiatives would normalise the data that was collected by the Amber

nodes for this and other non-information security metrics. This was deemed as outside of the scope for Amber as it focuses more on the school of Economics than Computer Science.

5.3.6 IPv6 Integration

Future versions of the Amber ecosystem will need to be able to support IPv6 as it becomes more prevalent in enterprise networks. This will be needed to keep Amber viable in the future.

References

S. Acharya, J. Wang, Z. Ge, T. Znati, and A. Greenberg. Simulation study of firewalls to aid improved performance. In *Simulation Symposium, 2006. 39th Annual*, pages 8–16pp. IEEE, 2006.

D. Akkaya and F. Thalgott. *Honeypots in Network Security: How to monitor and keep track of the newest cyber attacks by trapping hackers*. LAP Lambert Academic Publishing, 2012. URL `http://lnu.diva-portal.org/smash/get/diva2:327476/FULLTEXT01`.

E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb. Lessons learned from the deployment of a high-interaction honeypot. *Dependable Computing Conference*, 6: 39–46, 2006.

C. Alme and D. Eardly. Mcafee anti-malware engines: Values and technologies. Online, 2010. URL `http://www.mcafee.com/us/resources/reports/rp-anti-malware-engines.pdf`. Accessed on 19 Oct 2013.

L. Auriemma. Details about the ms12-020 proof-of-concept leak. Online, March 2012. URL `http://aluigi.altervista.org/adv/ms12-020_leak.txt`. Accessed on 20 Mar 2012.

P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The nepenthes platform: An efficient approach to collect malware. In *Recent Advances in Intrusion Detection 9th International Symposium, RAID 2006*, volume 4129, pages 165–184, Hamburg, Germany, 2006. Springer.

M. Bailey, E. Cooke, F. Jahanian, D. Watson, and J. Nazario. The blaster worm: Then and now. *Security & Privacy, IEEE*, 3(4):26–31, 2005.

E. Balas and C. Viecco. Towards a third generation data capture architecture for honeynets. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 21–28. IEEE, 2005.

D. Bernardo, P. João, C. A. Anderson, M. Nascimento, D. Amaral, and R. Timóteo de Sousa Júnior. Blind automatic malicious activity detection in honeypot data. *International Conference on Forensic Computer Science*, 6:142–152, 2011.

D. J. Bernstein. Syn cookies. Online, Feb 1997. URL http://cr.yp.to/syncookies.html. Accessed on 11 Feb 2013.

T. Caldwell. Locking down the VPN. *Network Security*, 2012(6):14 – 18, 2012. ISSN 1353-4858. doi: http://dx.doi.org/10.1016/S1353-4858(12)70055-7. URL http://www.sciencedirect.com/science/article/pii/S1353485812700557.

G. Chamales. The honeywall cd-rom. *Security & Privacy, IEEE*, 2(2):77–79, 2004.

D. B. Chapman and E. D. Zwicky. *Building Internet Firewalls*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition, 1995. ISBN 1565921240.

Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1890–1900. IEEE, 2003.

L. Cherkasova. Flex: Load balancing and management strategy for scalable web hosting service. In *Computers and Communications, 2000. Proceedings. ISCC 2000. Fifth IEEE Symposium on*, pages 8–13. IEEE, 2000.

B. Cheswick. An evening with Berferd in which a cracker is lured, endured, and studied. In *Proceedings Winter USENIX Conference, San Francisco*, 1992.

F. Cohen. Models of practical defenses against computer viruses. *Computers & Security*, 8(2):149 – 160, 1989. ISSN 0167-4048. doi: 10.1016/0167-4048(89)90070-9. URL http://www.sciencedirect.com/science/article/pii/0167404889900709.

F. Cohen. A note on the role of deception in information protection. Electronically, 11 2012. URL http://all.net/journal/deception/deception.html. Accessed on 25 Oct 2013.

M. J. Coss, D. L. Majette, and R. L. Sharp. Methods and apparatus for a computer network firewall with stateful packet filtering, Oct. 31 2000. US Patent 6,141,749.

M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th international conference on World Wide Web*, pages 281–290. ACM, 2010.

S. Das. *Your UNIX: The Ultimate Guide*. McGraw-Hill, Inc., New York, NY, USA, 2 edition, 2006. ISBN 0072520426, 9780072520422. ISBN: 9780072520422.

A. P. de Barros. Res: Honeytokens and detection. Online, April 2003. URL `http://seclists.org/focus-ids/2003/Apr/18`. Accessed 3 Jan 2014.

J. Dickinson. The new anti-virus formula. Online, 2005. URL `http://ebooks.z0ro.com/ebooks/Virus/ironport_new_anti-virus_formula.pdf`. Access on 1 Oct 2012.

Z. Durumeric, E. Wustrow, and J. A. Halderman. Zmap: Fast internet-wide scanning and its security applications. In *Proceedings of the 22nd USENIX Security Symposium*, pages 605–619, 2013.

M. Egele, E. Kirda, and C. Kruegel. Mitigating drive-by download attacks: Challenges and open problems. In *iNetSec 2009–Open Research Problems in Network Security*, pages 52–62. Springer, 2009.

Exogen. A geocoding toolbox for python. Online, 02 2010. URL `https://code.google.com/p/geopy/`. Accessed on 18 Mar 2013.

E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An internet con game. *Software World*, 28(2):6–8, 1997.

P. Ferguson. Rfc 2827: Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. IETF, 2000. URL `http://tools.ietf.org/html/rfc2827.html`.

D. P. Fidler. Economic cyber espionage and international law: Controversies involving government acquisition of trade secrets through cyber technologies. Online, Oct 2013. URL `http://internationallawandpractice.ncbar.org/newsletters/internationallawoctober2013/espionage`. Accessed on 18 Sep 2013.

N. Fitzgibbon and M. Wood. Conficker. c: A technical analysis. Online, 2009. URL `http://www.sophos.com/medialibrary/PDFs/marketing%20material/confickeranalysis.pdf`. Accessed on 3 Jul 2013.

J. Francois, O. Festor, et al. Activity monitoring for large honeynets and network telescopes. *International Journal On Advances in Systems and Measurements*, 1(1):1–13, 2009.

D. Gallagher. Mcafee earnings climb in second quarter. Online, July 2010. URL `http://goo.gl/nv9fT`. Accessed on 9 Feb 2013.

M. Garetto, W. Gong, and D. Towsley. Modeling malware spreading dynamics. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1869–1879. IEEE, 2003.

T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 193–206, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945464. URL `http://0-doi.acm.org.wam.seals.ac.za/10.1145/945445.945464`.

I. Gashi, V. Stankovic, C. Leita, and O. Thonnard. An experimental study of diversity with off-the-shelf antivirus engines. In *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pages 4–11. IEEE, 2009.

J. G. Göbel and A. Dewald. *Client-Honeypots: Exploring Malicious Websites*. Oldenbourg Verlag, Berlin, Germany, 2011. ISBN: 9783486711516.

R. Heady, G. Luger, A. Maccabe, and M. Servilla. *The architecture of a network-level intrusion detection system*. Department of Computer Science, College of Engineering, University of New Mexico, 1990. doi: 10.2172/425295.

L. T. Heberlein and M. Bishop. Attack class: Address spoofing. In *Proceedings of the 19th National Information Systems Security Conference*, pages 371–377, 1996.

M.-Y. Huang, R. J. Jasper, and T. M. Wicks. A large scale distributed intrusion detection framework based on attack strategy analysis. *Computer Networks*, 31(23):2465–2475, 1999.

B. Huffaker, D. Plummer, D. Moore, and K. Claffy. Topology discovery by active probing. In *Applications and the Internet (SAINT) Workshops, 2002. Proceedings. 2002 Symposium on*, pages 90–96. IEEE, 2002.

J. Hunker, B. Hutchinson, and J. Margulies. Role and challenges for sufficient cyber-attack attribution. Whitepaper, 2008. URL http://www.thei3p.org/docs/publications/whitepaper-attribution.pdf. Accessed on 3 Jun 2013.

C. Iheagwara, F. Awan, Y. Acar, and C. Miller. Maximizing the benefits of intrusion prevention systems: Effective deployments strategies. In *Proceedings of the 18th Annual Forum of Incident Response and Security Teams (FIRST) Conference*, 2006.

S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 190–199. ACM, 2000.

B. Irwin. A source analysis of the conficker outbreak from a network telescope. *South African Institute of Electrical Engineers African Research Journal*, 104(2):38–53, Jun 2013a.

B. Irwin. A baseline study of potentially malicious activity across five network telescopes. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–17. IEEE, 2013b.

ITweb. It salary survey july 2013. Online, July 2013. URL http://goo.gl/mZCbrx. Accessed on 13 Jan 2014.

D. Kaminsky. Rdp and the critical server attack surface. Online, Mar 2012. URL http://dankaminsky.com/2012/03/18/rdp/. Accessed on 25 May 2013.

D. Kaminsky. Black ops of tcp/ip 2011. In *Black Hat USA 2011*, page 44. Black Hat USA 2011, 2011. URL http://www.slideshare.net/dakami/black-ops-of-tcpip-2011-black-hat-usa-2011.

G. Keizer. Symantec false positive cripples thousands of chinese pcs. Online, May 2007. URL http://www.computerworld.com/s/article/9019958/Symantec_false_positive_cripples_thousands_of_Chinese_PCs. Accessed on 5 Jan 2014.

D. Kennedy. Project artillery - the most advanced threat intelligence solution. Electronic, 05 2012. URL https://www.trustedsec.com/downloads/artillery/. Accessed on 23 Feb 2013.

K. Kent and M. Souppaya. Guide to computer security log management. *NIST special publication*, pages 800–92, 2006.

J. Koziol. *Intrusion detection with SNORT*. Sams Publishing, Indianapolis, USA, 2003. ISBN 157870281X.

N. Krawetz. Anti-honeypot technology. *IEEE Security & Privacy*, 2:76–79, 2004.

C. Kreibich and J. Crowcroft. Honeycomb: creating intrusion detection signatures using honeypots. *ACM SIGCOMM Computer Communication Review*, 34(1):51–56, 2004.

G. Kress and T. Van Leeuwen. *Reading images: The grammar of visual design*. Routledge, London, England, 2006. ISBN 0203619722.

I. Kuwatly, M. Sraj, and Z. A. Masri. A dynamic honeypot design for intrusion detection. In *Pervasive Services*, pages 95–104, American University of Beirut, Lebanon, July 2004. IEEE/ACS International Conference, IEEE.

J. Lell. Quick blind tcp connection spoofing with syn cookies. Electronic, August 2013. URL `http://www.jakoblell.com/blog/2013/08/13/quick-blind-tcp-connection-spoofing-with-syn-cookies/`. Accessed on 15 August 2013.

J. Leyden. Kaspersky blocks bbc news over false phishing fears. Online, Jul 2010. URL `http://www.theregister.co.uk/2010/07/15/kaspersky_blocks_bbc_news/`. Accessed on 5 Jan 2014.

Y. Ma, Q. Yang, Y. Tang, S. Chen, and W. Shieh. 1-tb/s single-channel coherent optical ofdm transmission over 600-km ssmf fiber with subwavelength bandwidth access. *Optics express*, 17(11):9421–9427, 2009.

Mandiant Central Intelligence, MCI. Apt1 exposing one of china's cyber espionage units. Online, February 2013. URL `https://www.mandiant.com/blog/mandiant-exposes-apt1-chinas-cyber-espionage-units-releases-3000-indicators/`. Accessed on 18 Feb 2013.

P. K. Manna, S. Chen, and S. Ranka. Inside the permutation-scanning worms: Propagation modeling and analysis. *Networking, IEEE/ACM Transactions on*, 18(3):858–870, 2010.

MaxMind. Geoip products. Online, Mar 2013. URL `http://dev.maxmind.com/geoip/`. Accessed on 18 Mar 2013.

McAfee. Mcafee inc, form 10k. Electronic, Dec 2010. URL `http://goo.gl/j79oSw`. Accessed on 30 Aug 2013.

B. McCarty. The honeynet arms race. *Security & Privacy, IEEE*, 1(6):79–82, 2003.

D. McPherson and B. Dykes. Vlan aggregation for efficient ip address allocation (rfc3069). IETF, 2001. URL `http://tools.ietf.org/pdf/rfc3069.pdf`. Accessed 2 Jul 2013.

B. Merino. *Instant Traffic Analysis with Tshark How-to*. Packt Publishing, Birmingham, England, 2013. ISBN 9781782165392.

E. Michelakis, I. Androutsopoulos, G. Paliouras, G. Sakkis, and P. Stamatopoulos. Filtron: A learning-based anti-spam filter. In *Proceedings of The 1st Conference On Email And Anti-Spam*, pages 231–240. Citeseer, 2004.

Micosoft. Microsoft security bulletin ms08-067 - critical. Online, Oct 2008. URL `http://technet.microsoft.com/en-us/security/bulletin/ms08-067`. Accessed on 10 Jan 2014.

Microsoft. Mcafee delivers a false-positive detection of the w32/wecorl.a virus when version 5958 of the dat file is used print print email email. Online, June 2010. URL `http://support.microsoft.com/kb/2025695`. Accessed on 19 Aug 2012.

Microsoft. Microsoft security bulletin ms12-020 - critical. Online, April 2012. URL `http://technet.microsoft.com/en-za/security/bulletin/MS12-020`. Accessed on 10 May 2012.

D. Moore, C. Shannon, et al. Code-red: a case study on the spread and victims of an internet worm. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 273–284. ACM, 2002.

National Vulnerbaility Database. Database search. Electronic, October 2013. URL `http://goo.gl/4iqge6`. Accessed on 13 Oct 2013.

J. Nazario. Phoneyc: A virtual client honeypot. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, pages 6–6. USENIX Association, 2009.

R. M. Needham. Denial of service. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 151–153. ACM, 1993.

Net Security. Apple confirms being hit in recent watering hole attack. Electronic, February 2013. URL `http://www.net-security.org/secworld.php?id=14449`. Accessed on 12 Dec 2013.

H. Nomura, A. Watanabe, E. Tartakowskï, and E. Six. Vulnerability scanner: The complete toolsbox. *Metis*, 2:1, 2007.

J. Oberheide, E. Cooke, and F. Jahanian. Cloudav: N-version antivirus in the network cloud. In *Proceedings of the 17th conference on Security symposium*, pages 91–106. USENIX Association, 2008.

V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for internet hosts. *ACM SIGCOMM Computer Communication Review*, 31(4): 173–185, 2001.

R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of internet background radiation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 27–40. ACM, 2004.

P. Phaal. Method to associate input and output interfaces with packets read from a mirror port, July 17 2007. US Patent 7,245,587.

PhiBo. Dionaea is meant to be a nepenthes successor, embedding python as scripting language, using libemu to detect shellcodes, supporting IPv6 and TLS. Electronic, 12 2013. URL `http://dionaea.carnivore.it`. Accessed on 29 Aug 2013.

I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye. Ip geolocation databases: unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

P. Porras, H. Saidi, and V. Yegneswaran. Conficker c analysis. Online, 2009. URL `http://goo.gl/XGvn9s`. Accessed 10 Jun 2012.

J. Postel. Rfc 793: Transmission control protocol, 2003.

PR Newswire. Network associates ships cybercop sting - industry's first 'decoy' server silently traces and tracks hacker activity. Electronic, October 1998. URL `http://goo.gl/jbvo7`. Accessed on 10 Sep 2012.

K. M. Prasad, A. Reddy, and M. G. Karthik. Flooding attacks to internet threat monitors (ITM): modeling and counter measures using botnet and honeypots. *International Journal of Computer Science & Information Technology (IJCSIT)*, 6:159 – 172, 2012.

N. Provos. Honeyd-a virtual honeypot daemon. In *10th DFN-CERT Workshop*, February 2003.

K. Rieck, T. Krueger, and A. Dewald. Cujo: efficient detection and prevention of drive-by-download attacks. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 31–39. ACM, 2010.

A. Schiemenz and H. Igel. Accelerated 3-d full-waveform inversion using simultaneously encoded sources in the time domain: application to valhall ocean-bottom cable data. *Geophysical Journal International*, 195(3):1970–1988, 2013.

C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 208–223. IEEE, 1997.

B. Scottberg, W. Yurcik, and D. Doss. Internet honeypots: Protection or entrapment? In *International Symposium on Technology and Society, 2002.(ISTAS'02).*, pages 387–391. IEEE, 2002.

C. Seifert, I. Welch, P. Komisarczuk, et al. Honeyc-the low-interaction client honeypot. *Proceedings of the 2007 The 6th New Zealand Computer Science Research Student Conference, Waikato University, Hamilton, New Zealand*, 1:15–21, 2007.

Sherwyn. Ms12-020 rdp vulnerability overview and testing. Online, March 2012. URL `http://infolookup.securegossip.com/tag/metasploit/`. Accessed on 11 Aug 2012.

C. Sheth and R. Thakker. Performance evaluation and comparison of network firewalls under ddos attack. *International Journal of Computer Network and Information Security (IJCNIS)*, 5(12):60, 2013.

S. Shin and G. Gu. Conficker and beyond: a large-scale empirical study. In *Proceedings of the 26th Annual Computer Security Applications Conference*, pages 151–160. ACM, 2010.

S. A. Slaughter, D. E. Harter, and M. S. Krishnan. Evaluating the cost of software quality. *Communications of the ACM*, 41(8):67–73, 1998.

L. Spitzner. *Honeypots: Tracking Hackers*. Addison Wesley, Boston, United States, September 2002. ISBN: 0-321-10895-7.

L. Spitzner. To build a honeypot. Online, 1999. URL `http://www.spitzner.net/honeypot.html`. Accessed on 5 Feb 2012.

L. Spitzner. Honeypots: Catching the insider threat. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pages 170–179. IEEE, 2003.

W. R. Stevens and G. R. Wright. *TCP/IP Illustrated: Vol. 2: The Implementation*, volume 2. Addison-Wesley Professional, Indianapolis, IN, USA, 1995. ISBN 9780201633542.

C. Stoll. *The cuckoo's egg: tracking a spy through the maze of computer espionage*. Pocket, New York, USA, 1995. ISBN 0671726889.

S. Sudaharan, S. Dhammalapathi, S. Rai, and D. Wijesekera. Knowledge sharing honeynets. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 240–243. IEEE, 2005.

M. Tanase. Ip spoofing: an introduction. *Security Focus*, 11:30–37, 2003. URL `http://66.14.166.45/sf_whitepapers/tcpip/IP%20Spoofing%20-%20An%20Introduction.pdf`. Accessed on 19 Aug 2012.

S. J. Templeton and K. E. Levitt. Detecting spoofed packets. In *DARPA Information Survivability Conference and Exposition, 2003. Proceedings*, volume 1, pages 164–175. IEEE, 2003.

B. Tierney, B. Crowley, D. Gunter, J. Lee, and M. Thompson. A monitoring sensor management system for grid environments. *Cluster Computing*, 4(1):19–28, 2001.

M. Vizváry and J. Vykopal. Flow-based detection of rdp brute-force attacks. In *Proceedings of 7th International Conference on Security and Protection of Information (SPI 2013)*, 2013.

M. J. Warren and W. Hutchinson. Australian hackers and ethics. *Australasian journal of information systems*, 10(2):151–156, 2007.

S. Wei, A. Hussain, J. Mirkovic, and C. Ko. Tools for worm experimentation on the deter testbed. *International Journal of Communication Networks and Distributed Systems*, 5(1):151–171, 2010.

H. Welte. The netfilter framework in linux 2.4. In *Proceedings of Linux Kongress*, 2000. URL `http://goo.gl/GkxzSt`. Accessed on 7 Mar 2012.

G. Wicherski. Medium interaction honeypots. Online, 2006. URL `http://goo.gl/tHDYzm`. Access on 4 Jun 2013.

E. Willems. Number of new computer viruses at record high. Electronic, September 2010. URL `https://www.gdatasoftware.co.uk/press-center/news/article/article/1760-number-of-new-computer-viruses.html`. Accessed on 30 Aug 2013.

J. B. Williams. Entrapment. A legal limitation on police techniques. *The Journal of Criminal Law, Criminology, and Police Science*, 48(3):343–348, 1957.

M. Wolfgang. Host discovery with nmap. Online, 2002. URL `http://moonpie.org/writings/discovery.pdf`. Accessed on 20 Nov 2013.

World Bank Development Data Group. *World Development Indicators 2012*. World Bank-free PDF, Washington, D.C., USA, 2012. ISBN 9780821389850.

X.-R. Yang, Q.-B. Song, and J.-Y. Shen. Intrusion detection system. In *Proceedings of 2001 International Conferences on Info-tech and Infonet (ICII)*, pages 19–23, 2001.

F. Yarochkin. Nmap. Electronic, Decemeber 2013. URL `http://nmap.org/`. Accessed on 13 Nov 2013.

V. Yegneswaran, P. Barford, and J. Ullrich. Internet intrusions: global characteristics and prevalence. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 138–147. ACM, 2003.

T. Ylonen. Ssh–secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium*, pages 37–42, 1996.

F. Zhang, S. Zhou, Z. Qin, and J. Liu. Honeypot: a supplemented active defense system for network security. In *Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on*, pages 231–235. IEEE, 2003.

X.-s. Zhang, T. Chen, J. Zheng, and H. Li. Proactive worm propagation modeling and analysis in unstructured peer-to-peer networks. *Journal of Zhejiang University SCIENCE C*, 11(2):119–129, 2010.

C. C. Zou and R. Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 199–208. IEEE, 2006.