Localisation of Attacks

Combating Browser-based Geo-Information and IP Tracking Attacks

by

Masood Mansoori

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2017

Abstract

Accessing and retrieving users' browser and network information is a common practice used by advertisers and many online services to deliver targeted ads and explicit improved services to users belonging to a particular group. They provide a great deal of information about a user's geographical location, ethnicity, language, culture and general interests. However, in the same way these techniques have proven effective in advertising services, they can be used by attackers to launch targeted attacks against specific user groups. Targeted attacks have been proven more effective against user groups than their blind untargeted counterparts (e.g. spam, phishing). Their detection is more challenging as the detection tools need to be located within the targeted user group. This is one of the challenges faced by security researchers and organisations involved in the detection of new malware and exploits, using client honeypots. Client honeypots are detection systems used in the identification of malicious web sites. The client honeypot needs to mimic users in a pre-defined location, system, network and personality for which the malware is intended. The case is amplified by the use of Browser Exploit Packs/kits (BEPs), supporting these features. BEPs provide simplicity in deployment of targeted malicious web sites. They allow attackers to utilise specific geographical locations, network information, visit patterns or browser header information obtained from a visiting user to determine if a user should be subjected to an attack.

Malicious web sites that operate based on targeted techniques can disguise themselves as legitimate web sites and bypass detection. Benign content is delivered to attacker-specified users while avoiding delivery to

suspicious systems such as well-known or possible subnets that may host client honeypots. A client honeypot deployed in a single location with a single IP address will fail to detect an attack targeted at users in different demographic and network subnets. Failure in detection of such attacks results in high rates of false negatives which affect all honeypots regardless of detection technique or interaction level. BEPs are hugely popular and most include tracking features. The number of malicious web sites that utilise these features is currently unknown. There are very few studies that have addressed identifying the rate and number of malicious web sites utilising these techniques and no available client honeypot system is currently able to detect them. Any failure to detect these web sites will result in unknown numbers of users being exploited and infected with malware. The false negatives resulting from failing to detect these web sites can incorrectly be interpreted as a decline in the number of attacks.

In this work, a study of information that can potentially expose users to targeted attack through a browser is examined through experimental analysis. Concrete approaches by attackers to obtain user-specific information in the deployment of targeted attacks through browsers are discussed and analysed. We propose a framework for designing a client honeypot capable of detecting geolocation attacks. Our framework relies on HAZard and OPerability (HAZOP) studies to identify components of the client honeypot, its processes and attributes of the experimental setup which could potentially introduce bias into our study. Any potential bias neglected, would affect the results of our real-world experiments and undermine our analysis through deviation from the intent of the study. To facilitate in our experiments, we developed a low interaction client honeypoy (YALIH) and performed real-world experiments on large selection of web sites. We determined the popularity of targeted malicious attacks based on likely attributes of a visiting user's system. Our approach relies on previous research performed in the area of online spam detection which has similar attributes to malicious web sites. Our experiments show that referer,

via, X-Forwarded-For and browser language attributes of HTTP protocol header, retrieval behaviour (i.e. IP tracking) and geographical location of a visitor identified by an IP address can be used in a targeted attack. These attributes can have significant effect on the number of detected malicious web sites in a study and should therefore be reliably controlled in an experimental setup. This findings in this research can potentially reduce false negative rates in all types of client honeypots, measurement studies of malicious malicious web sites and help researchers and malware analysts capture and analyse new malware and exploit samples from malicious web sites.

Acknowledgments

PhD is not merely the process of writing a thesis but a journey through which one gets to reflect on himself, test self resilience and patience; life and people around him. Sadly all the hardship, emotions and gratitude has to be fit into a page. A line in this page but a sincere and ever lasting respect and admiration for those who helped me through this journey. Firstly, I would like to thank my family who stood by me throughout these years, ensuring me that their love and support would not lessen regardless of the outcome and gave me courage in my darkest times.

I would also like to express my gratitude to my advisor Associate Professor Dr. Ian Welch for his continuous support of my study, for his patience, motivation and emotional support. I could not have imagined having a better supervisor for my Ph.D study. The kindest, nicest and most supportive supervisor anyone could ever ask for.

I would also like to sincerely thank Patricia Stein and Suzan Hall for their kindness and support. One can not fathom the kindness and care Patricia has for students. She has been a counsellor and a truly great friend.

It would only be fair to thank my close friends in Wellington who always welcomed me with open arms and gave me a shelter away from this madness into another. Finally I would like to thank the medical community and modern science for their advancement in treating insanity, without which one would not start a Ph.D. Without their help, I'd surely be nowhere but six feet under.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Determining user specific information such as a user's geographical location, language and browser information is an approach used by many organisations and corporations to deliver improved services to particular users in a region. For instance, services such as Netflix and video streaming providers employ identification of users' geolocation to provide content within a particular country. Any user located outside the predefined region will be blocked from accessing the resources. Location-aware advertising or geo-marketing allows advertisers to deliver Ads to users based on highly accurate location-specific information or languages that can pinpoint a user within a region, ZIP code or within a certain radius of an advertising business. The Ads are displayed in the local language (e.g. Google AdWords). Attributes defined within the browser are also captured by online web servers, and then various versions of a web site in terms of language or system-specific are delivered.

User and geo-information identification techniques and other information that can be derived, whilst being an attractive approach to delivering improved services tailored to users, give rise to concerns about users' security. Targeted attacks can be deployed against users from a particular demographic, using current events, and cultural or social information to achieve better return rates [3, 4, 5]. Spam campaigns take advantage

of user location to deliver specially crafted content to users from certain groups or demographic profiles. The localisation of attacks makes their detection challenging. A detection system has to be actively monitoring, be present within the geographical boundaries, or pose as a member of the demographic being targeted to be able to detect the attacks. The availability of such techniques for an attacker allows them to identify and bypass security systems or domains where those systems might be located, by refusing to provide services or providing benign content [1].

Localisation and targeted attacks are not only limited to spam campaigns, but seen across other types of attacks. Statistics provided by research organisations running server honeypots have indicated patterns of attack that are unique to a particular region or environment (i.e. academic, corporate) [6]. Localisation and targeted attacks are believed to be factors in the distribution of malware, malicious web sites and a technique to evade detection [7, 8, 9, 10, 11, 12].

According to studies, attackers use certain specific information to determine which users should be subjected to a particular malware or exploit. The classification of users to be attacked can be by country, region, language, subnet, network and browser information [11, 8, 10]. For instance, due to the popularity of online gaming in China and e-banking in South America attackers have targeted respective users of these services in these regions [13, 14]. Malware also uses localised content and tactics designed to lure and then infect users (e.g. user's language) [8]. A large number of available toolkits (e.g. MPack, IcePack, Blackhole) [15, 16] are used to facilitate the deployment of targeted attacks.

Attackers are also able to monitor and log IP addresses of a user and deliver an exploit based on the suspicious behaviour of a visiting agent, its potential affiliation with security organisations and pattern of visits [10, 16, 17]. Studies on spam web sites have illustrated that IP tracking is used in determining whether a benign or spam email is delivered to the user [18, 19]. The same technique is believed to be used by malicious web sites

as several browser exploit kits provide such capability [10, 20].

Exploit kits also allow attackers to utilise a blacklist approach and block malicious content delivery to a particular subnet or IP address where the presence of a client honeypot is suspected. Accurate mapping of IP addresses and organisations allows an attacker to distinguish between low risk and high risk targets and attack or avoid them accordingly [18, 20, 21, 22]. Blacklisting subnets or IP addresses based on their risk of having a client honeypot, ensures evading detection and a longer lifetime, resulting in higher numbers of infected users over a longer period of time [17].

Malicious web sites that operate based on geo-information, IP tracking and browser header detection techniques can easily disguise themselves as legitimate web sites and bypass detection by redirecting visitors to benign web sites or delivering benign content [8, 11]. Failure to detect these web sites will result in unknown number of users being exploited and infected with malware. The false negatives resulting from failure to detect these web sites can incorrectly be interpreted as a decline in the number of attacks. "A measurement study could conclude there is a decline in attacks, whereas in reality an increase in false negatives is being measured" [10]. False negatives occur when a detection system fails to correctly identify an attack and does not raise an alarm. A false positive is a false indication of a threat which does not exist. A zero false negative rate is the ideal. False positives in a security system are less concerning, as a false positive alert can be correctly identified by manual analysis. A single false negative, however, could allow a successful attack and consequent compromise of the system. Generally false alarms are a direct product of the detection mechanisms used in the client honeypots. In a low-interaction client honeypot, a false negative may arise from the fact that the signatures have not been developed for that particular attack. However malware and exploit delivery based on geo-information, IP/subnet tracking are detection-technique independent and so will cause many false negatives in all types of honeypots regardless of the interaction and detection

method. A study to find malicious servers using distributed client honeypots in different physical locations is required to assess and understand the magnitude of the problem [16].

The challenges facing research in detection of localised and targeted attacks are multifold. The numbers of sensors, types of detection systems, and cost/analysis ratios are all issues that need to be addressed. Detection sensors need to be present within the domain of the attack to be able to detect them. A large number of attributes (e.g. regions, countries, languages, economic systems) need to be taken into account, requiring decisions on how many sensors should be placed, and at what locations to detect these attacks.

Analysis engines for the detection of malicious web sites vary in their implementation but all rely on active interaction for detection. This means retrieval of web site content for analysis. The cost of retrieval is a factor that needs to be addressed. Malicious web sites have capabilities for blacklisting traffic raising the issue of what detection platform can overcome this feature without compromising detection rate. Other questions include whether detection is best achieved using a distributed system with various detection engines, or a semi-distributed system with a single detection engine and multiple retriever agents and how to achieve detection without being detected.

1.1 Research Goals

The goal of this research is to identify how the reliability of detection systems can be improved by greater understanding of how attackers target vulnerable users with localised and targeted attacks. The specific research goals are:

- **Understanding the extent of the threat -** Despite the tools that attackers employ being widely known, no measurement study has

been conducted to accurately identify the scale of geo-information attacks and their characteristics.

- **Characterising the nature of the attacks -** A fundamental question is whether social or cultural attributes contribute to the campaign of targeted attacks within a region.

- **Identify the user attributes used in targeted attacks -** One of the outcomes of this research is identifying the most salient client attributes contributing to, and their prevalence in, geo-information and targeted attacks.

- **Define a conceptual system for geo-information that goes beyond geo-IP** - Retrieval systems used in the identification of malicious web sites include a large number of attributes which could trigger malware delivery or expose its identity to an attacker. These systems also require reliability components to control the system attributes in a real-world experiment. This research proposes the design of an information system retrieval and its components which utilise geo-targeting attributes to efficiently detect geolocation attacks.

1.2 Major Contributions

This research focuses on understanding the motives behind targeted browser-based attacks against visiting users of malicious web sites and attempts to determine their prevalence. This research also identifies physical and virtual attributes of visiting users of web sites which are used to determine their location and target them for explicit attacks. Hazard and Operability methodology is proposed to design real-world and large scale experiments on malicious web sites. The study identifies risks and challenges in real-world experiment setups using different types of client honeypots. Other contributions of this research include:

1. Identifying the motivations behind targeted attacks from an attacker's perspective and system or browser attributes which could subject a user to targeted attacks

2. Development of experimental tools

   (a) Design of a modular and multi-engine low interaction client honeypot.

   (b) Investigating potential bias and threats posed by client honeypot system components and proposing mitigation strategies in the study of malicious web sites.

   (c) Investigating potential bias and threats posed by processes and attributes of components in a multivariable experimental setup and proposing strategies to avoid and minimise their effects. This study discusses threat control and mitigation studies to maximise the internal and external validity of validity the experiments using client honeypot systems.

3. Large-scale and real-world experiments to measure the impact of identified attributes in contribution #1 on malicious web site behaviour and malware delivery

   (a) Determining the effects of HTTP referer value in malware delivery using a logistic regression model.

   (b) Determining the role of IP, browser language Network, HTTP Proxy, Via and From header values in malware delivery.

   (c) Evaluating the role of location based IP techniques in malware delivery by malicious web sites through large scale experiments

4. Propose and implement a prototype system for efficient detection of malicious web sites with (a) Target Identification. (b) Compression, minification and code extraction, (c) Scalability and (d) Reliability features.

1.3 Thesis Organisation

This thesis is organised as follows:

In **Chapter 2** we focus on socioeconomic motives behind targeted attacks in terms of return of investment and operational risk for an attacker. We discuss how that social factors such as a common attribute or an activity within a population, socioeconomic status of a target population and cyber security and extradition laws can influence the likelihood of a population to be the target of an attack.

Chapter 3 focuses on identifying the potential attributes of a user's browser, system and operational environment which could expose them to targeted attacks. The feasibility of these techniques in the identification of a user's location and deployment of malicious web sites to target them are studied. This chapter also discusses types of attacks directed at end user browsers and focuses on related work in the area of browser and malware based geolocation attacks. Existing client honeypot systems are classified, reviewed and their capabilities in detection of geolocation attacks are evaluated.

Chapter 4 discusses challenges in the design of large scale and real-world experiments of malicious web sites and the potential introduction of threats and bias into such studies resulting from various components of the detection engines, subjects of the study and the attributes of the experiment's environment. HAZOP and Operability study is discussed in this chapter through which we analyse the components of the experiment, identify variables which may introduce erroneous analysis in the study and discuss ways to mitigate or minimise them.

Chapter 5 focuses on the development of a low interaction client honey-

pot in accordance with the HAZOP analysis performed in chapter 4. Several experiments are performed to compare its detection and performance against other client honeypots.

In **Chapter 6** we perform a large scale study on the effects of HTTP referer attribute and examine its role in the process of a web site malware delivery. The results of this experiment is used in all other experiments to input the optimal referer settings which may trigger the highest number of attacks. We also discuss the setup and experiments on large sets of suspicious web sites and determine the impact of several potential attributes of visiting users such as IP address, location inferred from IP address, network subnets, HTTP header fields and language settings of users in delivery attack behaviour of malicious web sites.

In **Chapter 7** we propose and implement a geolocation retrieval system for efficient retrieval of geolocation malware using physical and logical attributes of a client honeypot.

Chapter 8 concludes this thesis, discusses the contributions and suggests future areas in which this study could be further enhanced.

1.4 Publications

Chapters 3, 4, and 6 have been partially or entirely published in the following conferences and journals.

- **Chapter 3 -** Masood, Mansoori, Ray Hunt, (2011), An Overview of Browser Vulnerability Attacks, Countermeasures and Detection Tools, Journal of Network Forensics, secAU Security Research Centre, Australia, Vol. 3, issue 1

- **Chapter 3 -** Masood Mansoori, Ray Hunt, (2011), An ISP Based Noti-

fication and Detection System to Maximise Efficiency of Client Honeypots in Protection of End Users, International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.5, Sep 2011

- **Chapter 5 -** Masood Mansoori, Ian Welch, Qiang Fu, "YALIH, Yet Another Low Interaction Honeyclient", Australian Information Security Conference (ACSW - AISC), January 20 - 23, 2014, Auckland, New Zealand [Best Student Paper Award]

- **Chapter 6 -** Masood Mansoori, Ian Welch, Seyed Ebrahim Hashemi, "Measurement of IP and Network Tracking Behaviour of Malicious Web sites", Australian Information Security Conference (ACSW - AISC), February 2 - 5, 2016, Canberra, Australia

- **Chapter 5 -** Masood Mansoori, Yuichi Hirose, Ian Welch, Kim-Kwang Raymond Choo, "Empirical Analysis of Impact of HTTP Referer on Malicious Web site Behaviour and Delivery", The 30th IEEE International Conference on Advanced Information Networking and Applications (AINA), March 23-25, 2016, Crans-Montana, Switzerland

- **Chapter 4 -** Masood Mansoori, Ian Welch, Kim-Kwang Raymond Choo, "Application of HAZOP to the Design of Cyber Security Experiments", The 30th IEEE International Conference on Advanced Information Networking and Applications (AINA), March 23-25, 2016, Crans-Montana, Switzerland

- **Chapters 4 and 6 -** Masood Mansoori, Ian Welch, Kim-Kwang Raymond Choo, Roy A. Maxion, Seyed Ebrahim Hashemi, "Real-World IP and Network Tracking Measurement study of Malicious Web sites with HAZOP", International Journal of Computers and Applications - Special Issue in Information Security, Taylor & Francis, Vol. 39 , Iss. 2, 2017, Pages 106-121

Part I – Why are Attacks Targeted and How Do They Do It?

Chapter 2

Targeted Attacks: The Motivation

Targeted attacks are defined as per-user attacks where an attacker puts an effort into finding and exploiting vulnerabilities of each single or small group of users using gathered information about the particular target(s) and utilising more personal and sophisticated approaches. On the other side of the scale are non-targeted attacks which are directed at a large magnitude of people with less commonality with ready tools and scripts to exploit popular vulnerabilities of users. In this chapter we discuss the socioeconomic motivations behind targeted attacks from an attacker's point of view.

An attacker, whether a person, group or organisation has a limited amount of resource (e.g. bandwidth, time) and cost per user (per user effort); although these two terms are interchangeable. Any attacking entity expects the highest returns against costs regardless of the attacking approach used. In a conventional non-targeted attack scenario, an attacker initiates a broadcast internet-scale attack and scans a large subnet of IP addresses for open ports and services to identify live and vulnerable hosts. This is then usually followed by attacking those identified hosts with certain predefined exploits using an automated tool. Initial knowledge or close proximity to the targeted hosts are not required in a non-targeted attack. An attacker employing a scalable approach follows an economic

approach where costs decreases and profit increases with increasing number of target hosts. generic spam, phishing, Drive-by download, worms and viruses are subsets of non-targeted or scalable attacks.

Scalable attacks are automated and do not adopt to individual users regardless of their value, as argued by Zeeuwen *et. al.*[84]. These attacks are profitable as long as the increase in return value is greater than the additional cost introduced by the cost (e.g. time spend to include personal information in a spam/phishing attack or employ more sophisticated attacks) to make it more directed. While a simple targeted attack yields a higher return value than a non-targeted counterpart, its economic cost/return ratio does not justify the effort, hence most of non-targeted attacks have no single target personalisation, are fully automated and simply ignore the target and move to the next if initial attacks fail. Their automation however allows them to be replicated by people with little knowledge of the actual attack, in large scales.

Targeted attacks require more resources and have higher cost (e.g. time and effort per user). Unlike scalable non-targeted attacks however, the costs are linear and increase at an identical rate regardless of the number of the users attacked. This is because individual per-user effort is put into attacking every target. This may seem like a disadvantage; however keep in mind that targeted attacks expect a higher return value per target. This is partially because targeted attacks are directed at hosts with potentially higher than average extractable value. The identification of such targets nonetheless proves difficult to achieve as many targets appear uniform within a pool of thousands of potential hosts with no observable value to distinguish them. Per-user effort to identify them increases the cost of the attack, resulting in lower return value. Attacking all possible hosts is not an efficient approach with targeted attacks as the cost increases linearly per user. Therefore an attacker must take into account attributes that could be used to help him lessen the number of the potential targets that have higher desired extractable values. One of these attributes is deter-

mining users that have observable values. An observable value can be in the form of fame, wealth, public or political power/reputation or anything valuable that is observable to the attacker. The cost of extracting values from observable high value targets however is expected to be higher than average user, with no certain return. A publicly known credit card database has a high concentration of extractable value. Considering the strict security mechanisms in place to protect the public database against unauthorised access, a targeted attack could take significant amount of time and resources to penetrate the system with no success. This significantly increase the overall cost of the targeted approach. An attacker could also spend hours gaining access to an expected high value target and not obtain the expected return. It is simply because the value of the target extractable data can be measured after the attack has taken place successfully and data has been extracted. This is the dilemma faced by targeted attackers that only target users with high expected extractable value.

In order to reduce the overall cost of targeted attacks, a logical step would be to deploy scalable automated attacks against target users where the cost of extraction is lower and expected return value exceeds the cost of extraction. High extractable value targets within those targets range can then be identified and approached for more sophisticated targeted attacks (non-scalable attacks). Attributes that contribute to lower extraction cost can be in the form of users with lower security knowledge and unpatched systems or communities/organisations with weak security infrastructure. An attacker might also have proximity advantage to the target or possess information that increases the probability of a successful initial attack. Familiarity with the culture, language or events within a community and personalizing attacks based on those information significantly increases the probability of successful attacks [85]. On the other hand, sending phishing and spam with English contents to targets within China for instance, where a limited number of users are familiar with English language only results in higher costs than returned value. We define

attacks targeted at a large population based on the geographical attributes and cultural similarities as geolocation attacks.

The common practice of delivering malicious content to users from certain countries by attackers is called geo-targeting. There are several reasons which would justify such a behavior from an attacker's point of view. Generally cyber criminals target a region based on an assessment of the value of the target, number of targets, associated risk and ease of operation. These assessments requires an insight into the socio-cultural and economics of a region and its population.

Malware distribution similar to any business strategy operates based on profits ratio compared to the investment. The cost and investment for a malware distributer or malicious Web site operator can be in the forms of server, time, traffic, IP address and domain name purchases and any tools purchased for deployment and distribution of malicious software (e.g. Browser exploit kits). The Return on Investment (ROI) for a targeted attack on a large geographical area (e.g. country) depends on several factors which individually determine the rate of return for a malicious campaign. We categorise these factors into three categories of:

- Social factors
- Value of obtained information
- Economic factors
- Availability of a market and marketplace for the information
- Legal drives

2.1 Social factor

Targeting a particular group for information requires the information to be popularly available within the group. These attack campaigns target

general population based on factors which are an integral part of that particular population. These factors could be an activity, characteristic or a sociocultural attribute common between the members. Targeting strategies have been widely observed in spam campaigns for years. Spam are localised within specific regions and vary from a region or a country to another. The common pattern in most of the targeted spam attacks have been the existence of a common sociocultural attribute or activity to socially engineer the attack for higher impact. Similar patterns of attacks are observed in campaigns propagated through malware.

A study by Medler [3] indicated campaigns of attacks targeting users in East Asian countries for online gaming credentials. The reason behind such a targeted attack is quite evident. Online gaming has a strong presence today with large connected networks of millions of players. A multibillion dollar business involving not only the sale and reselling of games but in-game items and currencies exchanged within each game environment. Gaming companies have shifted their focus from individual gaming platforms to game-as-service model in which a gamer's identity is extended across a number of games. These game-as-service platforms allow players to generate capital either financial or social through obtaining additional services through intensive time-consuming activities or online purchase of in-game items resulting in players placing high value in their game identities. Game companies further contribute to the expansion of such model by releasing updates and extension packs which add new objects to attain or purchase. New objects, and new ways of acquiring them ensure that the market for in-game objects doesn't stagnate; the game's developers also track the market to ensure that balance is maintained within the game. This model has further increased the value of game and player ownership [3].

Delivering spam emails with subjects and content related to popular activities and interests within a community ensures higher chances of successful attacks. This is a personalised attack but at a broader context. Pop-

ularity of an activity within a large portion of the population and availability of communities which facilitate the exchange of such items for other currencies has created an underground economy ensuring a high rate of return and resell value for such a targeted attack. The size of the gaming community plays an essential role in such a targeted attack. The target group must be large enough to maximise the probability that a certain percentage of recipients will also be those who play a particular game. This ensures that the cost, resources and effort placed into an attack are justified. Online gaming is a widespread activity within younger generation in Asian countries. Therefore large portions of gaming attacks are targeted toward these communities [4].

Online banking is another example of regional targeted attacks based on prevalence of a socioeconomic factor in a region. Several countries in the region have gone through large economic boom in the last decade. Countries such as Brazil, Argentina with high population and large workforce have transformed themselves into large global economies. As a result of such an economic boom, online banking has become a widespread activity and a way of life. Brazil and Argentina have the highest percentage of online banking activity in the world per capital. Brazil doubled its online banking clients from 8 to 18 million between 2002 and 2004. Online banking adaptation surpassed the rate of Internet adaptation in the country [5]. In another report by Federação Brasileira de Bancos, in Brazil 51 percent of all banking transactions were conducted via internet connected devices during first half of 2013, of which 45 percent were performed using online systems. During a survey performed on March 2013 Brazil was among the top three Latin American countries to lead in online banking representing a 43.7 percent of all Internet users who banked online surpassed by Chile and Venezuela each achieving 43.8 percent and 47 percent respectively (Figure 2.1) [1].

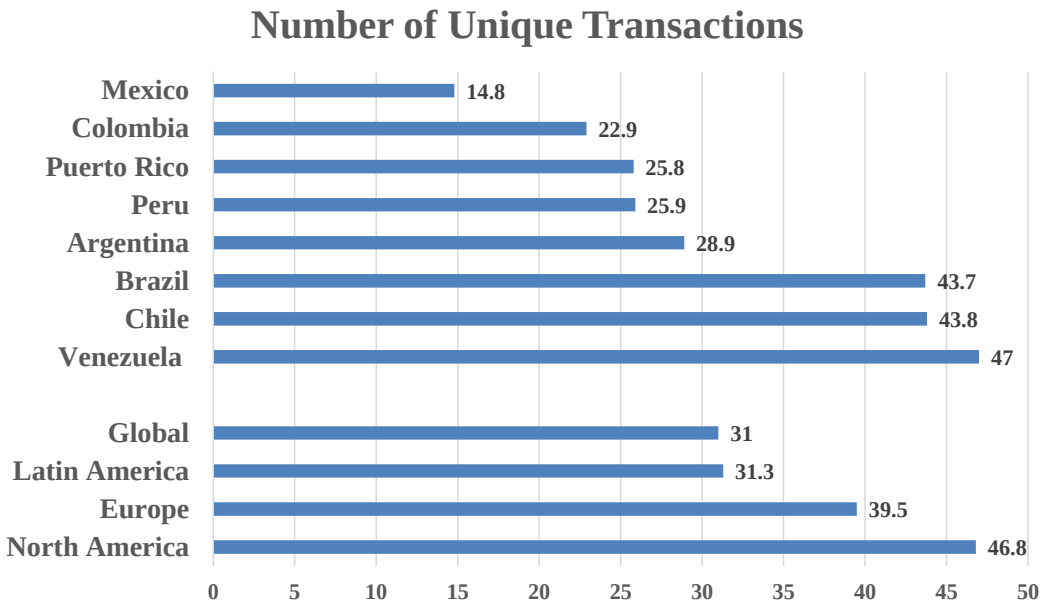


Figure 2.1: Percentage of all banking transactions in 2013 [1]

2.2 Value of Obtained Information

Value of obtained information plays a key part in any attack and determines the cost an attacker is willing to accept and the resources to use. Since geo-targeted attacks are directed at a smaller population than full scale non-targeted attacks, the value of expected return should justify the reduced probability of infected targets and return on investment. The value of information or data which is obtained can in all cases be attributed to the real-world currency the information is exchanged for regardless of the type of the obtained resources.

The value of information in a market similar to goods and services is highly dependent on supply and demand. A resource or information generally sees highest return when the type of attack to obtain it has not evolved to be performed by automated tools at large scale. We can assume that the more targeted an attack is, the higher the value of obtained

information may be. These attacks relied on the knowledge of attackers who had put time and resources into obtaining that information. With the introduction of automated tools which enabled infection at high rate, the value of the information on the market rapidly dropped. For instance, the cost of phishing and sending spam and its value has been declining significantly in the last few years partially because of the high supply of infected machines. This has resulted in oversupply of the market with information generated by automated tools which has driven the price down [6, 7]. The high number of infected hosts is the results of automated tools such as browser exploit kits. The cost of sending spam through the infected machine has declined sharply from $30 (10,000 messages) to $2 from 2011 to 2013 [8]. A similar pattern of price decline due to increased supply can be observed for various other types of products and services within underground economy.

The credit card market has always been one of the most lucrative markets in the underground economy due to countless ways credit card information can be used for fraud. Its demand is partially assisted by the low number of supplies. However incidents involving large scale attacks on retailers which resulted in the breach of a large number of customer and credit card information, the prices have decreased significantly [8]. Similar pattern can be observed in the gaming community where the value of an in-game item in an underground economy depends various factors. These factors are the difficulty and game-play hours required by which the item can be obtained through legitimate processes, the size of the game community or the number of items available online through stolen accounts. As the number of stolen accounts increased through malware, the market was saturated with in-game items which would have otherwise obtained through higher resources and cost. The market plummeted and profit per head decreased significantly.

2.3 Economic Conditions

Large scale targeted attacks are also directly associated with the economic state of the targeted region and corresponding population. Income levels and standard of living scale may predict the likelihood and type of an attack in a region. Population in wealthier countries are more attractive targets for attackers since they are much more likely to have:

1. Have access to online financial services and credit cards: Number of credit cards per person in circulation for developing countries far exceeds those in third world regions. According to a report in 2015, there are more than 2 credit cards issued per person in most of the European and developed East Asian countries. In Taiwan, Singapore, South Korea and Canada for instance, an individual has access to four different credit cards while credit card per capita is 0.4 and 0.1 percent in Kenya and South Africa respectively [9, 10].

2. Internet penetration and reliability: Penetration and the reliability of the Internet infrastructure also play an important role in the attack process and are directly related to the economic estate of a particular country. Higher Internet penetration corresponds to a larger target population while reliability of access ensures higher attack surface in which more complex and large scale attacks can be performed. Internet penetration rate and reliability of the connections also directly influence the amount of data people store online. Such practices makes end users within that particular population more intriguing targets to attackers and more vulnerable to blackmailing and ransom.

   A good economic state generally can be associated with more advanced infrastructure, Internet proliferation and digital platforms. Advanced infrastructure results in companies offering commercial services online and people using those services for their everyday activities including financial transactions and communications. The

more digitally connected a country is, the more economic value is digitised and stored on networks which makes the population a more intriguing target for attackers due to the higher value stored information [11]. The relationship between economics and number of attacks can be visibly seen in large economies in South America where developing economy and digital development has resulted in a higher number of middle class consumers and widespread use of online banking. Banks therefore have streamlined their processes to be performed online as opposed to in-bank transactions. On the contrary, in regions where financial developments are weak and online infrastructure is not available to support online services (e.g. online banking), a small number of financially targeted attacks toward general population is observed.

3. Mobile devices: Continuous Internet access on mobile devices not only provides a communication platform for users but also provides additional platform to target specific users.

Since the ROI is higher on a wealthy population than a poor region, considering the similar costs, efforts and risk associated with an attack, it is justified for an attacker to target a wealthy population. The Magnitude browser exploit kit's list of banned countries shows the relationship between valued targets and their economic development in which a large portion of the list of the countries where traffic is directed to benign Web sites consists of countries with under-developed economies (Figure 2.2).

```
function checkcountry() {
   $block=false;
   $blocked_country = array('Al','A2','01','SU','RU','UA','BY',
  'UZ','KZ','AZ','LT','MD','LV,'KG','TJ','AM','TM','JP','JA',
  'CN','MY','TZ','PH','SG','TT','YE','LK','PK','SA','BG','UY',
  'RS','OM','IQ','KW','DO','V','KE','EU','NP','BD','MN','SK',
  'CR','JO'LU','BB','MU','NI','AP','BS','NG','CY','BO','AO',
  'PY','MK','GU','BH','SI','NA','LB','BA','BN','GD','LA','BZ',
  'PG','ZM','SY','LY','SD','HT','MO','PS','UG','GF','RE','AF',
  'SN','LR','NC','GP','BW','HN','AW','PF','CW','VI','IS','KN',
  'AG','BM','GY','DM','MT','BT','MZ','EE','GL','CI','MG','MV',
  'MC','GA','CD','LI','GQ','ZW','CM','SR','JE','DJ','CV','SZ',
  'ME','FJ','LC','KY','GH','SB','VU','ET','RW','MW','ER','LS',
  'EG','AE','TW','ZA','MM','BF','MA','DZ','RO','TN','MQ','GE',
  'MM')
    if (empty($_SERVER[GEOIP_COUNTRY_CODE]) ||
       empty($_SERVER[HTTP_ACCEPT_LANGUAGE]))
  $block = true;
    else {
  for($i=0; $i<count($blocked_country); $i++)
     {
      if (($_SERVERPGEOIP_COUNTRYCODE1 == $blocked_country[$i])
          ||
      (stripos($SERVER['HTTP_ACCEPT_LANGUAGE'],$blocked_country[
         $i])!== false))
   $block = true;
     }
   return $block;
}
. . .
if(checkcountry())
   $redir=O;
```

Figure 2.2: The Magnitude browser exploit kit's list of banned countries which will be redirected to a benign Web site

Several other geotargeting malware exhibit a similar pattern in terms of user selection based on economic factor and popularity of online financial service. Bradesco, Sicredi and Ric malwares targeting users in Brazil for online banking credentials; Dridex is widespread in the U.S. and Germany; Locky ransomware targets Danish and Portuguese speaking users and Trustezeb is active in German speaking counties.

Ratio of return value over the cost of obtaining the information might not be clear to an attacker targeting a large population. There are however factors that may help classify them based on ”expected” return value and can help narrow the search. For instance targeting users in economic systems where online banking is highly prevalent is expected to return higher values. Targeting users that might have high value in form of assets in the bank but belong to an economic system where online banking is not prevalent has little to no extractable value and results in high cost, simply because even if attacks are successful and access to the target system is gained, there is no way to transfer those extractable values to the attacker.

2.4 Availability of a Market

An important yet less significant factor in return value of information is the availability of a secure infrastructure or marketplace in which the obtained information can be exchanged for real-world currency. A marketplace ensures that sellers and customers interested in the product can communicate with confidentiality. Depending on the confidentiality of the information various exchange systems exist. Stolen In-game items can be either sold on public Web sites such as eBay or forums through direct purchase [12] Generally the more illegal the content of these boards, the more vigorous the efforts of the operators are to protect themselves against unauthorised readers [13]. Credit cards may go through transition to gift cards or purchased items and subsequent reshipping and resell.

2.5 Legal Drives

Jurisdiction is one of the major constraints that affect the results of cybercrime prosecution. Jurisdiction may refer to the territory within which a court or government agency may properly exercise its power. An im-

portant obstacle related to jurisdiction in cybercriminal cases is that many attacks are initiated from a different region outside the jurisdiction of the target's court system. Even if the location of an attacker is identified, federal agencies in the victim's country lack the proper authority to prosecute the cybercrime offender. Authority of a state beyond its borders would require similar cybercrime laws and extradition policies signed between two states. Extradition laws however entail double criminality law which requires an offence to be considered a crime both in the jurisdiction the crime was committed at as well as the country requesting an extradition. Many countries however do not have a clear definition of a cybercrime or have a judicial system in which these crimes are not prosecuted and where regulations and enforcement arrangements are weak [14]. For instance Brazil's weak laws for cybercrime and intellectual property protection has resulted in widespread cybercrime activity and active underground hacking groups who face little risk of arrest or prosecution within the country [15].

Law enforcement agencies are always limited to jurisdictional boundaries. Only few countries have a well-developed and robust legal system that prohibits the various types of cybercrimes and possess resources, technology and international influence to pursue criminals beyond their borders. Night Dragon attack is an example which illustrates the dissimilarity of regional laws in cybercrime cases where lack of robust cybercrime laws in the attackers country (i.e. China) and lack of collaboration and extradition between the two countries (i.e. China, US) did not result in an arrest [16]. Attackers take advantage of international law structure by targeting users from a country where extradition laws are not signed by the countries (Country where the attacker resides and where the victim resides). An examples directly related to this research is acquisition and use of the Magnitude Exploit Kit which allows targeted attacks yet does not permit its operation within or directed at users in several highly valued countries due to legal reasons.

2.6 Summary

In this chapter, we identified the socioeconomic and legal motivations behind targeted attacks and discussed them in terms of return of investment and operational risk for an attacker. We identified that social factors such as a common attribute or an activity within a population and socioeconomic status of a target population can be a starting factor for an attacker to launch a targeted attack. Common attribute or activity between the target population makes them susceptible to attacks based on social engineering and local content, which have higher probability of success compared to a generic attack.

Value of an information and availability of a market place to trade the retrieved information in exchange for real currency plays a significant part in any attack; either targeted or generic. Targeted attacks are however highly influenced by cyber security legislation and specially extradition laws enforced in the country of the attacker and the country where the targeted users reside.

Chapter 3

Background and Literature Review

In this chapter, we discuss various geolocation techniques and attributes of a visiting user's browser, operating system and intermediary devices which could potentially be used by malicious web sites to identify the location of the visiting users. We will also provide a taxonomy and survey of available client honeypots and investigate their capabilities in terms of detection and specifically geolocation targeted attacks. We will finally identify the gap in research on geolocation targeted attacks which will be the basis of this research.

3.1 Examples of Geo-targeting Malware

Example 1 - Ransomeware

The Ransomware family of malware includes localised examples based on different languages, designed for various countries. They are distributed by malicious Web sites operated by Blackhole or similar exploit kits [17]. Upon infection, the malware locks the victim's system. Then using localised language and banners representing the corresponding police force

of the user's country, it demands a sum of money as a fine for alleged possession of illicit material. The ransom is paid through legitimate online payment services. The malware is active in most developed countries. Widespread infection by Ransomware malware has been observed in Germany, United Kingdom and Spain. The Ransomware family of malware is reported to have a high rate of accuracy in geo-locating users (i.e. 90% accuracy).

Example 2 - EoRezo

Malware distribution based on geographical location and language was observed in a study by security vendors. Win32/EoRezo is an adware program that targets users in France and delivers French-language advertisements. It was the most commonly detected malware in France. This software creates pop-up windows and causes web browsers to redirect to advertising Web sites, and may track a user's activity and send reports back to the creator [86].

Win32/Pameseg is a family of malicious installers; most variants target Russian speakers in Russia and former Soviet territories. They require users to send an SMS message to a premium number to download and install certain free and pay cracked shareware applications. Figure 3.1 shows an example of a Win32/Pameseg malware packed using ZipArchive, interacting with the remote server tool and requesting a validation code [19].

Figure 3.1: Interaction of Pamesga malware with remote web server and retrieving country list

```
<ziparchive>
<response>
..
<countries>
<hash>0</hash>
<item key="49"> Россия</item>        #Russia
<item key="25"> Казахстан </item>    #Kazakhstan
<item key="10"> Индия </item>        #India
<item key="99"> Бразипия </item>     #Brazil
</response>
</ziparchive>
```

```
<ziparchive>
<response>
<number>9999</number>
<code>59596</code>
<small_number>7375</small_number>
<small_code>59596</small_code>
<prefix>41110</prefix>
</response>
</ziparchive>
```

Example 3 – Waledac Malware Family

The Windows based Waledac family of malware is a prime example of targeted attacks that use GeoIP information to lure potential targets. The malware uses geographical proximity information through matching the IP address of a visiting user with a certain location to provide luring content in the form of socially engineered local events, promotions or fake bombing news. The malware infects users through drive-by browser vulnerabilities, or direct download installation of a rogue flash plugin for a video of a fake bombing incident in a region in close proximity to the target's location (Figure 3.2). Upon infection, the worm also propagates through spam by which similar socially engineered content is forwarded based on the geographical location of a recipient [20, 21]. Infected hosts are subsequently used as spam bots to propagate spam and infect other hosts.

```
"At least 12 people have been killed and more than 40 wounded in a bomb blast
near market in [User Location e.g. Amsterdam]. Authorities suggested that
the explosion was caused by a "dirty" bomb. Police said the bomb was detonated
from close by using electric cables. "It was awful" said the eyewitness about
blast that he heard from his shop. "It made the floor shake. So many people were
running"
```

Figure 3.2: Magnitude's list of banned countries which will be redirected to a benign Web site

Example 4 – Conficker/Downadup Malware Family

The Conficker family of malware is considered to have produced one of the most successful campaigns reaching over 15 million infections over a short period [22]. The malware has been widely studied due to its high rate of infection, its speed and methods of propagation, target selection and detection evasion techniques [87, 88]. Its propagation techniques are through server-side services and include direct exploitation of a popular vulnerability MS08-067 (CVE-2008-4250), and through Remote Procedure Call (RPC), removable media infection, scanning local networks for potential targets and NetBIOS shares. Upon successful exploitation of a host, the malware creates a web server with a random port on the infected host and forwards the URL address to remote hosts for further infection. The infection of the remote host requires knowledge of the type and version of the operating system and the language set within the targeted host system. Conficker malware estimates the language of the remote system by contacting MaxMind's publicly available IP geolocation databases for IP-to-location information [89, 90]. Newer versions of Conficker contain encrypted IP geolocation information embedded within the malware to allow higher reliability and avoid detection through contact with outside services. Conficker matches the IP address of the remote host to a country using the embedded database and subsequently maps that country to a particular language [90].
Ransomare, EoRezo, Waledac and Conficker are few examples of malware which utilise various geolocation techniques for target identification and attack. The question is: How does an attacker know where you are located; and what attributes within a client's environment or in our case, its interaction with the web server reveal its location?

3.2 Geolocation Detection Mechanisms

In the following sections, we discuss various geolocation techniques and attributes of a visiting user's browser, operating system and intermediary devices which could potentially be used by malicious Web sites to identify the location of the visiting users. We categorise these techniques into three distinct categories:

- **Communication and Transmission Protocol Attributes -** HTTP protocol includes multiple header fields which inform a remote web server of client's browser and system properties and supported services. These information provided can be used to estimate the location of a visiting user.

- **Client-Based Techniques -** All modern popular browsers support add-ons which extends the capabilities of the browser. JavaScript for instance is an example of a popular extension for modern browsers which provides support for creation of dynamic content and complex in-browser services. These services allows execution of code which directly inform a remote web server of the client's location.

- **IP-Based Active Measurement Techniques** - IP-based active measurement techniques exploit the routing behavior, network topologies and physical properties of packets such as transfer time from multiple predefined landmarks to the client system, identified by its IP address to calculate a relative distance to the remote node. These techniques may also rely on active probes to directly retrieve information from the intermediary devices in close proximity with the client.

- **IP-Based Passive Measurement Techniques -** Passive measurement techniques as the name implies, take a passive approach to IP geolocation and location information about from public registries and

free and paid geolocation databases provided by public or private entities. Some of these techniques exploit the IP allocation policies of IPv4 to narrow down a location to a specific region or country. We will discuss these techniques in details in the following sections.

3.3 Communication and Transmission Protocol Attributes

HTTP is the underlying protocol used by the World Wide Web and forms the foundation of its communication. As the name implies, HTTP deals with transfer of hypertext between links or interrelated documents. HTTP is an application layer protocol based on a request response where resources are requested by a client and information in the form of a response is generated and transferred by a server. The most widely used application of HTTP protocol is within browser applications used to surf online Web sites. HTTP defines a structure for managing how messages are transferred between a client browser and a web server, what commands can be issued and what actions must be taken in response to the issued commands. HTTP is an application layer protocol. Information residing within applications and corresponding application layer protocols are more prone to alteration than network protocol properties such as IP addresses. While IP addresses can be spoofed and rerouted, additional resources (e.g. VPN, Proxy) are required to perform such actions. Most of the application layer protocol properties can however be changed on the client without affecting the overall transmission. HTTP is a request response protocol [91]. HTTP request headers contain various fields defining the type of the request, information about the client, the requesting application and properties of the request. Similarly the response message headers generated by the server contain information and guidelines about the server, the properties of the requested document and the response mes-

sage. Information contained in the response message is generated by the server therefore not viable in the geolocation of the requesting client. Some of the information in the request message however can be directly or indirectly used to identify the location of the requesting client. Table 3.1 lists the HTTP header information which could potentially be used by a web server to geo-locate a client. These HTTP request message header fields are discussed in the following sections.

Table 3.1: HTTP header fields which could be used to geo-locate a user

Header Values	**Description**	**Example**
Accept-Language	Lists of language preference of the visiting user	Accept-Language: en-us
Accept-Charset	Used to indicate the character sets acceptable to a client	Accept-Charset: windows-1253
From	The Email address of the user controlling the user agent	From: webmaster@vuw.ac.nz
Via	Informs the server of proxies through which the request was sent	Via: 1.1 www-cache2.ecs.vuw.ac.nz (squid/3.3.11)
User-Agent	Information about the agent initiating the request	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0

3.3.1 Accept-Language

The Accept-Language header expresses the visiting user's preferred language for the retrieved content. Accept-Language settings are set by the user through browser configuration. It is primarily used by multilingual Web sites which provide translated or variation of content based on user region or their preferred language settings. It is also used by many Web sites to derive and provide information in the local format. The local format information obtained indirectly from language settings could be in the form of numeric formats, measurement values or user's time-zones (Figure 3.3). The Accept-Language is expressed as:

```
Accept-Language = "Accept-Language" ":"
   1#( language-range [ ";" "q" "=" qvalue ] )
language-range = ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) | "*" )
```

Figure 3.3: Accept-language header field format

Each language is separated by a comma (,) and an optional quality value which denotes an estimate of user's preference for the language (min. 0 max. 1). For instance de-de, en-gb; q=0.8, en; q=0.6 indicates the preferred languages of the user in the following order: 1) German, 2) British English valued at 0.8/1 and 3) International English valued at 0.6/1. It should be noted that the Accept-Language header values only indicate the preferred choice of language and content for the visiting user and impose no restrictions or rules for the server to follow. The Accept-Language header values can be extracted using various server side languages. Several server-side languages provide built-in APIs to process and deliver language-specific content based on Accept-Language header values. The .NET framework, for instance, determines the default language, country, date-time and currency format to use with a Web site or application through the built-in CultureInfo Class which parses Accept-Language header values. The following are examples of simple codes to achieve language and geolocation on server-side using PHP and Java Servlet (Figure 3.4 and 3.5). Upon detection of the language and corresponding country, further actions such as redirection or content modification can be performed.

```
if (subster($_SERVER["HTTP_ACCEPT-LANGUAGE"], 0, 2) == "es"
   {
  header(Location: "http://spain.domain.com");
   }
```

Figure 3.4: Redirection to a specific domain using HTTP "Accept-Language" header field with PHP

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Locale;

public class GetLocale extends HttpServlet{
 public void doGet(HttpServletRequest request,
    HttpServletResponse response)
     throws ServletException, IOException
 {
    //Get the client's Locale
    Locale locale = request.getLocale();
    String language = locale.getLanguage();
    String country = locale.getCountry();
 }}
```

Figure 3.5: Java Servlet code to detect locale, language and country based on Accept-Language header values

The HTTP Accept-Language header alone is not a reliable indication of the locale or location of a visiting user as several factors undermine its accuracy.

- A particular language may span several continents and countries with large populations. For instance .es (Spanish) is the official language of Spain and several South American nations (i.e. Argentina, Chile). A web server delivering Spanish Spain-based content will reflect the trending social and political views of Spain which does not serve as appropriate content to visiting users from Chile. The Accept-Language header may include additional strings reflecting the dialect of a language which could reflect a user's region (e.g. es-CL = Chile).

- Accept-Language values are derived either from the operating system locale or set upon installation of the browser. They do not reliably reflect the language of the visiting user.

- Users accessing a Web site through public hosts or restricted environments may not have adequate permissions or knowledge to change

the language values. The delivered content may then reflect the administrator's preference or the spoken language rather than the location.

- The Accept-Language header could potentially be removed or altered by intermediary devices (e.g. proxy)
- Users may enter inaccurate Accept-Language header values through modifying browser settings.
- The Accept-Language values may indicate the language preference of a user located outside the particular region where the language is prominently spoken.

3.3.2 Accept-Charset

Accept-Charset specifies the set of characters acceptable or preferable to a visiting client in which the response generated by the server should be encoded. The value could include several character sets. Similarly to Accept-Language header values, a quality (q) value indicates the order of preferences in the character set. If no Accept-Charset header is present in the request message, it indicates to the server that any character set is acceptable. Generally Accept-Charset depends on the locale set within the user's browser setting (Figure 3.6). Modern browsers however do not transmit this header. Accept-Charset is expressed as:

```
Accept-Charset = "Accept-Charset" ":"
   1#( ( charset | "*" )[ ";" "q" "=" qvalue ] )
E.g. Accept-Charset: iso-8859-5, unicode-1-1;q=0.8
```

Figure 3.6: HTTP Accept-Charset header field format and example

Accept-Charset can potentially be used by a web server to determine the language preference and subsequently the client's region. Since there is

a limited number of character sets, the value obtained by the web server can be matched against the list of available datasets to estimate the language and region of the client. For instance charset=windows-1255 represents Hebrew and could indicate a user from Israel, while iso-8859-7 and windows-1253 point to a user whose preferred encoding and locale is Greek. Geolocation detection issues existing in Accept-Language header values are also inherently present in Accept-Charset header attributes.

3.3.3 From

The "From" header field contains the email address of the user who is initiating and controlling the user agent's request message. It can be used for logging purposes to identify the source of the request in case an invalid request is received by the server. The field is however not obligatory and many modern browsers installed on end-user systems omit inclusion of this header for privacy reasons. The format of the From header field is shown in figure 3.7)

```
From = "From" ":" mailbox
#E.g. From: webmaster@vuw.ac.nz
```

Figure 3.7: HTTP "From" header field and example

A request message sent from a host to a web server may, for instance, contain the email address of masood.mansoori@vuw.ac.nz to identify the requesting client. Geolocation information can be derived from the email address using its top domain part and so the requesting user is assumed to be located in New Zealand. While true in this case, similarly to any user input values, there are a few issues undermining its reliability:

- Email address may only specify the current user and not the actual location of the system. A user with .nz email address might be using a system in an entirely different region.

- From field may contain email addresses with a generic rather than distinguishable regional attribute or information (e.g. mansoori@gmail.com).
- User may enter false information
- The header value can be intercepted and altered through intermediary devices (e.g. proxy). Proxy servers in general may alter the ”From” header value to represent the webmaster while accessing external web servers. This practice allows debugging and error resolution by administrators and webmasters in cases where an internal user makes repetitive invalid requests.

3.3.4 Via

Via field informs the web server of any intermediary protocols and proxies through which the client's request was sent. Its usage includes tracking messages and identifying protocol capabilities. Via is an obligatory field which must be set by intermediary devices. Its usage however is not enforced by any entity and solely depends on the decision of proxy and intermediary device administrators. The ”Via” header format is shown in figure 3.8.

```
Via = "Via" ":" 1#( received-protocol received-by [ comment ] )
received-protocol = [ protocol-name "/" ] protocol-version
protocol-name = token
protocol-version = token
received-by   = ( host [ ":" port ] ) | pseudonym
pseudonym     = token
```

Figure 3.8: HTTP ”Via” header field format

Similar to the ”From” field, any geolocation information within the given Via values may expose a user's location. This holds especially true for

users within large public organisations whose email addresses and profiles may indicate their geographical location. Victoria University of Wellington (VUW) for instance routes all internal Computer Science Faculty packets through a proxy which can be used by a Web site (e.g. www.xhaus-

e.com) to identify its location through the .nz value.

3.3.5 X-Forward-For

This header field is a non-standardised optional value used mainly to identify the IP address of the requesting client to the server in connections that pass through intermediary devices such as proxies, cache servers or load balancers. The idea behind this field is to minimise loss of information belonging to the client system when the connection passes through an intermediary device which might alter the request header values. Web servers may need the actual information of the requesting client rather than those of intermediary device for diagnostics, access control and abuse management purposes. Information available in the X-Forwarded-For value can expose the IP address of the requesting client and subsequent geolocation through various IP-to-Location techniques. Administrators may however restrict the transmission of such information to remote Web sites. Table 3.2 shows the VUW proxy (130.195.199.196) forwarding the IP address of the visitor (130.195.11.136) to the remote Web site (www.xhause.com). A remote Web site can perform a simple IP to geolocation on 130.195.11.136 to determine the location of the requesting agent as a client at Victoria University of Wellington, New Zealand.

Table 3.2: HTTP header information of a client at Victoria university of Wellington, captured by a remote server

Request parameter	**Value**
Requested URI	/headers
Request Method	GET
Remote IP Address	130.195.199.196
Remote IP Port	56760
Protocol version	HTTP/1.1
HTTP Header	Value
Accept	text/html,application/xhtml+xml,application/xml; q=0.9,*/*;q=0.8
Accept-Encoding	gzip, deflate
Accept-Language	en
Cache-Control	max-age=259200
Connection	keep-alive
Host	www.xhaus.com
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:35.0) Gecko/20100101 Firefox/35.0
Via	1.1 www-cache2.ecs.vuw.ac.nz (squid/3.3.11)
X-Forwarded-For	130.195.11.136

3.3.6 User-Agent

The User-Agent field provides information about the agent making the request. It generally includes the name of the agent (e.g. browser), operating system and its version, browser version and comments. The User-Agent format does not specify any variables which may identify a user's location but may include the language accepted by the client (e.g. en-us) and, similarly to the Accept-Language field, the information can be parsed by the server.

3.4 Client-Based Techniques

Variables exposing the location of a user may also be obtained by a web server through running scripts within the client's browser. Browsers and scripting languages have access to the operating system and therefore information such as IP address, time-zone, user and system languages can be directly extracted from the operating system. Variables obtained from the browser are also harder to spoof than those of the HTTP protocol. The following section presents a brief discussion of methods by which geolocation related information can be obtained using a scripting language from within a user's browser agent. The most popular client-side scripting language is JavaScript. JavaScript's popularity can be credited to its capabilities as a client and server-side enterprise web application language. Its ability to run on the client side makes it an ideal language to support a large number of users through reducing load on the server. Providing detailed explanations of JavaScript's capabilities is beyond the scope of this research, but its support of all modern operating systems and web browsers make it an ideal platform for data retrieval and manipulation and transfer from within clients' browsers. The location of a visiting user can be estimated through running JavaScript code to access the locale and time-zone of the client's operating system.

3.4.1 Locale and Language

Locale and language settings can be detected and parsed using client-side scripting languages, primarily JavaScript. JavaScript does not have access to the values of HTTP header fields. It may however obtain similar values using its embedded navigator object to determine browser, user and operating system language and locale properties from within the user's browser. These methods are:

- window.navigator.language(s) = the preferred language of the user, usually the language of the browser user interface

- window.navigator.browserLanguage = Language of the operating system's user interface
- window.navigator.userLanguage = Regional and language settings of the operating system
- window.navigator.systemLanguage = Language of the installed operating system

These JavaScript methods, however, do not return matching results when executed in different browsers on a single system. Table 3.3 shows variations in results obtained from the above methods for several popular browsers. Although JavaScript cannot directly access the values of HTTP headers, the returning string by window.navigator.languages function matches HTTP Accept-Language values, retaining their order based on quality parameters in Chrome and Firefox browsers.

Table 3.3: Locale settings detected by executing JavaScript within popular browsers

Operating system and browser language variables	
Operating System	Windows 8.1
Operating System's installed language	English US, en-us
Operating System's user language	English Great Britain, en-gb
Operating System formatting settings	English Canada, en-ca
Browser Language	English Australia en-au, English South Africa en-za, English en

Popular browsers

	Chrome	**Firefox**	**IE**	**Safari**
window.navigator.language	en-gb	en-au	en-ca	en-us
window.navigator.languages	en-au, en-za, en	en-au, en-za, en	undefined	undefined
window.navigator.browserLanguage	undefined	undefined	en-us	undefined
window.navigator.userLanguage	undefined	undefined	en-ca	undefined
window.navigator.systemLanguage	undefined	undefined	en-gb	undefined

As observed above, there is no reliable technique to extract regional settings from within the browser using a client-side scripting language, as variations in the results point to several regions. A combination of the these methods, however, could give an indication of the language settings used by the user and help malicious web servers pick a malware based on the local language rather than social, economic or current trends of a particular region. Figure 3.9 shows an example of a JavaScript code which extracts a user's preferred browser language through a combination of the "navigator.language" and "navigator.userLanguage" methods to maximise compatibility with popular browsers.

```
<script type="text/javascript">
   var userLang = navigator.language || navigator.userLanguage;
   alert ("The language is: " + userLang);
</script>
```

Figure 3.9: JavaScript excerpt to extract language settings from a visitor's web browser

3.4.2 Time-Zone

The time related fields within the HTTP header (Accept-Datetime [92], Date) do not indicate and transmit the client's time and date for when

the message was originated. The HTTP protocol simply does not transfer user's time and time-zone information across to web servers. System time related information can however be captured through execution of JavaScript code. JavaScript has access to system time information through native support and is capable of obtaining and transferring the information back to remote web servers. There are a few methods by which date-time information is retrieved. The most common method is through implementation of JavaScript's native getTimezoneOffset() of the Date object. This implementation however returns date as an integer offset to UTC (Co-ordinated Universal Time) and does not consider daylight saving. Subsequent conversions are therefore required for accurate results. Libraries such as jsTimezoneDetect [93] on the other hand manage daylight saving and return client's time zone based on IANA zone info key, and is standard for most platforms (e.g. Unix, Linux, Mac). This library accurately returned time zones for UAE, Iran, Malaysia, Germany, UK, Japan and New Zealand in our examination.

Time zones can be a vague indicator of a visitor's region. Time zones are based on lines of longitude which may span across several countries and continents. For instance Cape Town/South Africa, Angola, Chad, Libya, Rome, Germany, Denmark, Sweden and Hungary will all be identified as +1 UTC (Central European Time). Google Geolocation API is a free service which allows Web site administrators to locate a visiting user through its developer AJAX API script. Loading and executing simple JavaScript code in the client's browser which points to the API's interface returns information such as latitude, longitude, country, city, region and postal code – if available – associated with a client's IP address. The precise method by which this information is obtained is not clear, however it is believed such data are obtained through a combination of GPS, cell tower locations, Wi-Fi hotspots and user-submitted data. For instance, nearly a billion mobile devices run Android and built-in Google Maps services. Enabling location information can potentially track and identify

users with great accuracy using GPS. Assisted GPS enhances the quality and precision of the data significantly through triangulation using signal strength based on calculating a phone's distance from cell towers or Wi-Fi hotspots it is connected to. These techniques can be used alone, or in combination to improve accuracy [94].

A combination of GPS, cell tower or public Wi-Fi Access points can pinpoint a user's location to a small region within a city block. Location-enabled google services share users' location information obtained from one or multiple sources with Google to enable provision of adequate and accurate services. Figure 3.10 shows the simplicity of integrating Google's Geolocation capability into a Web site.

```
<script type="text/javascript">
  if(google.loader.ClientLocation) {
    var latitude = google.loader.ClientLocation.latitude;
    var longtitude = google.loader.ClientLocation.longitude;
    var city = google.loader.ClientLocation.address.city;
    var region = google.loader.ClientLocation.address.region;
    var country = google.loader.ClientLocation.address.country;
    var countrycode = google.loader.ClientLocation.address.
        country_code;
  } else {
    // ClientLocation not found or not populated
    // so perform error handling
  }
</script>
```

Figure 3.10: Geolocating users using Google's geolocation API

3.4.3 HTML 5 Geolocation API

Currently there are no unified standards for users to propagate their location information and for web services to retrieve them accordingly. Nearly all accurate geolocation services are offered by third party companies. HTML5 intends to change this situation using a built-in universal geolocation API [95]. According to the draft of the RFC, sources of location infor-

mation will include Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, Wi-Fi and Bluetooth MAC addresses and GSM/CDMA cell IDs, as well as user input. The API is however designed around user privacy, ensuring no location information will be available without the user's consent.

Latitude, longitude and altitude information are currently specified as returned values to a requesting service. Information such as country, city and region can subsequently be retrieved from latitude and longitude using coordinate conversion APIs and services (e.g. Google Maps) with high accuracy. This process is called reverse geo-coding. HTML5 geolocation API is currently supported by all major web browsers. Enabling the API in Firefox version 35, for instance, returns Latitude: -41.2905188, Longitude: 174.76791649999998 pointing out to my precise location at Victoria University of Wellington, Kelburn Wellington.The accuracy of devices using HTML 5 geolocation is highly dependent on the type of technology available in the device used to collect the information. It is assumed that devices having GPS could reach accuracy of 10 meters (open area) while Wi-Fi hot spots and cell towers have an estimated accuracy of 20 and 1000 meters respectively [96].

3.5 IP-Based Approaches

IP Geolocation is the process of discovering the geographical information of a given IP address, including, the country, city, longitude and latitude. Depending on the requirement of the application and type of the measurements used, varying degrees of information with varying degrees of accuracy may be obtained. There are few fundamental challenges in associating an exact location to an IP address on the Internet.

- The Internet is a dynamic environment where IP address of network subnets and their corresponding end systems change drastically in

response to changes in infrastructure. Although large address spaces assigned to large geographical areas are barely altered, smaller subnets assigned to local ISPs and organisations within a region can change constantly.

- There are no standards and protocols which provide reliable geographical positioning information for every IP address. Information provided by public DNS or WHOIS servers managed by Regional Internet Registries may contain outdated or inconsistent information about an IP address. Decentralised management of the Internet ensures that there is no reliable and centralised database of host locations.

- Availability and accessibility to a large database which points out the exact location of users raises huge privacy and security concerns.

These fundamental challenges may have stopped the development of a universal protocol for user identification around the world, but have not stopped companies and researchers developing techniques to determine a geographical location of a user, with various reliability rates. The following section presents a brief discussion of various techniques used in geolocation of IP addresses. These techniques are mostly focused on IP address geolocation of devices on the Internet. Various properties of an end user may expose its location depending on the type of technology used. However the assumption in using these techniques is that IP address is the only information available to a monitoring system. IP geolocation is generally based on two main approaches using active and passive measurements. Figure 3.11 Summarises the most widely used passive and active IP-based geolocation techniques.

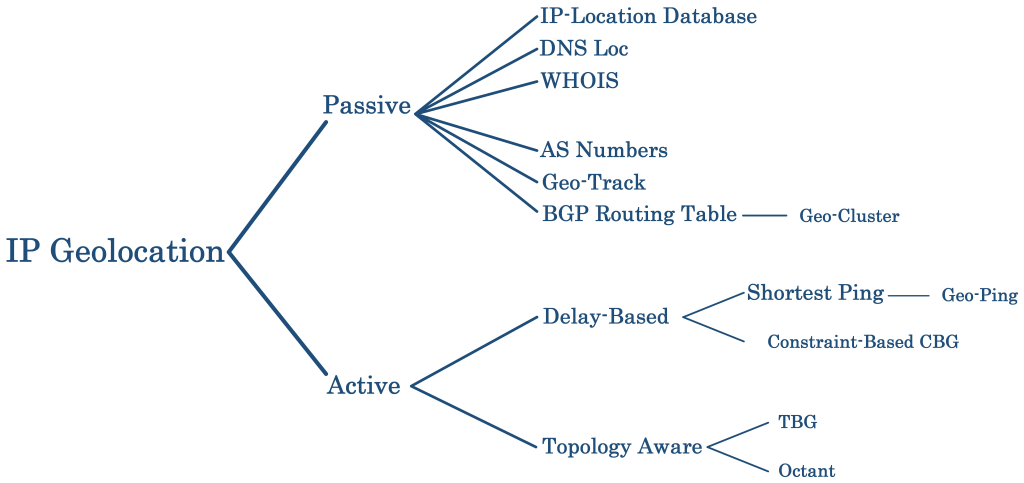


Figure 3.11: Most popular IP to location techniques

3.6 Active Measurements

Active techniques rely on probes and active queries to measure the target's location for each node. These techniques rely on a mapping process which uses the publicly available geographical location of IP address or adjacent network routing devices to lookup its respective geographic location Active measurement techniques can be further divided into multiple categories. These techniques rely on a combination of information gathered from network routing, positioning information for fixed points referred to as landmarks and timing information between host, landmarks and target IP address, to determine the location of a target host.

3.6.1 Shortest Ping Technique

This is the simplest form of active geolocation probing which relies on a series of delay measurements in the form of ICMP Ping messages from a set of landmarks to a target IP address. The location of the target IP address is then mapped to the location of the landmark with the shortest

delay time. The accuracy of this technique is dependent on the number of landmarks and the distance between the target and the nearest landmark used in the measurement [97].

Geo-Ping [98], an algorithm based on the shortest ping relies on multiple landmarks and reference nodes. Initially a delay map is constructed using the minimum delays between known landmarks and the reference nodes. A delay vector is then measured between each landmark and target IP address. The location of the target IP address is the landmark which has the lowest Euclidean distance between the delay vectors of the target IP address and landmarks. Similarly to the shortest ping geolocation algorithm, the accuracy of this technique is dependent on the number of landmarks and the distance between the target and the nearest landmark used in the measurement. Network congestion can also introduce significant noise and subsequent inaccuracy into the measurement [99].

3.6.2 Constraint-based Geolocation (CBG)

Constraint-based Geolocation (CBG) [100] approach determines the location of a target host using a process for estimating a position based on a sufficient number of distances to some fixed points. This process is called multilateration. CBG initially uses delay-based probes from landmarks and delay-to-distance relationships to calculate the maximum distance for a given delay. Delays are translated into distances using a bestline approach which considers the best network condition for each landmark. These estimations are then used to draw circles around each landmark. The centre of the region generated from the intersection of all circles around landmarks is the location where the target host is most likely to be positioned. CBG's accuracy is highly dependent on the number and proximity landmarks to the target [98, 101].

3.6.3 Topology-aware Geolocation

Delay-based geolocation techniques fail to consider scenarios such as circuitous end-to-end paths in which the correlation between delay and distance between landmarks and landmarks-to-target can vary significantly. Topology-aware geolocation attempts to resolve this issue by limiting the impact of circuitous end-to-end paths [102]. Topology-based geolocation (TBG), proposed by [98], considers network topology based on the fact that routers adjacent to a target host can be more accurately geolocated. Initially the furthest possible target location is calculated based on the maximum transmission speed of a packet in fibre. This gives a conservative estimate of the possible region in which the target might be located.Topology and inter-router latencies from landmarks to target are discovered based on single-hop link delay estimation using traceroute and interface clustering applications. Taking the delay between intermediate routers into account narrows down the possible location of a target.

Octant, another variation of topology-aware geolocation enhances accuracy by combining topology-aware geolocation with other techniques such as DNS and WHOIS lookups. It also considers positive and negative information in its estimation of the target location. Positive information refers to the maximum distance at which the target may be located from the landmark based on latency calculation. The negative information refers to the regions where the target will certainly not reside based on maximum latency estimation and geographical and demographical constraints such as uninhabitable areas (e.g. Deserts, Oceans) [103]. Several issues makes measurement-based techniques undesirable for small web applications where accuracy is not as critical:

- Topology-aware geolocation techniques require significant resources to achieve their claimed accuracy [104]. Availability of a large number of known fixed landmarks – either active or passive – in regions around the world is essential in measurement-based detection tech-

niques.

- Topology-aware geolocation techniques require significant computational time to estimate a location for an IP address. They must construct the network topology first and require probes in the form of ping and traceroute packets to calculate delay-to-distance and obtain information from intermediary devices [105]. These probes will undoubtedly add between a few seconds to several minutes calculation time for a single IP address (excluding the computational time) to the process [106]. Such delays would not be acceptable in a Web site visiting scenario.

- Maintaining a large up-to-date database of IP addresses with high accuracy obtained from measurement-based techniques requires a level of resources neither feasible nor cost effective for an attacker.

- Measurement-based techniques rely on active measurement probes, assuming a reply is to be received from the target host [107]. In the real world however, many hosts and routers are configured to detect or drop ICMP packets [104].

Delay and topology-based techniques are sensible approaches for large organisations and geo-precision applications where accuracy is a key factor and overhead time and resources are justified to reach it. Applications which need detection speed for prompt service delivery in exchange for reduced confidence in accuracy must rely on alternative techniques that utilise passive approaches. A few such techniques and services are discussed in the following section.

3.7 Passive Measurements

Passive geolocation measurements are achieved using prepopulated geolocation databases containing a large number of IP blocks, prefixes, ad-

dresses and their corresponding geolocation information. Geolocation in passive measurement relies on queries performed on the databases through direct connection or API. Such approaches are designed for applications and services where accuracy is not vital, while low measurement overheads and low response times are essential. These properties make database-driven passive geolocation suitable for web server applications. The reliability and amount of information contained in these databases can vary depending on the type of service.

Managing the distribution and assignment of large IPv4 and IPv6 addresses is performed based on a hierarchical structure which involves several organisations, each responsible for a large geographical region. These organisations form a framework for global Internet governance. Their responsibility is to build Internet standards, distribute resources and facilitate policy development.

IANA (Internet Assigned Numbers Authority) is the highest entity responsible for distributing large blocks of address space, with five subsidiary organisations called Regional Internet Registries (RIR) corresponding to the five major continents of the world. These large blocks are subsequently distributed to Internet Service Providers (ISPs) and Local Internet Registries for further distribution. Figure 3.12 shows the hierarchical structure of IPv4 and IPv6 management and associated organisations.

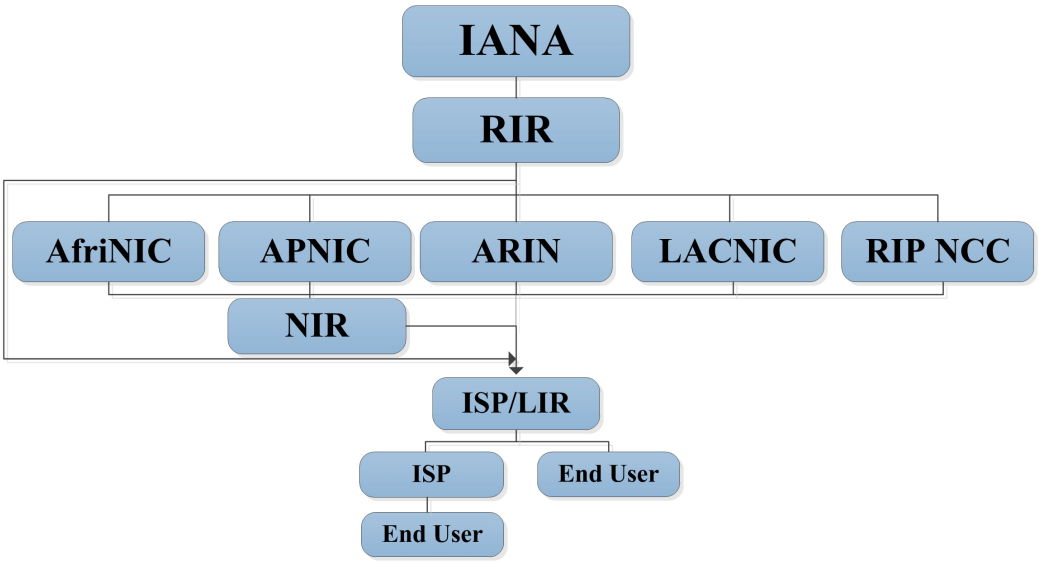


Figure 3.12: Hierarchy of IP block address allocation from IANA to end user

IANA: is a non-profit organisation responsible for managing the allocation of global IP address blocks to regional entities for further distribution. It also oversees the registry and administration of DNS root zones, domain names (e.g. .us, .nz), URI schemes (e.g. http, https), Internet protocols and the corresponding ports and character encodings used in Internet protocol suits. IANA is the highest entity that manages IP allocation and distribution. The IANA will allocate IPv4 address space to the RIRs in /8 blocks corresponding to 16,777,216 hosts per subnet. The minimum IPv6 block allocated by the IANA to an RIR is a /12 address space corresponding to 1,329,227,995,784,915,800,000,000,000,000,000,000 IP addresses (Table 3.4) [108, 109].

RIR: Regional Internet Registries are smaller organisations operating in five major continents. They are authorised by the IANA and distribute and coordinate IP and AS allocation in their respective regions. These organisations allocate addresses to companies or end users through Local Internet Registries and large Internet Service Providers. Their policies and activities are regulated by a coordinating body called the NRO (Number

Resource Organisation). Regional Internet Registries generally do not assign IP addresses directly to end users unless they meet certain requirements. ARIN (American Registry for Internet Numbers) for instance allocates blocks of addresses to users who meet criteria such as a minimum assignment of /24 address space, 25% immediate utilisation and immediate need. Users failing to meet the criteria must obtain address space from their Local Internet Registry (LIR) or Internet Service Providers within that particular RIR. Figure 3.13 shows the 5 major regional Internet registries responsible for IP allocation and coordination.

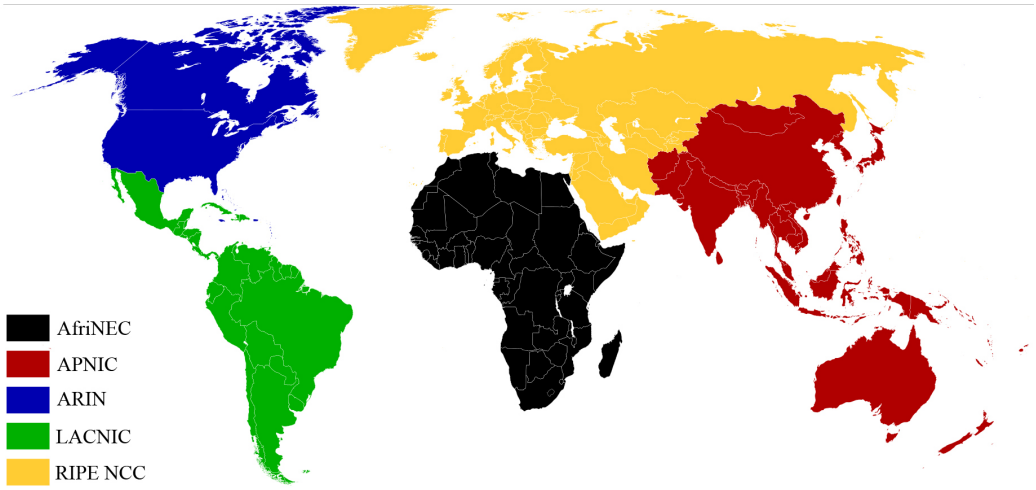


Figure 3.13: Regional Internet Registries (RIRs) corresponding to associated continents

NIR: National Internet Registries (NIR) exist mostly in the Asia Pacific region (APNIC) and manage smaller regions in comparison to RIRs. They are responsible for allocating address spaces to members who are generally LIRs organised at a national level. APNIC covers the Asia Pacific's four regions: South, Southeast, Central-east, and Oceania. LIR: Local Internet Registries are generally Internet Service providers which assign IP addresses directly to end users or other smaller ISPs.

Table 3.4: Excerpt of allocated and reserved IPv4 blocks

Prefix	Designation	Date	WHOIS	Status
IPv4				
000/8	IANA	1981-09		RESERVED
001/8	APNIC	2010-01	whois.apnic.net	ALLOCATED
002/8	RIPE NCC	2009-09	whois.ripe.net	ALLOCATED
...				
174/8	ARIN	2008-02	whois.arin.net	ALLOCATED
175/8	APNIC	2009-08	whois.apnic.net	ALLOCATED
176/8	RIPE NCC	2010-05	whois.ripe.net	ALLOCATED
177/8	LACNIC	2010-06	whois.lacnic.net	ALLOCATED
178/8	RIPE NCC	2009-01	whois.ripe.net	ALLOCATED
179/8	LACNIC	2011-02	whois.lacnic.net	ALLOCATED
...				
239/8	Multicast	1981-09		RESERVED
240/8	Future use	1981-09		RESERVED
241/8	Future use	1981-09		RESERVED
IPv6				
2001:0000::/23	IANA	1/07/1999	whois.iana.org	(Reserved, IETF Protocol
2001:0200::/23	APNIC	1/07/1999	whois.apnic.net	ALLOCATED
...				
2001:0e00::/23	APNIC	1/01/2003	whois.apnic.net	ALLOCATED
2001:1200::/23	LACNIC	1/11/2002	whois.lacnic.net	ALLOCATED
2003:0000::/18	RIPE NCC	12/01/2005	whois.ripe.net	ALLOCATED
2400:0000::/12	APNIC	3/10/2006	whois.apnic.net	ALLOCATED
2a00:0000::/12	RIPE NCC	3/10/2006	whois.ripe.net	ALLOCATED
...				
3000:0000::/4	IANA	1/07/1999		RESERVED

3.7.1 WHOIS-based Mapping

WHOIS is a simple query/response protocol based on TCP/IP which allows users to retrieve information on domain name registries, IP address allocations and registries, and IP routing registries (AS number allocations). It comprises databases distributed over servers located and managed by each Regional Internet Registry (RIR). The main WHOIS servers of the top Regional Internet Registries are: whois.afrinic.net, whois.apnic.net, whois.arin.net, whois.lacnic.net, and whois.ripe.net [110]. These servers can be queried by clients using a query/response protocol based on TCP/IP operating on port 43.

The most common usage of a WHOIS service is by average users to determine domain availability and find the entity to which the resource has been allocated. WHOIS is widely used by administrators to find contact information regarding the registrar of a resource. This allows them to resolve technical issues related to DNS errors, routing and overall stability of the Internet. The information also helps authorities deal with trademark infringements, and combat distribution of spam, phishing or illegal content.

WHOIS provides certain information to the public regarding a registered resource. ICANN policies specifically dictate that registrars comply with the WHOIS Accuracy Specification policy and provide accurate contact information upon registration. The required and verified information includes registered name, email addresses, phone numbers and postal address of the registrar and administrative contact [111]. Figure 3.14 shows the public information which can be obtained from a WHOIS service. Country, address, PO Box number, phone number, and top-level domain postfix of an email address can pinpoint the geographical location of a user.

```
\% [whois.apnic.net]
\% Information related to '130.195.0.0 - 130.195.255.255'
inetnum:    130.195.0.0 - 130.195.255.255
netname:    VUW
descr:      Victoria University of Wellington
country:    NZ
admin-c:    VA53-AP
tech-c:     VT91-AP
status:     ALLOCATED PORTABLE
mnt-by:     APNIC-HM
mnt-lower:  MAINT-NZ-VUW
mnt-routes: MAINT-NZ-VUW
remarks:
   ------------------------------------------------------
remarks:    To report network abuse, please contact mnt-irt
remarks:    For troubleshooting, please contact tech-c and admin-
   c
remarks:    Report invalid contact via www.apnic.net/
   invalidcontact
remarks:
   ------------------------------------------------------
role:       VUW ADMIN
address:    Victoria University of Wellington
address:    PO Box 600
address:    Wellington
country:    NZ
phone:      +64-4-463-5931
e-mail:     its-service@vuw.ac.nz
admin-c:    PM380-AP
tech-c:     SW1143-AP
nic-hdl:    VA53-AP
mnt-by:     MAINT-NZ-VUW
changed:    asjl@lpnz.org 20081126
changed:    asjl@mcs.vuw.ac.nz 20081216
source:     APNIC
\% This query was served by the APNIC Whois Service version
   1.69.1-APNICv1r7-SNAPSHOT (WHOIS3)
```

Figure 3.14: WHOIS data on VUW IP address subnet

Comparable information can be obtained about the location of a target device using AS numbers. Autonomous Systems are a network or group of networks managed by a single entity or organisation. Autonomous System (AS) numbers are allocated to Regional Internet Registries who assign

them to organisations located within their prospective regions. This information is available for the public to access in the WHOIS databases in each RIR. Each public IP address or range of IP addresses belong to an AS number. Performing a WHOIS query on the AS number within the WHOIS database can reveal registration and assignment information, including the geolocation data for an IP address. While this information may seem adequate to geolocate a registered entity for an IP address or a resource, there are a few issues that should be considered:

1. IP addresses or resources are sold to organisations, companies and wholesalers who distribute them to a large number of end users or retailers. The end users could be located in a vast geographical area spanning a state or a small country. Obtaining additional information regarding the sub-registration of registered entity involves communication with local registries or Internet service providers.

2. The registration information may be incorrectly provided by the registrants.

3. Information provided in the DNS record may not represent the definite location of a user or domain. Large companies provide Top-Domain Level (TDL) domains (e.g. .uk) to reach a specific market and provide services to users in a geographical location. However, the company registering the domain and its servers could be located in an entirely different region or country. A large number of organisations place their servers within United States because of the availability of reliable hosting companies and reduced costs.

4. Not all IP addresses necessarily map to a specific domain name.

3.7.2 GeoTrack

IP geolocation using this technique is achieved through performing a Traceroute (i.e. ”Traceroute” on Linux and ”Tracert” on Windows systems) on

the targeted IP address and obtaining DNS information for the intermediary devices leading to the target device. The location of the router closest to the target IP address is an estimation of the location of the target host.

Traceroute reveals the information about the path and intermediary devices a packet takes to reach its destination. The idea is to locate the last identifiable intermediary device with recognisable geolocation information. Traceroute determines the route to a destination IP address by sending Internet Control Message Protocol (ICMP) echo packets. It uses varying IP Time-To-Live (TTL) values in its packet headers which act as a hop counter. For every intermediary router the packets travel through to reach the destination IP address, the TTL value is decreased by 1 and a "Time Exceeded" message is returned. Traceroute forwards multiple packets with incremental TTL values until the destination responds or until the maximum TTL is reached. It then displays an ordered list of intermediary routers with DNS information (if applicable), which can be utilised to find the location of the network subnet and subsequently the IP address. Traceroute operates by sending active messages to destination intermediary devices; however it primarily relies on information saved on intermediary devices to estimate the location of a target user and is therefore categorised as a passive geolocation technique.

Table 3.5 shows the information obtained from intermediary devices using traceroute for a generic domain address of "tvgids.tv". Final intermediary device "amstnl02.nl.bb.gin.ntt.net" contains the country code ".nl" representing Netherlands, in which the target domain is hosted. Although this approach returns geolocation information about a target in most cases, there are however the following issues with such an approach:

1. DNS information must have been entered into the router
2. Reliable DNS lookup information must have been entered into the router
3. Depending on the configuration, some routers are configured not to

forward ICMP or ICMP error messages (i.e. Time Exceeded)

Table 3.5: Tracert information for tvgids.tv which is located in Netherlands (.nl)

Name of the System (DNS)	Roundtrip 1	Roundtrip 2	Roundtrip 3
***			
210.7.39.248	8.932 ms	8.798 ms	8.747 ms
xe-0-2-4.bdr1.a.sjc.aarnet.net.au	178.883 ms	162.374 ms	163.438 ms
ae-7.r23.asbnva02.us.bb.gin.ntt.net	222.116 ms	228.794 ms	223.067 ms
bit-0.r01.**amstnl02.nl**.bb.gin.ntt.net	318.190 ms	314.961 ms	318.819 ms
***			

3.7.3 DNS LOC Mapping

DNS LOC (RFC 1876) is a voluntary and experimental protocol allowing users and organisations to propagate geographic location information for a domain name though DNS records information. The information contained in the LOC record includes latitude, longitude, altitude and the diameter of a sphere containing the geographical domain (Figure 3.15). LOC information is provided by the users and organisations so its reliability is questionable. The LOC information also generally reveals the geographical coordinates of the headquarters of a domain, rather than the physical area a domain covers. A LOC record could pinpoint a domain which serves a large number of users who are geographically dispersed across a wide area. Less than 1% of all domains provide locational information in their DNS LOC data. LOC information is nearly non-existent for local users IPs and small ISPs. The LOC record is expressed in a master file in the following format:

```
<owner> <TTL> <class> LOC ( d1 [m1 [s1]] {"N"|"S"} d2 [m2 [s2]]
                     {"E"|"W"} alt["m"] [siz["m"] [hp["m"]
                     [vp["m"]]]] )
where:

  d1:   [0 .. 90]       (degrees latitude)
  d2:   [0 .. 180]      (degrees longitude)
  m1, m2: [0 .. 59]     (minutes latitude/longitude)
  s1, s2: [0 .. 59.999] (seconds latitude/longitude)
  alt:  [-100000.00 .. 42849672.95] BY .01 (altitude in meters)
  siz, hp, vp: [0 .. 90000000.00] (size/precision in meters)
```

Figure 3.15: DNS LOC format

Figure 3.16 shows the DNS LOC record for Waikato University in New Zealand which contains latitude and longitude information.

```
Domain: waikato.ac.nz
DNS LOC record: 37 47 16.000 S 175 19 4.000 E 64.00m 1000m 10m 10
    m
Latitude: -37.787777777778
Longitude: 175.31777777778
Altitude: 64m
Size: 1000m
Horizontal precision: 10m
Vertical precision: 10m
```

Figure 3.16: LOC information of Waikato university domain

3.7.4 GeoCluster

GeoCluster uses information available in Address Prefixes (APs) of BGP routing tables to construct a topological cluster that corresponds to a set of hosts most likely to be located within a similar region, using longest prefix match. This information is then combined with partial IP-to-location mapping information gathered from multiple online sources such as a Web-based email site or a business Web hosting site to determine the likely location of the cluster [99].

3.7.5 Autonomous System (AS) Numbers

Autonomous System number [112] is a globally unique number assigned to a set of network subnets and routers operated by one or more network operators under a single administrative domain. An AS domain uses an interior gateway protocol (IGP) to route packets within the domain, and an exterior gateway protocol (e.g. BGP) to route packets to other autonomous systems. An AS number is a global identifier for a specific autonomous system and used in the process of exchanging exterior routing information between neighbouring autonomous systems. AS numbers are 16-bit integers, assigned and managed by the IANA.

Not every network or network subnet requires a unique AS number to allow nodes to forward packets externally, as many networks do not have the requirement to express their unique set of routing policies. An AS number can be requested from and assigned by IANA and associated Regional Internet Registries (RIRs) and, as with WHOIS information and domain registration, a registry of AS numbers and related information is created and is publicly available. AS numbers are very limited compared to IPv4 and IPv6 addresses, hence they require firm registration policies and thorough verification of details prior to approval. The assignment and modification of AS information is less frequent than for its WHOIS and domain counterparts, and the associated information is more frequently updated [113]. We therefore assume that such information is more reliable in terms of updatability compared to WHOIS and DNS information.

AS numbers can represent a large domain consisting of several ISPs spanning a large region and include multiple network subnets. Autonomous System numbers are however accurate at the country level. NetGeo is an example of a system utilising a combination of WHOIS and ASN for geolocation of IP addresses [114]. Yook *et. al*. [115] utilised autonomous system numbers to identify all main routers around the world and then model the Internet's large-scale topology.

3.7.6 IP to Location Databases

Free and commercial services utilise the public information available in registry databases, and in combination with other techniques, to create databases which map a large set of IP addresses and subnets to physical locations. A few geolocation databases contain nearly 3-4 billion IPv4 addresses/subnets and their corresponding information. Although the methodology used in collecting this information is unknown, in theory creating and maintaining such databases is possible for an organisation with large resources, even if it involves geolocating IP addresses with active measurements. Creating and maintaining such databases in IPv6 is, however, a much more challenging task considering the 128 bit structure of IPv6 and large size of its allocated subnets. For instance as mentioned earlier the IPv6 block allocation to an RIR is a /12 subnet which corresponds to 83,076,749,736,557,240,000,000,000,000,000,000 IP addresses for 2400:0000::/12 subnet allocated to APNIC (Table 4.6). Current allocation of IPv6 subnets and numbers of devices using the addresses are however limited at the moment which allows integration of the information into manageable databases for geolocation query purposes. In the following section we briefly discuss a few such services.

MaxMind, IPligence and IP2Location are possibly the most well-known pay geolocation services, providing geolocation information directly through their databases and APIs for several programming languages. Their services are priced depending on the amount of information required about an IP address. The information provided by such services includes country, city, zip code, lattitude/longitude ISP, domain or company name, time zone and even connection type. In table 3.6 we show the summary of the acclaimed accuracy of geoinformation provided by the most popular IP to geolocation database services.

Table 3.6: Claimed accuracy rate of IP to location database services

Databases	IPv6 Support	No. of IPs	Countries	Accuracy at Country Level	Accuracy at City Level
MaxMind	Yes	3.6 Billion	250	99.8%	80% (avg.)
IPligence	Yes	2 Billion	242	99%	80%
IP2Location	Yes	–	249	99.5%	75%

There are other high-profile commercial companies active in geolocation of IP addresses, with well-known customers mainly in advertising, region specific web applications, gambling, DDoS and fraud detection. GeoBytes, Quova and Digital Element are a few examples. Digital Elements for instance indicates Facebook, Yahoo, Twitter, CNN, Ebay as its customers. Examples of free services are HostIP.Info, MaxMind's Geolite service, IP2Location LITE databases, IPInfoDB and Software77's Geo-IP. Table 3.7 shows an excerpt from the MaxMind database for several IPv4 subnets. Based on our observation of free geolocation databases, MaxMind's GeoLite and IP2location provide the most information (See table below) whereas IPgillence's free and lite service only display the continent and country information for a particular IP address or subnet.

Table 3.7: Excerpt of MaxMind IP to location database information [2]

Network	Geoname id	Registered country id	Satellite provider	Postal code	Latitude	Longitude
IPv4						
41.223.138.0/24	2318044	2328926	0	0	9.2	12.4833
72.198.201.0/24	5074472	6252001	0	68104	41.2959	-95.9991
84.104.69.24/30	2747373	2750405	0	2563	52.078	4.2922
121.215.154.0/24	2077454	2077456	0	6233	-33.2139	115.7196
177.6.217.0/24	3470177	3469034	0	79260	-22.1	-56.5167
207.226.32.34/32	*	*6252001	1			
210.212.158.240/28	1269743	1269750	0	0	22.7167	75.8333
IPv6						
2001:4270::/32	2318044	2328926	0	0	10	8
2600:1014:b100::/47	5074472	5074472	0	68101	41.2586	95.9378
2a02:29b0:400::/38	2747373	2750405	0	0	52.25	5.75
2400:c500::/32	2077454	2077456	0	0	-25	135

Geo. id	Continent name	Country code	Country name	Subdivision name	City name	Code	Time zone
2318044	Africa	NG	Nigeria	Adamawa	Yola		Lagos
1673825	Asia	TW	Taiwan	Kaohsiung	Gaozhongyicun		Taipei
5074472	North America	US	United States	Nebraska	Omaha	652	Chicago
2747373	Europe	NL	Netherlands	South Holland	The Hague		Amsterdam
2788051	Europe	BE	Belgium	Antwerp Province	Rijkevorsel		Brussels
2077454	Oceania	AU	Australia	Western Australia	Australind		Australia/Perth
3470177	South America	BR	Brazil	Mato Grosso do Sul	Bela Vista		Campo Grande
6252001*	North America	US	United States				
1269743	Asia	IN	India	Madhya Pradesh	Indore		Asia/Kolkata

The main issue with these databases and nearly all other free and paid services is that the geolocation techniques used to create such databases or services are entirely unknown [106, 116]. Several research studies have investigated the accuracy of these databases at country or city level, comparing the measurements of geolocation databases against active measurement techniques. Their findings questions the accuracy of database services especially at the city level. The main problem with these studies is they lack a large and diverse set of geographically known IP addresses to use as the basis for the measurement. The reliability of comparison studies between databases and active measurement can also be questioned, as active measurement techniques are not 100% reliable. Siwpersad *et. al*. [104] examined the accuracy of two geolocation database services (i.e. MaxMind and IP2Location) against the confidence region measured using their CBG with Bandwidth Estimation approach, and concluded that their

resolution is too coarse. Shavitt and Zilberman [116] compared the accuracy of 100,000 IP addresses obtained from several databases including MaxMind, IPligence and web-based services against their own approach, which relied on a combination of PoP classification and delay-based measurement. PoP (Points of Presence) is a group of routers which belong to a single AS number, generally consisting of few backbone and several client routers which are physically located at the same building or campus. Their findings showed that geodatabases have a minimum convergence rate (i.e. one Kilometre) of between 82 to 90 percent. The accuracy of the databases in their ground base experiment which consisted of 25000 known IP addresses provided by CAIDA (Centre for Applied Internet Data Analysis) confirmed greater than 80% accuracy at country level for all services, but less accuracy at city level. According to their findings, the accuracy of geodatabases is sufficient for many location aware applications and even consistent at city levels.

Similarly to [116], authors in [106] examined the accuracy of several commercial and free databases against a set of 380,000 IP ground truth IP addresses and blocks obtained from a backbone European router. The IP prefixed included in the databases achieved an accuracy rate of 96-98 percent at the country level. The city-level accuracy was as expected much lower than claimed. These studies confirm that such services provide precise and decent geolocation estimation at country and city levels respectively. Table 3.8 shows comparisons between various geolocation techniques in terms of their performance, accuracy and required resources. Techniques with high access time are not suitable for real-time geolocation which is generally used in web server applications as they introduce significant delay into a client's response process.

Table 3.8: Comparison of IP to location techniques based on performance, accuracy and maintainability

	Accuracy	Falsifiability	Needed Resources	Updated	Access Time
WHOIS	Medium	Medium	Low	Medium	Low
DNS LOC	Low	High	Low	Low	Low
Traceroute	Medium	Low	Low	High	High
Geo-Cluster	High	Low	Medium	High	Medium
Delay-Based	Medium	Low	High	High	High
Topology-Based	High	Low	High	-	High
IP-Database	Medium	Low	Low	Medium	Low
HTTP Headers	Medium	High	Low	High	Low
Client Browser	Medium	High	Low	High	Low

There are variety of techniques by which a malicious web site may estimate the geographical location of a visiting user, as discussed in previous sections. Malicious web sites may subsequently lure and target users from that specific location using a combination of social engineering and local content. A detection system would have to acquire the system and network attributes, and simulate the behaviour of a user in that specific geographical location in order to be subjected to an attack. Unfortunately available client honeypot systems and associated research mostly focus on improving the detection rate of various state, signature and behaviour based analysis. These are standalone systems and can therefore observe an attack directed at a particular location in which the system is placed. They fail to detect any attack utilising geolocation attributes discussed previously to either target a specific user or perform geolocation cloaking. In a geolocation cloaking attack, benign content is served to visitors who are not requesting the content from the location specified by the attcker or for

which the content is designed.

In the following sections we provide a taxonomy and survey of available client honeypots and investigate their capabilities in terms of detection and specifically geolocation targeted attacks. We finally identify the gap in research on geolocation targeted attacks which will be the basis of this research.

3.8 Client Honeypot Systems

Detection and analysis of malicious web sites is generally performed using decoy systems or applications called honeypots. Honeypots are defined as information security resources whose value resides in being probed, attacked and compromised. In server honeypots, such systems comprise of real or simulated operating systems or services with exposed vulnerabilities which are placed within the software, hardware and network infrastructure of an information system, luring attackers. These bait systems then wait for attackers to target them and in the process gather information about the targeted resources, tools and techniques used by the attackers. Server honeypots are however passive systems which means they are not a part of the operational network and therefore do not initiate any connection and should not be receiving any traffic. Any connection made to a server honeypot is generally considered an attack or suspicious nonetheless. Server honeypots are used to detect attacks on server-side services.

Client honeypots on the other hand take an active approach to detection by actively interacting with a potential attacker. They are specifically designed to detect attacks on an operating system browser and its embedded services. Vulnerable browsers interact with potentially malicious web sites, retrieve the content and detect any embedded attack through analysing the content or the behaviour of the web server using various static and dynamic analysis systems discussed in section 3.8.2. These analysis systems are a part of a general architecture which is common between

all client honeypots. A client consists of a queuer responsible for collecting URLs to be retrieved and analysed, a visitor agent which interacts with web sites, exposes vulnerabilities and allows the web server to target them delivering malicious code. The retrieved content is finally classified by the client honeypot's analysis engine. Client honeypots are generally classified into low and high interaction. Interaction level refers to the capabilities of the client honeypot in terms of simulating real browser's functionalities and user activities.

3.8.1 Low Interaction Client Honeypots

Low Interaction Client Honeypots (LICH) rely on simulated browsers and plugins to mimic the behaviour of a user's client and interact with a remote web server. Due to limited simulation capabilities, LICHs require minimal resources to for retrieval and analysis which makes them ideal for large scale detection. Operating system, browser personality, plugin types and versions are instantly altered to meet a specific environment using the attributes of simulated service and HTTP header values. A low interaction client honeypot does not provide the full functionalities of a real browser therefore requesting a service typically supported by a real browser which is not emulated on a LICH will result in detection of a virtual environment. LICH are run as a software on top of the host operating system and unlike a high end virtualisation software, they cannot be hijacked.

LICHs primarily rely on signature, heuristics and pattern matching techniques for detection in which the retrieved web site content and its components are matched against a database of attack signatures or analysed for patterns of suspicious behaviour (e.g. hidden links and iframes). As LICHs rely on precollected signatures of attacks for detection, they are inherently susceptible to miss attacks which are obfuscated or those for which attack signatures have not been developed.

There are a number of client honeypot systems designed and used by researchers to detect browser based attacks. PhoneyC is a low interaction client honeypot designed on a combination of signature based and vulnerability driven detection. It is comprised of an input collector and an input evaluator component. The input collector utilises Curl library to retrieve a web site content. It mimics popular browser personalities using User-agent headers to collect vulnerabilities designed and delivered to specific browsers. The contents are then scanned and matched against ClamAV antivirus engine. It also isolates dynamic scripts and renders them on corresponding interpreters. Java Scripts are rendered within SpiderMonkey Java Script interpreter while VB scripts are translated to Python scripts and run within a Python interpreter [23].

Script interpretation allows PhoneyC to render dynamic scripts in real-time, allowing it to detect and bypass obfuscation techniques. It also allows PhoneyC to implement its vulnerability driven detection that relies on identifying malicious activity against a vulnerable method (e.g. passing long strings to a method to cause buffer over-flow). Extraction of JavaScripts and payloads from .pdf files are also supported using a simple algorithm that extracts arguments passed to the "unescape" JavaScript function. The unescape method decodes an encoded string.

Honeyware [24] is a low interaction client honeypot written in PHP and runs in a web browser. It provides the ability to simulate almost all commonly used web browsers and utilises 5 different antivirus engines in the analysis process. An important component of Honeyware is integration of two search engines namely Yahoo and MSN APIs to query specific keywords. Honeyware differs itself with other available honeypot implementations by providing the capability to detect geolocation attacks. Honeyware's answer to geolocation and IP-Tracking attacks (MPack attacks) lies in utilisation of a client component which is installed in a different geological location. The main Honeyware server connects to malicious web server and retrieves the content while the client component initiates

several web spider requests. Honeyware then compares the responses for any differences [24]. Honeyware's solution is however, far from simple or efficient in terms of required resources and speed. The first issue resides in the fact that resources in different geological locations are needed to install and maintain the client component. The second concern is the performance in terms of speed by which the Honeyware carries out detection.

Low interaction honeypots are generally faster because they use signature matching on the contrary to state based detection of high interaction client honeypots. In a state based detection, a state of a system is monitored for any changes after a web site visit and needs to be reverted back to a "safe state" (i.e. state before visiting a web site). Honeyware however relies on five different antivirus signature engines for pattern matching. Utilizing a high number of signature engines results in slow detection speed, as mentioned by the authors [24]. A hybrid system, combining the speed and state based detection capabilities of high interaction client honeypots (i.e. Capture-HPC) with the simplicity and detection rate of Honeyware has also been proposed and tested, to overcome the slow detection pace of Honeyware. The hybrid system performs the operation by analysing each specified URL with Capture-HPC and passing the benign URLs to Honeyware for further analysis. In initial tests, Honeyware was able to scan a single URL in 1 minute versus 17 seconds for Capture-HPC. Honeyware however managed to find 83 out of 84 malicious web sites fed into its analysis engine.

Monkey-Spider [25] is another low interaction client honeypot comprising of the typical components of client honeypots, queuer, visitor and analysis engine. The queuer utilises search engine APIs (i.e. Google, Yahoo and MSN), link extraction from SPAM emails and online blacklisted URL databases, to feed the visitor. The search engine APIs allow Monkey-spider to feed URLs to the queuer based upon keywords, most likely to return search results containing malicious links. POP3 Mail Seeder extracts email contents from a POP3 account, writes the content to a single

file and extracts URLs within the file. Suspicious Links are also obtained from free online databases of malicious web sites using a script which is then added to the queue for visit by Monkey-Spider's visitor component .

Unlike other implementation of client honeypots, Monkey-Spider does not operate a web browser to retrieve contents of the target server but rather fetches the content through the Heritrix Crawler. Heritrix allows for the extraction of links within a web site URL as well as JavaScripts and PDF files. Heritrix also normalises web sites to avoid repetition and overhead. The retrieved contents are stored in .ARC file format. The .ARC files are then extracted and analysed with ClamAV antivirus engines for patterns of known exploits and malwares. The binaries and JavaScript files are also stored for later analysis by malware analysis tools such as CWSandbox.

Similar to Monkey-Spider, HoneyC [26] implements a Yahoo search API to feed URLs to the visitor component which in turn retrieves and passes the content to be matched against Snort rules by the analysis engine. XML is used to exchange information between different modules in HoneyC. HoneyC achieves an average of 3.5 seconds for analysis of each URL. Most client honeypots discussed, perform content analysis and output the result to either a log file, terminal screen or a database in case of Monkey-Spider. They all perform similarly in terms of detection and speed, but their output representations are not easy or ideal for an end user to view. SpyBye [27] developed by Niels Provos on the other hand, takes a more effective approach and acts as a proxy server, allowing users to enter the desired URL into SpyBye where its content is matched against predefined patterns and ClamAV signatures. Utilisation of a proxy server component also allows a user to visit a web site using different browsers and browser versions, thus detecting exploits which target a specific browser vendor and version. The results are then displayed as a frame header for user to view before proceeding to view and browse the web site.

3.8.2 High Interaction Client Honeypots

A high interaction client honeypot is a full-fledged system running a real operating system, browser and extension to mimic a real user interacting with the malicious web site. On a HICH, no services or applications are emulated and real operating systems, browsers and browser plugins and extensions are installed. The system therefore requires a dedicated system or significant processing power and storage to host several virtual HICH systems running simultaneously. A HICH may require multiple versions of operating systems and various browser, plugin types and versions to provide the suitable environment to trigger a specific attack. Such a requirement adds significant complexity and requires significantly higher resources than a low interaction client honeypot.

High interaction client honeypots primarily rely on state based analysis engines to detect an attack. In a state based detection technique, a web site is classified as malicious if the process of retrieving and displaying the content of the web site results in changes to the underlying operating system, its files or running services. A suspicious web site is retrieved using HICH's real browser and embedded plugins. Parallel to interaction and retrieval of the web site content, a process monitors the system components for any changes including creation, deletion or modification of the system and browser variables. HICH monitors changes to Registry entries, operating system files and folders, browser settings, system processes and creation of any TCP/UDP ports. If a web site is classified as malicious, HICH reverts the system to a clean state prior to next retrieval.

An HICH introduces a specific delay set by user for every visiting URL. The delay ensures the detection of time-bomb attacks. A time-bomb attack is triggered and executed after a certain delay. A malicious web site may also require user input such as confirmation of a dialogue box or movement of mouse to execute. Time-bombs and user inputs are some of the antidetection mechanisms which are employed by malicious web sites to avoid detection by HICHs. Low interaction client honeypots on the other

hand are capable of detecting time-bombs and user-triggered exploits as they rely on signatures of an attack rather than the trigger process. A state based detection, having the right environment to trigger an attack can potentially detect zero-day attacks.

Finally, state based detection requires initiation of a virtual operating system in which a web site content is retrieved followed by a delay, monitoring host system and reverting back to a clean state if any changes are detected. This process may take between three seconds to few minutes depending on the URL content and processing power of the honeypot system. A low interaction client honeypot's speed in retrieval is only limited by the transmission medium's bandwidth while analyses is less than a second for a single URL. In terms of detection risk however, an HICH provides a real environment for the attacker to launch an attack, therefore the possibility of detecting the system as client honeypot is minimal.

SHELIA [28] is a high interaction client honeypot designed for attacks targeted at Windows operating systems and optimised to work with E-mail applications to perform URL and malware extraction to seed its visitor and analysis engine. SHELIA is considered a high interaction client honeypot for two main reasons:

1. It does not emulate the services of an operating system browser but rather uses the browser application installed on the user's machine/system.

2. Its detection engine does not solely rely on signature based detection but employs behavioural analysis and registry, file system and process monitors to detect an attack.

Analysis detection engine of SHELIA consists of two components; process monitor and attack detection engine. Process monitor utilises API hooking on certain Windows system APIs to monitor and log any changes to registry, file system and folders. Among these APIs are: NTSetValueKey, NTCreateFile, which allow for replacing or creation of a value entry for

a key in Windows registry and creating files and directories, respectively. Several other APIs include: CreateProcessA, CreateThread (Kernel32.dll file) and ShellExecute (Shell32.dll file). The attack detection engine works in conjunction with the process monitor and validates the origins of the each API call. If a call to a specified API, monitored by process monitor engine, originates from a non-executable region within the memory which is set by the Windows platform, an alarm is raised and the corresponding web site is flagged as malicious. SHELIA also allows for the browser application to run for a specific time, set by the user before killing the process. Setting an extended time enables the honeypot to detect exploits and malwares that might trigger based on a time bomb.

Sun, *et. al.* [29] designed an internet malware collecting system to find malicious contents that not only infect browsers but client side applications as well. The system is based on a customised web crawler to create a database of URLs to be visited by the honeypot. URLS and attachments from spam emails are extracted and inspected correspondingly. The honeypot itself is based on VMware running Internet Explorer, office applications, Adobe Acrobat and zip utilities. The client application component opens files and applications and simulates end user behaviour while browsing a URL. If any unusual activity is monitored, the URL is blacklisted.

Strider HoneyMonkey [30] developed by Microsoft uses virtual machines loaded with unpatched and patched Windows operating system to detect malicious web servers. By three stage detection, the first VM scans N number of web sites simultaneously. If a suspicious web site is detected, those N number of web sites are rechecked individually to detect the particular malicious one and at the third stage the detected malicious web sites are revisited by a fully patched system to find out the effectiveness of patches. In this system, registry modification, processes creation, creation of executable files and URL redirections are indicators of suspicious behaviour. Honeymonkey utilises crawlers as one of the mechanisms to

feed URLs to the system.

Capture-HPC [31] is a high interaction client honeypot consisting of a server and multiple clients based on VMware software. Server operates by instructing clients to start/stop and visit a specific URL while listening on port 7070 TCP/IP for results from each client. Every time a client visits a web site instructed by Capture-HPC server, it monitors registry entries, running processes and file system read and writes. If any changes occur when the site is visited, the client reports back the changes to the server and it classifies the server accordingly. The server then rolls back the clients to the default state before instructing it to visit the next URL in the list.

HoneySpider [32] is hybrid client honeypot consisting of low interaction and state-based high interaction client honeypots. It integrates multiple retrieval and detection tools and systems such as pdf analyser and JavaScript interpreter, Heritrix cralwer used in [25], Capture-HPC as high interaction client honeypot and Snort intrusion detection system [33]. The retrieval queries are also sent through a Squid web proxy server which scans the traffic with ClamAV antivirus. HoneySpider architecture allows the addition of modules to extend the capabilities of the system. These modules can be in the form of entire systems such as Thug low interaction client honeypot [34], Cuckoo sandbox and malware analysis [35] or online analysis systems (e.g. VirusTotal). Table 3.9 summarises open source and research client honeypots and their associated features.

Table 3.9: Summary of client honeypot systems

	Client Honeypot Systems					
	Type	**Analysis Engine**	**URL Collection**	**Visitor Agent**	**Geolocation**	**License**
Phoneyc	Low Interaction	ClamAV, Vulnerability Driven, De-obfuscation and Dynamic analysis of JavaScripts Shellcode Detection	Not Available	Curl Library	No	Free
Honeyware	Low Interaction	5 Antivirus Engines	Yahoo, MSN	Unknown	Very Limited	Not available
Honeyc	Low Interaction	Snort	Yahoo, Extraction from text files	HTTP Request Library	No	
Thug	Low Interaction	YARA De-obfuscation and Dynamic analysis of JavaScripts, Vulnerability Driven Shellcode Detection	Not Available	HTTP Request Library	No	Free
Monkey-Spider	Low Interaction	ClamAV	Google, Yahoo, MSN SPAM, Blacklisted Domains	Heritrix Crawler	No	Free
SpyBye	Low Interaction	ClamAV Custom Pattern matching	Not Available	None, Proxy based	No	Free
UW SpyCrawler	Low Interaction	Adware	Not Available	Internet Explorer	No	
Shelia	High Interaction	Custom	Email	Multiple Browsers	No	Free
Sun et. al.	High Interaction	State based	Web Crawler, Spam	Internet Explorer	No	Not Available
HoneyMonkey	High Interaction	State based	Not Available	Internet Explorer	No	Not Available
Capture-HPC	High Interaction	State based	Not Available	Multiple Browsers	No	Free
HoneySpider	High Interaction	State based, ClamAV, Snort	Spam, Proxy logs, MSN, Yahoo and Google results, Contracted URLs	Heritrix Crawler, Multiple Browsers	No	Free

Client honeypots may rely on various analysis engines to detect malicious behaviour of a retrieved web site. Low interaction client honeypots generally depend on signature, heuristics and pattern matching techniques while high interaction honeyclients identify state changes to detect malicious web sites. Most of these techniques are however independent of the type and interaction level of client honeypots and can be implemented

in low and high interaction honeyclients. State based detection engines are however exclusively implemented in high interaction client honeypots such as Capture-HPC and HoneySpider.

3.9 Static Analysis of Dynamic Content

Static analysis of content can generally be performed on scripts which are not obfuscated. It looks for common attributes associated with malicious script behaviour such as multiple redirects and zero-sized iframes.

ADSandbox's [36] static analysis of dynamic content is achieved through identifying iframes and redirect policies. It determines whether they follow the JavaScript rule for the same origin policy. It also examines the properties of iframes to determine key properties that resemble malicious behaviour such as height and width of zero, or out of area position attributes. A combination of hidden properties and access to foreign domains is a good indication of malicious behaviour.

Cujo's [37] implementation of static analysis is based on lexical analysis of the code, transforming the code into simpler lexical tokens in the form of tokens, keywords, identifiers (ID) and literals, including numeric and string. New keywords corresponding to functions to be searched are then added to the static analysis engine. The analysis system then looks for the keywords that are common functions of drive-by downloads such as string operations and/or calling the "eval" function.

3.10 Dynamic Analysis of Dynamic Content

Scripting languages such as JavaScript and VB script allow web designers to hide components of web sites through different techniques (for example: obfuscation), which also allow attackers to hide and thwart static detection approaches. A solution to overcoming the weaknesses/shortcomings of static analysis is to classify the script based on its behaviour at

runtime rather than heuristics. Detection based on signatures and heuristics however might (mis)identify plenty of legitimate web sites and generate false alarms. Techniques used in dynamic analysis of such content consist of detaching the dynamic script from the static HTML file and executing the script in a simulated environment or sandboxes, where different behavioural aspects of the script are monitored and analysed.

ADSandbox performs the extraction of the JavaScript portion of a particular HTML site and passes the script to be run in an environment using a modified SpiderMonkey JavaScript engine. In this environment every access to JavaScript objects is logged through the static "callback" function of "JavaScriptExecution" class. The log of the JavaScript execution is then processed and analysed through the Perl Compatible Regular Expression (PCRE) pattern matching library for patterns of malicious behaviour in file save functions and execution of the saved file.

JSand, one of the most powerful dynamic analysis engines, employs anomaly techniques incorporated into a web browser that runs Mozilla's Rhino JavaScript interpreter. Its detection engine is based on features or conditions that have to be met for a successful exploitation. The engine looks for attributes that define a malicious behaviour while de-obfuscating the dynamic content. JSand categorises some of these features as vital to a successful exploit, while others help pave the way. Signs of preparing the environment for an attack such as allocation of large memory space for string operations, and embedding static or dynamic shell code content in the form of long, non-printable uni-coded strings are strong indications of malicious behaviour. Other vital factors relate to the final steps in the process for exploiting the vulnerability. These factors include instantiation of a large number of browser components and plug-ins to maximise the rate of a successful exploit, tracking large string values passed between methods or set as values of a property, and sequencing of actions initiated by a script (for example: downloading a file and running a local executable file). Optional features that define an anomalous behaviour

include high numbers of redirections, high string definition-to-string use ratio, high numbers of Java interpreter calls (for example: eval) and Document Object Model (DOM) changes and, finally, the large size of string codes passed to the "eval" function to interpret; all good indications of malicious behaviour, especially when occurring in combination [38]. Systems such as JSand and ADSandbox are designed to perform detection and analysis offline. Others however have undertaken a more direct approach to provide protection to end users in real time.

A behaviour-based analysis of dynamic content is implemented in SpyProxy which can be run as a client, or as the name implies at the proxy level. It combines a simple static analysis of HTML; if the page only consists of simple HTML code, it passes it directly to the user browser to be rendered and viewed. Pages containing any dynamic content are parsed to a virtual machine where they are rendered and executed in a browser. The system then looks for trigger events that do not resemble typical browser behaviour, such as creating new processes, attempting to access non default folders and registry modification. If any such behaviour is observed, the proxy prevents the transfer of the contents to the user's browser. However, the system faces challenges in terms of detection of attacks that target users based on non-deterministic factors such as system properties or time. To overcome this problem, the system would need to track and mimic every user system's properties, browser type, time and so on, which in reality is nearly impossible. Other problems with the system include its ineffectiveness and reliability issues concerning denial of service where a page is not delivered to a user. Web sites might also require user input or be based on a timer activated in a dynamic content based attack [39].

Hallaraker and Vigna [40] incorporate a combination of static and behavioural based analysis to detect primarily cross-site scripting attacks. The detection engine relies on logging script behaviours and performing checks that match the code behaviour against signatures of attacks recorded in an intrusion detection database. These checks include moni-

toring the number of times a certain method is called and executed, keeping track of open windows and verifying the origin of scripts to detect cross-site scripting attacks. While these systems rely on monitoring the behaviour of the script upon execution, other research has focused on designing policies to regulate dynamic code execution, to monitor their behaviour or minimise the risk of an attack.

BrowserShield [41] follows a policy driven interposition approach for rewriting codes of web sites/HTML+Java into safer equivalents. This is achieved by applying logics to perform recursive run time checks on dynamic content against known exploits at the client or firewall level, before they are rendered by the web browser. These policies can be in the form of identifying properties of iframes and frames, tags and names to determine buffer over-run exploits or based on known vulnerability signatures. The system is flexible in terms of allowing users to create custom policies which can be applied at the firewall level to all web connections, providing centralised control and minimising the overhead of updating each client's policies. Internet explorer also relies on a security policy model where an instance of Internet Explorer run in protected mode, is assigned a low integrity level that only allows it to write or modify low integrity folders such as a temporary internet files folder. A low integrity level process is denied any other alteration to the user and operating system's folder data, which by default are set at a medium integrity level [42, 43]. Table 3.10 summarises the key properties and functions of popular dynamic content analysis engines.

System	Type of Detection	Analysis Engine Properties and Functions
ADSandbox	Static and Dynamic Analysis	⌡ Monitors iframes, Redirects, Same Origin Policy rules and Window's Properties ⌡ Extracts JavaScript ⌡ Runs the JavaScript in a controlled environment ⌡ Monitors and logs access to JavaScript objects ⌡ Employs pattern matching to detect malicious behavior within log file
Cujo	Static and Dynamic Analysis	⌡ Lexical analysis by converting scripts into tokens ⌡ Searches for common drive-by functions within tokens
JSand	Dynamic and Anomaly Analysis	⌡ De-obfuscates and runs JavaScript in Rhino interpreter. ⌡ Monitors conditions required for successful exploit. ⌡ Monitors environment preparation behaviors (for example: Allocation of large memory space). ⌡ Monitors final exploit behaviors (for example: Instantiation of large number of browser components). ⌡ Monitors optional exploit features (for example: High number of redirections, high string definition-to-string use ratio).
SpyProxy	Static and Dynamic Analysis	⌡ Simple HTML pages are passed directly to browser ⌡ Dynamic scripts are passed to a virtual machine and executed ⌡ Monitors suspicious behaviors (for example: Creating new processes, attempting to access non default folders and registry modification)
Auditing System for Mozilla	Static and Behavior Analysis	⌡ Focuses primarily on detection of Cross-Site Scripting attacks ⌡ Logs and matches behavior against intrusion detection system signatures ⌡ Monitoring behaviours such as: number of times a certain method is called and executed, tracks of open windows and verifying the origins of scripts.
BrowserShield	Dynamic and Policy driven Detection	⌡ Rewrites codes into safer equivalents ⌡ Performs recursive run time checks on dynamic content against known exploits signatures. ⌡ Policies in the form of identifying properties of iframes and Frames, Tags and names. ⌡ Allows creation of custom policies

Table 3.10: Key properties and functions of popular analysis systems

3.11 Discussion and Summary

In this chapter we discussed various techniques which could be used to estimate the location of a visiting user. Several geolocation techniques require access to multiple landmarks and require extended time to achieve accurate detection. Passive geolocation techniques on the other hand pro-

vide rapid detection at the expense of lower accuracy which makes them particularly suitable for web based applications and large scale deployment of malicious Web sites in which user geolocation is not a critical factor.Current research has mainly focused on improving the detection rate through improving the analysis of client honeypots by integrating various detection engines such as state based, anomaly based and behavioural analysis techniques. The current research fails to recognise that geolocation attributes discussed in this chapter can be used by a malicious web site to perform cloaking and avoid detection. Current client honeypots also fail to integrate the capability to allow a user to simulate the discussed geolocation attributes and mimic retrieval from a location different from the location in which the client honeypot resides in. Inability to simulate retrieval from multiple locations results in geolocation cloaking by a malicious web site in which benign content is delivered to a detection engine which does not reside in the geographical location for which the attack is intended. In such a scenario the attack is missed regardless of the detection engine of the client honeypot. Current client honeypot system are also: 1) Too complicated and resource intensive to be deployed in large scale 2) They have no means by which to reliably and efficiently retrieve URLs from multiple deployment locations using multiple instances or retriever agents. Currently there are no client honeypot systems capable of detecting geolocation and targeted attacks. Existing research's solution to geolocation attack has been to suggest retrieval through the Tor relay network [44, 45]. Malicious web sites however have been observed to block Tor exit nodes to avoid detection [46].

Finally there are few to no measurement studies to highlight the impact and ever growing threat of geolocation attacks and cloaking. Spam studies have determined the impact of referer attribute on the number of delivered spam. Referer attribute can similarly be used to mitigate detection through referer cloaking in malicious web sites. Measurement studies however need to identify the prevalence of geolocation cloaking in mali-

cious web sites using various other geolocation attributes of a client honeypot such as HTTP header information and IP addresses. Such studies will help researchers determine the impact of each attribute on the detection and discover methods to detect or control them in a study.

PART II – How Did We Investigate It?

Chapter 4

HAZard and OPerability

HAZard and OPerability (HAZOP) is a structured and systematic approach involving thorough analysis of a system, its processes and components by experts to identify if any system or environment attributes in which the system operates, may result in deviation from the design intent [117]. HAZOP provides a methodology for scientific experimental methods to design reliable and replicable experiments by controlling all variables and hypothetical scenarios in an experimental setup, allowing the validation of experiment's internal validity by confirming the relationship between dependent and independent variables. The aim of a HAZOP study is to identify and mitigate risk and hazards into the operability of the system functions. Originally found in chemical engineering, we are applying it to the design of network security experiments. In our context, risks and hazards in a quantitative computer science experiment could translate into increased risk of an unexpected and biased variable which would impact the internal and external validity of the experiment. High internal validity is desirable because it indicates that confounds coming from the experiment itself have been considered and dealt with in order to ensure we are measuring what we believe we are measuring. Confounds in this context are variables that cause hazards to the reliability of validity. External validity, on the other hand, is the degree of certainty in which it could be

argued that the conclusion of the study will hold true and can be used to make predictions about a larger population. High external validity in an experiment warrants a high degree of certainty about the generalisability of the findings. For example, that the findings about the particular web sites studies will generalise to web sites of a similar type but located in different locations.

4.1 Hazard and Operability Analysis

We have adopted HAZOP as a methodology for reviewing the design of our cyber security experiments. This meets the needs identified above because it is neither too prescriptive nor specific. Based upon the reviews of the literature a HAZOP study involves the following steps:

1. We define the hypothesis which needs to be tested. This is essentially the objective of the experiment. The aim is to validate that this hypothesis has been tested appropriately.

2. The intention of the study needs to be defined. The intention describes how the experiment is meant to be conducted. The notion is that there is an action that we wish to observe and measure for our experiment that will create events that can be analysed. Observation and measurement may be thought of as a function that is parameterised by the apparatus experimental protocol, subjects, stimuli and environment. The act of observation results in an observable event. This event will include error that has been introduced by the observation process itself. A set of events are then analysed and we conceive this as a function that is parameterised by data, type of test and sample size. The high level view of the design must be further analysed. The apparatus is usually a device or system which can be further divided into components and explained using step by step description or UML diagrams such as block, sequence or use

cases [66, 67]. Other parameters are the subjects, the stimuli and the environment. Each of these has to be described in detail and their selection justified.

3. Deviations are defined by applying the guidewords to the lower-level view of the experiment. It may not be feasible to apply all the guidewords to every component or process. The goal is to consider all feasible deviations.

4. The possible impact of the deviations upon the ability to disprove or prove our hypothesis is defined. This helps us make an assessment whether actions need to be taken to minimise or mitigate the deviation.

5. Where an action is required, steps are taken to rule out the hazards or threats to validity. If it is not possible or practical to completely eliminate a hazard, then considering the severity and likelihood of the assumed threat, actions should be taken to minimise its impact [118].

Features of the HAZOP approach mentioned in this chapter may vary compared to the original HAZOP model used in chemical engineering and have been adapted to suit the requirement of the study. This practice is accepted by many other studies applying HAZOP to other fields [69, 66, 70]. For instance, having a team leader and a group of specialists from other fields was not deemed necessary for this research and was not followed. Our methodology follows a similar apprach in analysis to the research performed by Seifert *et. al.* [65]. HAZOP methodology in this thesis is mostly focused on system and experimental design analysis and risk and hazard mitigation. The main components of our HAZOP study include:

- Study Node: Specific points in the design where the deviations are studied.

- Guide word: Short words suggesting deviations from the correct state of the system . The standard keywords of HAZOP include o, More, Less, As Well As, Part Of, Reverse, and Other Than. Other keywords such as Early, Late, Before and After are considered to be derivatives of ”Other Than” keyword to apply more specific meanings to a deviation [119]. The application of the guidewords results in hypothetical deviations related to that component or process (Table 4.1). Some guide words should be interpreted broadly as different form of these words may be more appropriate to the process or artifact to which they are being applied. Guide words used in our research were:

Table 4.1: Guide Words and their interpretation used in our study

Guidewords	**Meaning**
NO	No part of the design intention is achieved.
MORE	Quantitative increase in a parameter.
LESS	Quantitative decrease in a parameter.
OTHER THAN	Compared to the design, something completely different happened.
EARLY, LATE	The timing is different from the intention.
BIASED	Systematic error across a set of measurements usually introduced due to way the sampling done or the measurement process itself.
UNRELIABLE	A value or measurement is not the same when measured again in the same context as it was measured before.

- Deviation: Combined with guide words, they define a more descriptive departure from the design intent of the component or process.
- Cause: Describes the potential cause which would result in the deviation.
- Severity (Denoted by S, 1=Low, 2=Medium, 3=High): Expresses the potential hazard and threat if the deviation occurs. If it is not possible

to entirely eliminate a hazard, then it should be minimised in respect to its severity and likelihood [118].

- Likelihood (Denoted by L, 1=Low, 2=Medium, 3=High): Expresses the possibility of the deviation happening within the study node. The focus of our HAZOP study is to identify experimental scenarios with high and medium likelihood of occurrence and proposes measures to mitigate or significantly reduce their effects. The "cutoff" points represent the line below which the threats to validity of final data and result are so low and can be neglected [118].

- Consequence: Describes the potential results if the deviation occurs.

- Required Action: Identifies any measures to be taken to either remove the cause or eliminate the consequence.

Applying a HAZOP methodology to our repeated measurement experiments, we question every component of the experiment's design including client honeypot, malicious web sites, processes and protocols used in the interaction. These lower-level design components and their associated hazards will each be discussed in detail in the following sections followed by HAZOP tables. Figure 4.1 shows the relationship between risk, severity, likelihood and required actions in a HAZOP study. In our study of HAZOP we attempt to mitigate and minimise all hazards which may introduce medium or high risk into the study.

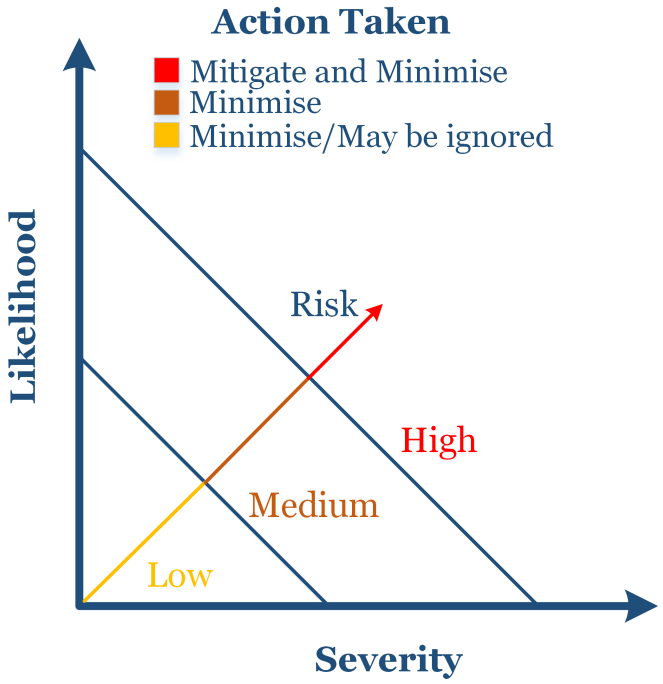


Figure 4.1: relationship between risk, severity, likelihood and required actions in a HAZOP study

HAZOP methodology relpaces the traditional safety methodologies such as fault Trees, Check lists and risk rankings. It is however very similar in concept to other approaches such as Failure Mode and Effects Analysis (FMEA) studies. The underlying concept is identification of components and processes which may cause failure (hazards) and determining actions to be taken to reduce or mitigate those failures entirely. FMEA relies on failure modes (complete failure, partial failure, intermittent failure, function out of specification and unintended function) to categorise each failure and similar to HAZOP, utilises severity and Occurance Rating (Likelihood) to measure the probability of a failure. FMEA differs from HAZOP by assigning a Detection Rating which indicates the probability of detecting a failure arising from a particular cause such that the effect of failure is prevented [120]. The Layer of Protection Analysis (LOPA) method iden-

tifies the level of risk associated with various hazards through severity and likelihood (Occurance rating) data gathered from HAZOP or FMEA studies.

We selected HAZOP over other approaches as it provides more in-depth analysis of the risks, causes and required actions associated with each risk. HAZOP also provides ranking for each risk through severity and likelihood measurements. Traditional safety methodologies such as check lists and fault trees lack such a level of detailed information about risks, their likelihood and effect. HAZOP also allows identifying risks in the forms of bias using guide words such as BIASED and UNRELIABLE. In our case, these allow us to measure risks and incorrect analysis associated with bias introduced through input URL datasets. FMEA and other methodologies lack such functionality. Bias through input URL dataset is thoroughly covered in section 4.2.1.

4.1.1 Background and Previous Work

Computer science experiments are used for theory falsification [47], investigation of assumptions [48] and developing theory based upon observations [49]. Some authors however argue that a high percentage of research in the field of computer science are not supported by a scientific experimental methods for theory validation and the design of many experiments do not demonstrate reliability of experimental design or internal validity [50, 51, 52, 53, 54]. The basic problems have not changed since they were identified at a panel discussion "Why is there no science in cyber science?" in 2010 at the New Security Paradigms Workshop [55], namely: (1) lack of testable or falsifiable hypothesis; (2) experiment is not adequately controlled or reproducible; (3) there was little if any analytical analysis; and (4) the hypothesis does not relate to a system model or causal relationship that could be used to advance the field.

We seek to address problem #2 identified above, i.e. many experiments

are not adequately controlled or reproducible. In particular, the lack of control in design of an experimental method leads to a lack of reliability or internal validity [50, 51, 52, 53, 54]. Reliability in an experimental method determines if consistent results can be obtained by repeating the experiment while internal validity concerns the extent to which a relationship between a dependent and independent variable actually exists and is not obscured by confounding factors [56, 57]. Experiments may produce inaccurate conclusion because the designers have not properly controlled design and experiment factors which may affect the final result. For instance, our own cyber security research investigates the effect that the metadata associated with a visit by a browser has on the likelihood that a malicious web site will serve a visitor either a page containing malware or a benign page. We are interested in this phenomenon because cybercriminals running these malicious web sites wish to hide themselves from security and web search companies, in their attempts to stay under the radar.

For instance, the experiment in chapter section 6.2 focuses on the relationship between maliciousness of a web site (Parameter X) and the pattern of visit by monitoring a visitor's IP address (Parameter Y). An experiment using nodes in different geographical locations (Parameter Z) could neglect the effect of geolocation in malicious web site attacks. It could lead to erroneously concluding that there is a relationship between X and Y when there is a stronger relationship between X and Z because of geographical factors such as language and social factors which contribute to the number and malicious behaviour of a web site. Experimenters could also fail to describe how they controlled for the effect of Z.

Tichy [50] is widely referenced as a key paper on experimentation by software engineering researchers and in cyber security experimental design (e.g. [58]). This chapter provides a general introduction to the concept of the traditional scientific method and argues that it is applicable to computer science. Carroll *et. al.* [59] identified four mandatory challenges when attempting to improve the state of cyber security experimentations,

namely:

- Generating a hypothesis that can be falsified is difficult because most cyber security experiments seek to show the absence of an event (e.g. an attack) rather the occurrence of an event. Attackers by their nature will attempt to hide evidence of their activity so we might assume the hypothesis is shown correct although we may have missed an actual attack (the need for forensic investigations is, therefore, essential [60]. The experimental design must explicitly consider this possibility and the validity of any underlying assumptions.

- Reliably reproducing cyber security experiments is more difficult than in physical experiments because of the level of detail necessary and the sensitivity of experimental outcomes to even small differences in configuration of hardware or software.

- Cyber security experiments require control of large numbers of independent variables using not only methods from physical sciences but also methods of control from social sciences, such as assigning control and experimental groups.

- Confounding factors must be identified and managed as part of the experimental design. These factors will not only be related to physical or logical flaws in the experiment but also human bias on the part of both the experimenters and the subjects.

The relatively recent development of cyber security testbeds goes towards addressing some of these challenges. In particular, DETER [61, 62] addresses the issue of reproducing experiments by implementing automated generation of networks with large numbers of hosts as specified by an experimental configuration provided by the experimenter. The Automated Experimentation System (AES) [58] extends the DETER approach by allowing the experimental process to also be described and automatically

carried out by the system. This improves reliability and ensures that experimentation can be scalable (challenge #2).

What these testbeds do not address are identifying a falsifiable hypothesis (challenge #1), and identification and mitigation of confounding factors (challenge #4). Carroll *et. al.* [59] suggest that these challenges might be addressed through completion of their survey and theory development as well the development and validation of an experimental checklist aimed at the design, conduct and evaluation of rigorous cyber security experiments.

Maxion *et. al.* [63, 64] have taken a more specific approach. Rather than trying to develop a general approach, they investigate how to design experiments that identify potential confounds and appropriately control or mitigate them. In their case study approach, they use their experiments on keystroke biometrics for user identification to illustrate the difficulties in implementing reliable measurement in the presence of confounds. Our approach is inspired by the need for some form of guidance to designers of cyber security experiments, as identified by Carroll *et. al.* and the work done by Maxion *et. al.* on reliable measurement. We believe that developing a checklist, as proposed by Carroll *et. al.*, might be too prescriptive especially given the need to consider an active adversary as part of the design of the experiment. However, we would like to provide more general guidance than is found in the work by Maxion *et. al.*

Seifert *et. al.* [65] performed a hazard and operability study and identified hazrads and bias which could be introduced into experimental studies using state-based high interaction client honeypot. Their study using experiments on real-world datasets demonstrated that subjects of studies - the input dataset and its source of collection - can have significant effect on a measurement study. Increased number of malicious web sites in datasets gathered from adult web sites and top level domains were observed. Their repeated measurement study on the number of malicious web sites in the .nz domain over an eight months period, revealed the impact of retrieval

time on the outcome of a measurement study as significant increase and decrease in the number of detected malicious web sites were observed.

Application of HAZOP to computer science field is most notable in the field of Software Engineering and particularly Security Requirement Analysis. Srivatanakul, Lano and Daruwala [66, 67, 68] apply HAZOP to UML and use-cases in the process of software and system development to identify security threats and vulnerabilities and define security requirements. Winther *et. al.*[69] argue that applying HAZOP to security assessment of safety critical systems is effective since safety and security equally ought to achieve similar objective. Similarly HAZOP was considered for safety critical applications and applied to an air traffic prototype application to improve usability and prevent critical errors [70].

The client honeypot throughout this research is a low interaction client honeypot called YALIH [121]. Yet Another Low Interaction Honeyclient (YALIH) is based on a simulated browser responsible for retrieval and multiple analysis engines for detection and classification of web sites. In the following sections, we provide a HAZOP analysis of a low interaction client honeypot. We identify the components, processes and any factors which could potentially introduce hazards and bias into an experimental and measurement study. We also identify any controls which could be placed to mitigate and minimise the identified threats and bias. Identified hazards through HAZOP study in this chapter are then mitigated and minimised by implementing controls in the design of YALIH which is discussed chapter 5.

4.2 Application of HAZOP to the Design of a Low Interaction Client Honeypot

Low interaction client honeypots may vary in their design, detection engines and capabilities. They are however designed based on the three

main components:

- The Queuer responsible for collection of suspicious URLs and feeding the URLs to the virtual browser.
- The Visitor: Responsible for interaction with the potentially malicious web server and content retrieval
- Analysis Engine: A combination of detection engines and techniques designed to identify and classify a web site as either malicious or benign.

4.2.1 Queuer

Potential bias introduced into the study through Queuer would ultimately affects the external validity of the experiment's results as it is responsible for selecting the input URLs. Depending on the number and nature of selected set of URLs, conclusion and external validity of the results would vary significantly. The bias caused by the queuer component could be through:

- ***Number of selected URLs to visit*:** Number of selected input URLs should be sufficient to meet the required external validity of an experiment. Depending on the purpose of the study and its external validity, the number of input URLs could vary significantly. The intent of an experiment might be to understand the impact of IP tracking on malware delivery within malicious domains and identify the patterns. Such an experiment would require a significantly smaller dataset than an experiment to generalise IP tracking behaviour across the Internet.
- ***How the URLs are selected*:** Similarly, purpose of the study and its external validity would determine the source of the input URLs. A study of malicious behaviour in a specific top level domain requires

only URL inputs from the studied domain. A high external validity (for the result to be generalised for the entire Internet) would require a large selection of random URLs consisting of all top level domains and categories (news, entertainment, warez, torrent, etc.)

- ***How the URLs are fed to the visitor*:** Crawling entire links within the main domain could be detected and interpreted as a crawling behaviour by the malicious web site. This could ultimately result in serving benign contents and subsequent false negative upon analysis. Alternatively, avoiding the crawling of all hyperlinks within a domain may result in false negative through oversight of potentially malicious embedded links. A balanced approach would be to only retrieve the links which point to executable files, PDF and document files. Initializing a high number of connections in a short period from a single IP address could be an indication of crawling activity and subsequent non-malicious content delivery and false negative. To minimise such a hazard, an experiment may insert a specific delay between visiting an extracted link from a URL.

4.2.2 Visitor

The attack surface is a factor that threatens a client honeypot's ability to detect malicious web sites. The mechanism by which a client honeypot detects an attack is by exposing vulnerabilities which match the attacks. A malicious web site detects the operating system, the browser and the browser plug-ins used by a visitor and serves an exploit or malware which targets that specific operating system, browser agent or plugin. The operating environment from which a client is requesting a we site content therefore directly influences the number of received and triggered attacks. Subsequently, the lack of exposed vulnerabilities by a visitor through operating system, browser and browser plugin may therefore introduces a threat into a study. A malicious web site may return benign content to

the visitor if the right environment is not provided and the expected vulnerabilities are not exposed. Systems such as Rozzle attempt to overcome the difficulty of detecting environment-specific malware through a multi-execution approach to explore multiple execution paths within a single execution [123]. Malicious web site may also detect the presence of a client honeypot by assessing the visitor agent befor launching an attack.

Deployment of an outdated browser with a large number of vulnerabilities may expose the client honeypot to a high number of attacks and result in increased detection. Using an outdated user-agent to represent a legacy browser however may not always be the best approach. The Magnitude exploit kit for instance bans malware delivery to users with Internet Explorer 6 simply because of widespread use of this user-agent by security researchers.

A low interaction client honeypot relies on simulated browser, operating system, plugins and extension; therefore, it does not provide the functionalities of a full-fledged operating system or browser. Monitoring the outcome of a JavaScript execution before redirecting users to the malicious code for instance, would fail to trigger an attack, as the simulated browser would fail to render the code properly to generate the desired output. Therefore, inability of a client honeypot's browser to match the required environment of an exploit to be triggered may result in failure of exploit execution and subsequent false negative.

4.2.3 Analysis Engine

Analysis engine is the component responsible for assessment and classification of web sites. An analysis engine of a low interaction client honeypot relying on signature-based detection could potentially introduce threat into a study by missing the following attacks, resulting in higher false negative rate.

- Encrypted attack

- Attacks for which no signatures have been developed
- Variations of attacks for which signatures have been developed

Mitigation strategies to minimise these hazards in a low interaction client honeypot would include:

- Encrypted attack - Decrypting the embedded encrypted attack using proper rendering engine (e.g. Rhino, V8) or observing the behaviour of the script during execution
- Attacks for which no signatures have been developed - Using Heuristic and pattern matching techniques to detect the environmental variables used in an attack (e.g. zero-sized iframe) or observing the behaviour of the script during execution
- Variations of attacks for which signatures have been developed - Create broader signatures to include the family of the attack and all its variations

A general approach to increase detection rate in a low interaction client honeypot would be to use a combination of analysis techniques to maximise the detection rate.

4.3 Subjects of the Study (Web Pages)

The primary threat introduced by the subject of the study is connectivity; in which a web page may not be able to participate in the study because network components, such as DNS server and HTTP server, are temporarily unreachable. These services are fundamental components of the design and their failure poses an enormous bias into the study. Potential malicious but unreachable web sites due to DNS operation failure or DNS server unreachability in a point of time will result in false negatives which

are not necessarily similar across all retrieving nodes. A mitigation strategy would be utilizing multiple high reliability DNS servers simultaneously and performing retrieval in near parallel timing. Unavailability of the subject upon initiation of retrieval process by the visitor is another hazard to be considered. Content unavailability can be mitigated through attempting to retrieve the web site content multiple times at various stages and logging unsuccessful attempts.

4.4 Retrieval Process and Intermediary Devices

Web site content retrieval and malicious detection involve protocols and services which are beyond the boundaries of the client system. They are, however, a part of the larger design and can pose a hazard into the operability of the system. A malicious web page may choose not to participate in the study by analyzing the way requests are made and mitigate malware delivery based on variables that make up that request. Several characteristics of the requests may cause this hazard, such as location, time, deceptive nature, and history.

4.4.1 Location

The location from where requests are made may pose a hazard to a measurement study. Malicious web sites lure users based on their location and delivering attacks customised for a specific location or region (e.g. language, cultural, and popular trends associated with that particular location). Popularity of online gaming for instance has resulted in malware targeting users from Asia for gaming credentials. Ransomware is a malware that adjusts itself and language based on the location of the infected user. Adapting attacks based on social and cultural trends are a part of social engineering attacks which have proven to be more successful than generic attacks [124]. Location-based attacks also help malicious web sites

evade detection by not exposing their attack to users located outside of the particular location and, therefore, maximizing their life. Features to deploy malicious web sites that operate based on location of a visiting user have been observed in large number of BEPs, an indication of their popularity [125, 126]. The hazard introduced by location-based malware could result in false negative if the web server determines an IP tracking node resides outside of a designated region and serves non-malicious content. A mitigation strategy to minimise such threat would be to determine the likely audience of a web site through Top Level Domain (TLD), WHOIS, DNS or Autonomous System (AS) number information.

4.4.2 Language

Similar to location, language variable set by the browser and particularly HTTP request header fields can influence the type and behaviour of the malware delivery. As languages can be associated with the location of a user through its IP address, the contrary may apply with less accuracy. A ”ZH-CN” Accept-Language header value for instance defines a user whose locale is set to Chinese, and located in mainland China. Spanish users located in Spain may also have their locale set to ”ES-ES” or ”ES-CL” for Chile. A combination of user-agent’s ”Accepted Language” string with IP address geolocation can determine a user’s region and preferred language setting. These settings can then be used to target the user with specific malware (e.g. Ransomware). In order to detect such attacks, the likely audience of a malicious domain may be predicted through a single retrieval from a different node, identifying the language and changing the IP tracking node’s setting accordingly. Multiple retrievals from various regions can also significantly reduce such a threat, albeit with a higher cost of retrieval and detection.

4.4.3 Operating System

Operating system of the visiting agent (if monitored) can influence the type of malware delivered. Malicious web sites may detect the operating system through browser's user-agent information or running JavaScript code within the browser and deliver specific malware. Such capabilities are widely present in BEPs and can introduce hazard if the right operating system is not present for the web server or the malware to trigger [126]. A client honeypot may fail to detect an attack if its user-agent value does not match the agent for which the attack has been designed. In this case, the malicious web site may redirect the client honeypot to a benign web site.

4.4.4 Intermediary Devices

Intermediary devices, such as firewalls and proxies, can individually change the operating environment's information and subsequently affect the result of an experiment. These devices may be setup to remove the HTTP header information, user-agent and referer information from packets. Intermediary devices may also cause the retrieval from multiple retrieving agents to be routed through a single IP address, resulting in increased risk of introducing hazards through IP tracking.

4.4.5 Time

The time of web site retrieval could introduce hazard into the study as the behaviour of web sites could change based on traffic pattern that are time-dependent. Malicious web sites could potentially target users during off hours to minimise detection or vice versa to maximise their rate of infection. The malicious web site may go offline or its behaviour change during the time it takes for two detection nodes to retrieve and perform analysis. Timing hazard based on traffic patterns may be addressed by aligning the client honeypot to request patterns. Timing hazards with IP-tracking de-

tection systems could be minimised by retrieving contents from multiple nodes within a short period from one another. This will not raise suspicion and minimise the risk of malicious web site behavioural change or takedowns during the visit.

4.4.6 History

The history of requests may pose another hazard to a study of malicious web pages. Particular malicious web pages implement a tracking functionality in which the attack is launched only once upon a target through monitoring IP addresses. A client honeypot requesting the identical page a second time would not lead to an attack and, therefore, the malicious web page would be missed. There are several mitigation strategies against this hazard. One could simply choose a visitation algorithm that does not require repeated interaction (such as the sequential algorithm). Tracking the history of visits can also be implemented through use of cookies by malicious web sites. Utilizing a browser capable of handling cookies and removal upon retrieval can minimise such threats.

4.4.7 Network and Organisational Profiles

Similar to public IP addresses, information on IP address ranges are available online which disclose information about the IP range owner and their association with organisations. A malicious web site could potentially block malware delivery to all IP addresses belonging to a certain subnet, suspicious of hosting client honeypots or associated with security organisations. Network banning feature has also been observed in multiple BEPs [126].

4.4.8 HTTP Header Values

HTTP protocol's request header contains information about the client's preferred language, operating system, the requesting application and character set which may be used in a targeted or cloaked attack. Some of these are Accept-Language, Accept-Charset, From, Via, User-Agent, X-Forwarded-For and Referer.

Referer is an HTTP request header that identifies the location a client followed to reach the resources on the web server. Referer validation has been widely used in spam web sites to increase ranking within the search engine results, mitigate detection by search engine crawlers, and serve users with add filled web sites. Similar to IP-tracking, malicious web sites can utilise referer header as a mean of evading detection by associating particular referers with average users and delivering malware to users who retrieve a web sites through specific gate-pages. Referer cloaking feature has also been observed in a number of BEPs.

4.5 Summary

In this chapter we discussed HAZOP methodology and how it can be used in the design of a system and its processes to minimise any risks identified by hazards and biases, which could potentially deviate the system from it's original intent. HAZOP methodology was selected for our study as it provides a systematic and step-by-step approach to identify and outline the possible deviations which could occur as the result of the large number of system or environmental components and factors affecting our experiments. Experimental designs to detect malicious behaviour of web sites include a large set of potential confounds associated with various components and parties involved. Some of these attributes are associated with detection system components such as analysis engine - due to lack of available attack signatures; environmental variables rising from network states

or system module malfunctions or possibility of bias due to the selection of the subject of the study (i.e. set of retrieved web sites).

In the context of our study, HAZOP is applied to the design of our low interaction client honeypot and the experimental setups of chapter 6. We identified potential risks and deviations in various components of a client honeypot such as the queuer, the visitor and the analysis engine. We also established that subjects of a study - in our case, the input URLs - can have significant effect on the outcome of the study and should therefore be carefully selected to deliver high internal or external validity based on the intent of the study.

Through HAZOP we also analyse components, processes and experimental environment in which our client honeypot is running; and determine the likelihood and severity of the threats which might be introduced by each. Subsequently the consequence of each threat on the outcome of the study is identified and measures are taken to mitigate those threats or minimise their impact on the outcome of the experiment.

Chapter 5

YALIH, Yet Another Low Interaction Honeyclient

In this chapter we discuss the design and development of the low interaction client honeypot used throughout our experiments and analysis. The client honeypot follows the HAZOP methodology discussed in the previous chapter and attempts to identify any hazards or bias in its design which could potentially undermine its detection and analysis capabilities and unintentionally affect the result of our analysis. We designed and implemented YALIH (Yet Another Low Interaction Honeyclient) to facilitate retrieval and analysis of malicious web sites in our experiments. Prior to designing our own client honeypot, several low interaction client honeypots and a high interaction client honeypot were tested and their suitability for our experiments assessed. The available systems however lacked the requirements of our HAZOP study and our own in terms of simulation and detection capabilities. There were several motivations behind designing a low interaction client honeypot:

- We required a low interaction client honeypot using minimum resources which could be deployed on remote locations with limited resources.

- The client honeypot had to be fast in its operations (e.g. retrieval and analysis).
- The client honeypot had to use multiple analysis engines to reduce false negatives and allow offline analysis. It should also allow additional analysis engines to be added.
- The client honeypot had to allow various browser configurations and support specific functionalities (e.g. redirections, cookies and session handling etc...).

Current available client honeypots were either too limited in their functionality, or the addition of further improvements to their non-modular designs, in order to meet HAZOP requirements would require significant time.

The main reason for developing a low interaction client honeypot was due to its detection speed and low required resources. Retrieval and analysis speed are crucial factors in a real-world experiment involving millions of web sites. As mentioned in chapter 3 section 3.8, a high interaction client honeypot requires up to 17 seconds to scan a single URL. For a million dataset, that would translate to 196 days versus few days for a low interaction client honeypot (e.g. 1.9 days for YALIH).

Integration of high interaction client honeypot, either standalone or as a hybrid system in combination with low interaction honeyclients was also not feasible in the experiments performed in this research due to lack of remote resources and the cost of required services. Our experiments in chapter 6 relied in retrieval from remote regions such as Chile where commercial virtual private server are scarce and limited in resources and do not meet the requirements to run high interaction client honeypots. High interaction client honeypots rely on real operating system, browser, browser plugins and application to interact with a remote web site and utilise state based analysis engines for detection.

Low-interaction client honeypots generally rely on signatures to identify potentially malicious web sites. Signature based systems in general provide fast scanning speed with very low false positive and higher false negative rates of detection. The high false negative rate is due to several main issues: (1) an inability to identify zero-day attacks where signatures do not yet exist; (2) an inability to deal with variations of existing attacks (e.g. obfuscated attacks); (3) their reliance on a single analysis engine in preference to lower scanning time; and (4) inability to simulates services of a real browser and operating system and therefore being detected.

YALIH was developed in early 2012 and has been maintained to present day (2017). YALIH was offered as an open source software for public use in 2014 to allow users to participate in its development and maintenance. Several reliability and stability improvements have been made based on the feedback received from other users.

Chapter 3 section 3.8 discusses the design and functionalities of low and high interaction client honeypots. In this chapter we discuss the design and development of our low interaction client honeypot YALIH which attempts to address the discussed shortcomings of low interaction client honeypots. YALIH attempts to integrate HAZOP's recommended controls to mitigate and reduce any potential threats identified in table 5.1 and 5.5.

5.1 Our Honeyclient's Design

Client honeypots are generally made up of three main components that manage the entire process of web site retrieval and analysis [26]. These components are: the (1) URL Collector (Queuer); (2) Visitor; and (3) Analysis Engine. The following discussion about YALIH is structured around describing each of these components. YALIH has a modular design to allow additional URL databases and analysis modules to be added in the future. Figure 5.1 shows the overall design of the YALIH honeyclient.

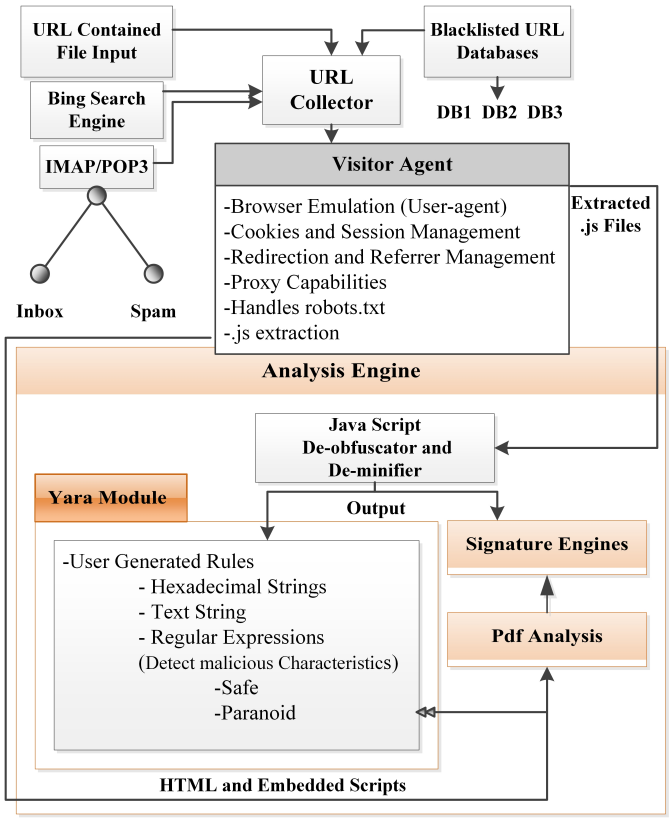


Figure 5.1: Overall design of YALIH honeyclient

5.1.1 URL Collector

The URL Collector module is responsible for gathering the hyperlinks of potential malicious web sites to be visited by the honeyclient. Depending on the implementation, these hyperlinks can be gathered from single or multiple sources. For example, HoneyC and Monkey-Spider gather hyperlinks from search engines while Monkey-Spider also employs crawlers [25]. Web crawlers, however, increase the chances that a honeyclient will be detected by a malicious web server. Crawlers may be detected because they initiate a high number of connections and fetch many URLs in a very short period of time. This behaviour can be detected by a malicious server through IP-tracking and may result in serving the honeyclient with benign

content.

YALIH uses the Mechanize visitor module which provides built-in functions for crawling and extracting links from the visited web sites to a depth defined by the user [127]. The crawling functionality is not used in our design however due to the risk of detection, as described above. YALIH's crawling functionality however does allow parsing the content of a web site, and extraction and retrieval of particular content, specified by its extension. This functionality allows YALIH to retrieve and store all embedded .exe or .pdf files which meet the size requirements specified by the user. The size specified can for instance indicate all executable files (e.g. .exe) smaller than 1 MB, which generally covers most malware and rogue malware. The URL Collector module is responsible for gathering the hyperlinks of potential malicious web sites to be visited by the honeyclient. Depending upon the implementation, these hyperlinks are gathered from single or multiple sources. HoneyC and Monkey-Spider for example use results from search engines, while Monkey-Spider also employs crawlers [25]. URL collection and queuing in YALIH can be achieved in several different ways:

1. **Direct Feed** - A user may input a single URL directly, or use a file containing a list of URLs to visit. The list is inspected beforehand to remove duplicates.

2. **Search Engine API Integration** - Search engine APIs are possibly the most widely used technique for URL collection by client honeypots. Monkey-Spider for instance utilises Bing and Yahoo APIs for URL collection. YALIH incorporates the Bing search engine into its design. Currently Bing is the only search engine which allows free retrieval of results using provided API. Bing however only allows a limited number of queries for a particular developer in a certain period of time (i.e. 2000 queries per month). It is also to be expected that search engines return few malicious web sites, as various strict

security filtering are applied to their returned results.

3. **Spam and Phishing** - Spam emails are a well-known source of malicious content, luring users into browsing web sites set up for phishing scams, or containing malware and drive-by download attacks. YALIH's unique spam and phishing component integrates URL harvesting from email addresses using the IMAP protocol. Provided with email credentials and supporting IMAP protocol, the honeypot fetches content from the user's Inbox and Spam folder – if available – and excludes the attachments. To minimise the traffic, the "Sent folder" and user-created folders are omitted. The consensus is that users are wary of the nature of emails filtered and placed on their IMAP generated folders, and would not knowingly send out emails containing malicious content. The fetched emails are then parsed for URLs, duplicates removed and sent to the visitor agent for content retrieval.

4. **Malicious Web site Database** - To assist the researchers with exploit code collection and illustrate the effectiveness of the honeypot, the blacklisted URL collection module downloads suspected malicious web site lists from three public databases and analyses them accordingly. These databases are

   (a) hosts-file.net [161]

   (b) malwaredomains.lehigh.edu

   (c) adblockplus.org

   The databases are updated constantly and contain links to web sites serving drive-by exploits, malicious executables and malicious Portable Document Formats (pdf).

Procedures of the Queuer in URL retrieval and processing are as following:

1. Suspicious URLs are retrieved from several potentially malicious list of domains (e.g. hp-feeds, .br domain) or an input by user.

2. the queuer module reads the input file line by line and inserts them into a set. The duplicate URLs are removed at this stage.

3. Queuer module reads the configuration file and determines the number of URLs to be passed to visitor module.

4. Each URL is assigned its own thread and passed to visitor for retrieval

5. Extracted links from main URL are passed to the visitor agent upon each visit.

5.1.2 Visitor Agent

The visitor component in a low-interaction honeyclient is a virtual browser which mimics the functionalities of a real browser, fetching the content of the potential malicious web site and storing it for the analysis engine. YALIH's visitor agent is based on the Mechanize module, which is available for various programming languages and provides a wide range of capabilities. The Mechanize visitor agent allows a user to set different personalities in the form of browser headers while communicating with a malicious web server. Browser headers may represent various browser characteristics such as type, version, underlying operating system, language and extensions. Exploit code targeting a specific vulnerability of a particular browser type or version may not execute on a different browser unless it targets a common rendering engine or extension, such as Java Runtime Engine. Multiple exploits targeting a specific browser and operating system are delivered to a user during a single visit to increase the probability of infection. Browser Exploit kits which are widely available online, facilitate and automate this feature. Examples of such tools are CrimePack,

Phoenix Exploit Kit, Eleonore exploit kit, IcePack and the Blackhole exploit kit.

The Mechanize module in the YALIH honeypot is configured to provide various capabilities to minimise detection of virtual browsers through cookies, redirection and session handling. The referer setting is also tuned to minimise exploit delivery blocking based on the referer data. Browser exploit kits such as the Blackhole allow an attacker to block the delivery of an exploit to a visitor if the referer does not match a predefined set of referers, or matches a set of referer domains for which the existence of a honeyclient is suspected. YALIH allows the user to automatically set the referer to predefined strings of:

1. A particular search engine results
2. The top-level domain of the visiting URL, or manually set referer string

Figure 5.2 shows a snapshot of the referer blocking features of the Blackhole browser exploit kit. This exploit kit uses a blacklist (Blocked Referers) and a Whitelist approach (Allowed Referers) approach for referer cloaking. The Blackhole exploit kit also allows traffic filtering and cloaking for automated bots such as search engine crawlers, specific IP addresses (e.g. security companies) and clients who use Tor network to request content. The visited and fetched web site content is saved on the local disk for analysis.

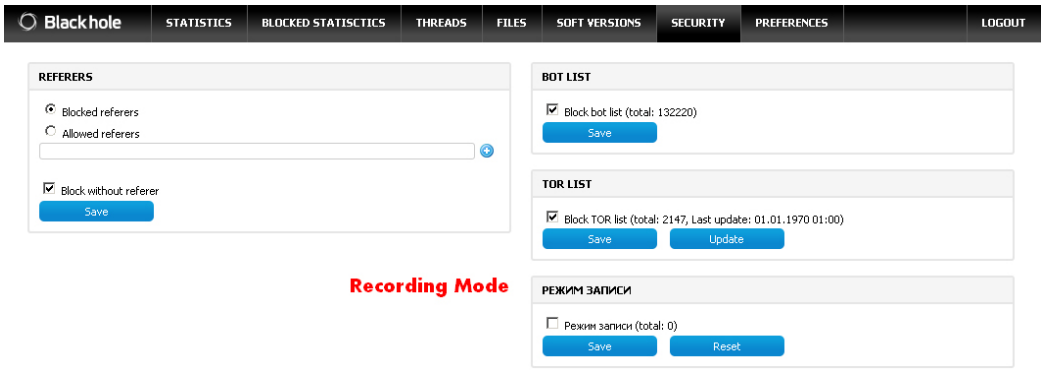


Figure 5.2: Referer and IP blocking features of Browser Exploit kits

The following are the steps in the process of retrival and analysis of a URL dataset input by the user with YALIH.

1. Visitor module fetches the URL list from the queuer and visits them sequentially.

2. Visitor module determines if crawling should be performed for each URL by reading the options passed to YALIH.

3. Visitor reads the configuration file and determines the user-agent value, accepted language and referer value to be passed to the remote web site.

4. Visitor initiates a request to web server and fetches the content.

5. Visitor checks the web server's response for redirections, request to place cookies and refresh and performs them accordingly.

6. Visitor scans the web site content and extracts embedded JavaScripts and hyperlinks to malicious file types -defined by user- and retrieves them accordingly. The files are written to local drive

7. Retriever module extracts JavaScripts and .js files and attempts to de-obfuscate and deminify them using the Rhino JavaScript engine.

The output of de-obfuscation and de-minification are written to local folder for each URL.

8. Visitor logs the URLs, fetched content and responses from the web server to the debug log file. If an error occurs during the retrieval due to DNS error or an offline web site, the visitor agent records the URL in a file for later analysis.

5.1.3 Analysis Engine

The analysis engines in low-interaction client honeypots generally identify malicious content through single or multiple detection techniques (e.g. Antivirus Engine, Snort Rules). The analysis engine on YALIH relies on several detection methods:

De-Obfuscation and De-Minification of JavaScript

Scripting languages provide numerous obfuscation techniques that allow attackers to encode the exploit code and bypass signature-based detection, even attacks for which signatures are available. The obfuscation can be in the form of base64 encoding, simple misplacement of characters, or strong encryption algorithms [23, 128, 129, 77]. Minification, while being used mostly to compress the size of the script, can also be utilised to bypass simple signature-based detection tools [23].

The emphasis on detection of attacks on JavaScript, besides its capability for obfuscation, is because JavaScript engine vulnerabilities have accounted for a vast number of exploits in the last few years, and provide a multi-platform environment for malicious code to be executed on nearly all browsers and operating systems. In 2012, JavaScript replaced Adobe Flash as the most targeted platform, accounting for 50% of all the targeted exploits [130].

YALIH parses, extracts and downloads the JavaScript files (.js) embed-

ded within the visiting web sites. It is assumed that malicious web sites avoid placing the exploit code directly within the HTML code to avoid detection. The downloaded .js file is rendered by the Rhino Java engine which de-minifies and de-obfuscates the JavaScript scripts encoded using popular techniques (i.e. base64, base62, numeric (base10), high ASCII (base 95), string substitution and/or concatenation). Once the output is generated, it is written to a file which is then scanned by multiple antivirus and signature/pattern matching tools. Figure 5.3 shows an example of a simple JavaScript code which generates an alert on a client's browser and redirects the browser to "http://www.google.com" domain. Minification removes the whitespaces while obfuscation hides the true nature of the script. Signatures generated for this script will vary significantly between the obfuscated and the original (De-obfuscated) script and slightly between the original and the minified script.

De-Obfuscated and Beautified Script

```
function ShowAlert()
   {
    alert('an alert!');
    window.open("http://www.google.com/")
   }
```

Minified Script

```
function ShowAlert(){alert('an alert!');window.open("http://www.
   google.com/")}
```

Obfuscated and Minified Script

```
eval(function(p,a,c,k,e,d);if(!''.replace(/^/,String)){while(c--)
   {d[c]=k[c]
||c}k=[function(e){return d[e]}];e=function(){return'\\w+'};c=1};
while(c--){if(k[c]){p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),
   k[c])}}
return p}('3 2(){0(\'10!\');4.5("8://7.6.9/")}',10,10,'alert|an|
   ShowAlert
|function|window|open|google|www|http|com'.split('|'),0,{}))
```

Figure 5.3: Example of an original, minified and obfuscated JavaScript exploit

Multiple Antivirus Signatures

YALIH utilises a selection of available and established detection engines, combining an open source (i.e. ClamAV) and commercial antivirus attack signature (i.e. AVG) database. The AVG antivirus engine was selected primarily due to its free licensing and availability for Linux operating systems. Other antivirus engines available for Linux operating systems were also tested but later removed to reduce the scanning time, as they did not provide sufficient detection improvement to balance the delay they introduced. The modular design of our honeypot however allows for several scanning engines and specifically antivirus engines to be added, with nearly two lines of code available. Multiple antivirus engines comprise part of YALIH's overall analysis engine, and are complemented by Yara [131]. Yara is a malware identification and classification library designed

to detect and classify malware samples using binary, textual and regular expression pattern matching. It is utilised in systems such as VirusTotal and Cuckoo SandBox [132].

Malicious Pattern Detection

Malicious web sites exhibit certain characteristics that set them apart from legitimate and benign web sites. Although some of these characteristics can be used in the design of complex web sites, their presence can be an indication of a malicious web site. Some of these characteristics are hidden, zero or small sized iframes, and availability of obfuscated scripts or shellcode patterns in JavaScript files. The Thug client honeypot has had a similar development timeline to our client honeypot and also integrates Yara, but only performs scanning to detect signatures of malicious web sites infected by browser exploit kits.

Utilisation of regular expressions by YALIH allows the system to detect any variations of these characteristics. As an example, the following rule is designed to detect hidden iframes or styles within the HTML code of the web site. Any single or combination of these can be used depending on the level of detection required by the user. The Yara rule in Figure 5.4 detects suspicious iframes using a combination of properties and attributes of HTML and Cascading Style Sheets (CSS) markup languages. These attributes allow an iframe's width and height to be set to a small number to hide it from a user (e.g. ”iframe width=”1” height=”1” which generates an iframe of 1x1 pixel) or hide the iframe entirely using ”visibility: hidden”, ”display: none” and ”style=hidden” attributes.

```
$i4 = /(<iframe.+?(?:visibility\s*:\s*hidden|display\s*:\s*none|(
    width|height)
\s*=\s*"[0-2]"|style\s*=\s*['\"]?hidden).+?<\/iframe>)/
```

Figure 5.4: Example of regular expressions to detect hidden or 0-2 sized iframes

Users are also able to develop their own signatures in Yara to match an exploit pattern in either ASCII, hexadecimal strings or regular expression (RE, Regex) format. For instance, a signature for the vulnerability "*Blackice Cover Page SDK insecure method DownloadImageFileURL() exploit*" can be a single format, or combination of these formats, in a web site content. The Yara rule in Figure 5.5 will generate an alert for any content which matches two of the following rules: 1) The keywords "vbscript", 2) Lower or upper case "BiDib.dll" and 3) Lower or upper case "DownloadImageFileURL". These signatures must be unique to this exploit to avoid raising false positive alerts. Yara rules are easier to create than the Snort rules of HoneyC and vulnerability modules of Thug and PhoneyC, as only a set of strings and boolean expression to determine their logic are required. These signatures are language independent and can detect signatures in malicious code written in any client-side scripting language (e.g. JavaScript, VBScript).

```
Rules:
$b1 = "vbscript" fullword
$b2 = " BiDib.dll" nocase
$b3 = "DownloadImageFileURL" fullword nocase
condition:
two of them
```

Figure 5.5: Example of user created string pattern rule to detect a specific attack

Portable Document File (PDF) Analysis

Portable document files or pdfs are a popular file type for storing and exchanging documents. Their popularity with end users and integration into popular browsers makes them high value targets for attackers to exploit. Infection through pdf documents can occur through various techniques:

1. Exploiting the vulnerabilities of the reader application associated with the file
2. Exploiting the browser plugin associated with the reader
3. Exploiting the vulnerabilities of the files, codes and associated applications embedded within the document

Pdf's open standard allows other file types and codes to be embedded within the document. The embedded content can be executed upon reading the document file. Execution of some executable files types is generally blocked by vendors, but other high risk scripting languages are allowed to fully load and execute [133, 134]. Pdf files for instance allow execution of embedded flash and JavaScript code, both of which are well known for their high number of vulnerabilities as explained in CVE-2015-5119 and CVE-2015-6712 respectively.

Malicious JavaScript code in a pdf document can be detected through extraction and analysis of the embedded code by various static or dynamic analysis engines. YALIH attempts to extract and de-obfuscate embedded JavaScript code using a popular pdf unpacker and analysis tool [135] and perform detection on the output using the signature and pattern matching detection engines.

The following excerpt is an example of a malicious JavaScript code embedded in a pdf document (Figure 5.6). The malicious code described in CVE-2013-2729 targets BMP/RLE heap corruption vulnerability in all Adobe reader versions 9.303 to 11.001. The specified vulnerability causes integer overflow by embedding two malformed BMP files with a repeating patterns of base64-encoded *"/AAAC/wAAAv8AAAL"*. Adobe Reader does

not perform validation while parsing the embedded BMP RLE encoded image. Arbitrary code execution is proved possible after the malicious bmp image triggers the heap overflow. The associated reader application crashes and exits with ”page not found !” error message [136]. Instances of the malware have been observed attempting to contact remote servers and install ZeuS-P2P/Gameover rootkit [137]. Malware in the Zbot/Zeus family are designed to steal information, primarily login credentials. YALIH detects such an attack through both signature-based detection and a Yara rule designed to detect long base64-encoding of de-obfuscated JavaScripts.

```
<script name="im" contentType="application/x-javascript">
String.prototype.trim=function(){return this.replace(/^[\s\n\r\t
    ]+|\s\n\r\t]+$/g, '');};

<image>
Qk0AAAAACgAUAAAAAABAAAAALgEAAAEAAAABACQVJHQEAAv8AAAL/AAAC/
    wAAAv8AAAL /AAAC/wAAAv8AAAL/AAAC/wAAAv8AAAL/./AAAC/
    wAAAv8AAALAAIAqwACAKsExMTExMTEw=
</image>
. . .
       var o = {
          "Reader": {
             "9.303": {
                "acrord32": 0x85,
                "rop0": 0x14BA8,
                "rop1": 0x1E73AF,
                "GMHWA": 0x7F245C,
             },
. . .
             "11.001": {
                "acrord32": 0xA9,
                "rop0": 0x19CBE,
. . .
               }}};
xfa.host.messageBox("Page not found !", "Adobe acrobat", 3, 1);
event.target.closeDoc(true);
```

Figure 5.6: Excerpt of an embedded JavaScript within a pdf file, targeting a heap spray vulnerability in Adobe pdf Reader

The following are the steps in the process of analysing a web site content by the analysis module of YALIH.

1. Analysis engine reads the the YALIH's temporary folder for retrieved URL contents.

2. All files in the designated dataset's folder are matched against attack signature databases and analysed for patterns of malicious characteristics.

3. Portable Document Files (pdf) are passed to pdf analysis script for extraction of potential embedded JavaScripts. Extracted JavaScripts are subsequently scanned as specified in step 2.

YALIH provides a large number of capabilities in terms of queuer and visitor agents and its modular design allows for integration of many detection and analysis engines. In Table 5.1 we summarises the hazards which could affect the result of the study, introduced by the modules of utilised client honeypot in the process of retrieval and analysis mentioned in previous section. We illustrate what actions are taken and controls are placed to minimise each hazard. Such analysis is important in this study as YALIH will be extensively used in the design of experiment in the following chapters and specifically in chapter 6. For instance, We identify that a hazard could be introduced by the visitor module due to the lack of redirection handling capability of the module. As multiple redirections is a popular techniques used by malicious and spam web sites to hide their true nature, this could result in a large number of malicious web sites going undetected. We identify that in our scenario such a hazard would represent a medium severity level represented by the number ″2″ if the hazard were to be ignored and a low likelihood represented by the number ″1″. Low likelihood is due to the fact that our simulated client honeypot is capable of handling all server-side redirections which are popular with malicious web sites and several non-obfuscated client-side redirections. The

required action represents the action we have taken to completely overcome, mitigate or minimise the threat by using a browser with redirection capability.

Table 5.1: Hazards introduced by our low interaction client honeypot and its modules

Study Node	Guide Word	Deviation	Possible Cause	S	L	Possible Consequences	Required Action
Queuer	No	No malicious website is detected	Queuer Module fails. Software bug.	3	1	URL retrieval fails	Debugging and functional testing.
	Less	Less malicious websites are detected than expected	Duplicate URLs in the input dataset.	3	1	Benign content is delivered on second retrieval and overwrites the first URL files resulting in false negative	Duplications be removed from input URL set.
	More	More URLs are passed to the visitor than visitor can fetch	Bulk processing of URLs.	3	2	Process is not able to allocate a thread to each URL	Lower the number of threads running at a time.
		Visitor processes more URLs than is fed	Bulk processing of URLs.	3	2	May cause the visitor to be identified as a crawler	Introduce a delay between creating each thread.
	Biased	URLs selected by the Queuer introduce bias	Small number of URLs is visited in the study	3	1	Threat to external validity	Select large dataset of URLs to represent malicious website behavior.
			Crawler module fails resulting in a small set of URLs.	3	1		
			Input URLs are selected from sources which have placed their own filters on the dataset (e.g. search engines).	3	2		Random sampling of input URLs from a larger population that the measurement shall be generalized to.
			URLs are only selected from few sources, a single top level domain or language. Particular type of URLs are selected which exhibit different malicious behavior.	3	2		
Visitor	More	More malicious websites are classified than expected	A website is visited more than once by a node URL is duplicated in the input list. URL is extracted several times from multiple websites by the crawler module.	2	1	Malicious websites tracks and determines IP address history and serves benign content resulting in false negative	Duplications be removed from input URL set. Crawler-extracted URL and its content is fetched but not over-written if URL content is already saved.
	Other than	Malicious website is visited with an IP address from a location other than the location malware was designed for	A request is initiated from a location other than intended by the malicious website.	3	2	Benign content is delivered resulting in false negative	Place retrieving nodes in similar geographical location.
	Less	Less number of websites is classified as malicious	Visitor fails to properly set the cookies while visiting.	3	1	Benign content is delivered resulting in false negative	Use a browser that can handle cookies.
			Visitor fails to extract all JavaScript (.js) files.	2	1	Website is not detected as malicious	Use multiple script extraction algorithms.
			Visitor fails to properly handle redirection or refresh.	3	1	Benign content is delivered resulting in false negative	Use a browser that can handle multiple redirections and refresh.
			Visitor cannot execute a function specified by the website and fails to trigger malware delivery.	3	2		Use a real or higher interaction simulated browser.
			Visitor does not expose a vulnerability the remote websites seeks to exploit				Integrate vulnerability modules into the visitor agent.

Study Node	Guide Word	Deviation	Possible Cause	S	L	Possible Consequences	Required Action
Analysis Engine	Less	Less number of websites is classified as malicious	Attacks are zero-day exploits. No signatures are available for the attacks.	3	2	Threat to external validity, May result in false negative	Create a broad signature or pattern. Add additional detection engines and update the existing database. Determine the false negative rate by manually running tests on a controlled set of malicious files.
			Malicious JavaScript is obfuscated to avoid detection by signature-based engines			Website is not detected as malicious resulting in false negative	Perform de-obfuscation and de-minification of JavaScript files
			Embedded files are larger than a size specified by user				Specify a large size higher than average or likely size of online malwares.
			Malicious JavaScript is embedded within the pdf or document files				Extract and de-obfuscate any hidden scripts within .pdf files.
	More	More webpages are classified as malicious	Incorrect signature for an attack	3	1	Threat to external validity, May result in false positive.	Determine the false positive rate by manually running tests on a large controlled set of benign files. Create rules which incorporates higher number signatures or a combinations of pattern to lower false positive.

5.2 Reliability Component

A client honeypot's retriever or analysis engine process may cease to operate normally for several reasons such as programming error or host operating system faults. Although it is assumed that the honeypot will be monitored by an administrator on a regular basis, any delay in retrieval of a set of web sites may introduce bias into an experiment and cause varying results and incorrect analysis. This issue is especially concerning in a distributed setup where multiple instances of the client are fetching content. A module has to monitor the operation and notify the administrator should the process stop or fail. YALIH's monitoring process monitors the operation of the client honeypot on predefined intervals set by the user. It utilises psutil API to verify the honeypot's operation during retrieval and analysis phases. Psutil is a cross-platform and system utilities library for retrieving information on running processes and system utilisation. During retrieval, the monitor agent verifies the operation of client honeypot through:

- **Monitoring the running process -** It verifies if the process is running normally. It also checks if its PID has been reused by another process, in which case it returns False. It initially records the number of URLs and the current URL the honeypot is retrieving and checks the set periodically to assure the job is completed and the end of the set is reached. If the retriever module stops on a selected URL, the monitor agent will restart the retrieval process from the last visited web site and notifies the administrator by email.

- **Monitoring bandwidth usage -** The retriever's process may be halted without being killed on a host operating system. The monitor agent detects any such abnormality using the traffic monitoring function of the psutil API. If no connection (e.g. IPv4, TCP) associated with the retriever process on the predefined intervals is detected , the monitor agent will restart the retrieval process from the last visited web site

and notify the administrator through email accordingly.

Similarly, YALIH's process is monitored during analysis through process and CPU utilisation on predefined intervals. A notification is sent to the administrator and the process is restarted if an error is detected.

Reliability of YALIH's components in operation can partially be due to its nature as an active open source project in which numerous improvements have been made over the past two years and a large number of software bugs have been reported to the contributors and fixed accordingly.

5.3 Detection Accuracy and Performance

We used two control sets of data to measure the detection accuracy of our honeypot design; in an approach similar to that used by Cova and Krugel [138].

- A *"**clean dataset**"* containing 10,000 web sites gathered from Alexa's top one million most visited web sites
- A *"**malicious dataset**"* containing 110 exploits

Figure 5.7 shows the overall setup of the experiments and associated datasets used in this chapter to evaluate the detection, performance and emulation capabilities of YALIH against other popular low interaction client honeypots. The experiments in this chapter focused on determining:

- The False Positive rate of YALIH which refers to the number of incorrectly identified web sites as malicious
- The False Negative rate which refers to the number of malicious web sites not detected by our client honeypot
- The speed of detection

- The emulation and retrieval capability of YALIH in comparison to the top low interaction client honeypot (i.e. Thug)

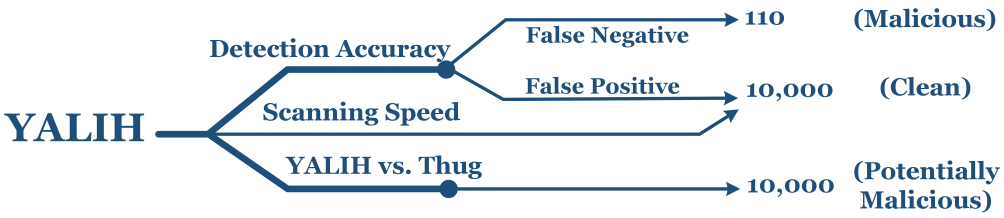


Figure 5.7: Overview of the tests done on performance and detection accuracy of YALIH

5.3.1 Detection Accuracy

The initial step was to determine the false positive rate of YALIH's individual analysis engines against our clean dataset. YALIH's antivirus detection engines were updated to the latest version and pattern matching rules were created for the Yara module. Pattern matching using regular expression involves rules to determine the static characteristics of a web site and its embedded scripts for identifying a malicious web site. The integrated rules were:

1. **Java Shellcode Pattern** – long sets of integers, special characters or string to indicate the presence of shellcode [139]

2. **Java obfuscation identified by "eval", "unscape" functions** – referred to as "dangerous functions", they can be used to encode and decode shell strings code to obfuscate an exploit [140]

3. **Java obfuscation using base64 encoding** – JavaScript malicious content can be encoded using base64 to obfuscate and avoid detection [52]

Decoded or hidden iframes are a popular method for delivering malicious content to a user browser by redirecting part of the browser to a malicious web site [140, 141, 142]. This is generally achieved through loading a malicious web site in an iframe with a zero or small size integer. Any web site with an iframe tag with one or more of the properties "Visibility: Hidden", "Display: None" or "Width and Height of 0" was considered suspicious. This rule was however removed in later experiments as it produced a high number of false positives. It was observed that hidden and small-sized iframes were widely used in large number of the web sites in our clean dataset.

Subsequent analysis of hidden iframes in the clean dataset revealed that they were widely used for seamless display of content and bypassing same-origin security policies for advertisement display purposes, and so were not necessarily a reliable indication of the maliciousness of a web site [143]. YALIH currently extracts and retrieves .js scripts which may be embedded within hidden or small-sized iframes, but the functionality for extracting other types of contents (i.e html links) was disabled based on our observation of follow-up links. Scanning was performed on the de-obfuscated and de-minified content, retrieved and saved on local disk by YALIH's visitor agent. Signature-based antivirus engines achieved a 0% false positive rate on the clean dataset. A false positive (FP) occurs when a file is incorrectly flagged as malicious, when in fact it is benign. Regular expression rules created to detect shellcodes and base64 obfuscated scripts generated 47 false positive alarms on the clean dataset (10,000 web sites). Table 5.2 shows the false positive rate of several client honeypots and detection engines of YALIH on the clean dataset. Pattern matching detection rules of YALIH can be tuned to detect higher number of attacks in expense of higher false positive rate.

Table 5.2: False positive rate of several client honeypots and detection engines of YALIH on the "clean dataset" (10,000 Web sites)

	Monkey-Spider	**HoneyC**	**YALIH**		
Detection Engine	ClamAV	Snort Rules	ClamAV	AVG	Pattern Matching
False Positive	0.0%	0.0%	0.0%	0.0%	0.47%

Determining the false negative rate of our YALIH honeypot involved scanning a malicious set of web sites containing 110 drive-by download exploits we had collected. YALIH's performance was also measured against other open source client honeypots (i.e. Monkey-Spider, HoneyC, Thug). MonkeySpider and Thug allow analysis of fetched web content on the local disk. On the other hand HoneyC's analysis engine operates based on Snort rules and therefore scans network streams for signatures of malicious activity. To emulate a malicious web server for analysis by HoneyC, a web server was placed on the local network and malicious files in our dataset were uploaded. HoneyC's rules were subsequently updated to the latest rules obtained from Snort's official web site and configured to retrieve the malicious content from the local web server. It should be noted that individual signatures were not created in Yara for each exploit and only regular expression rules were generated to detect malicious characteristics of a web site or script. Table 5.3 shows the false negative rate for each client honeypot's analysis of the malicious dataset and the corresponding analysis time. A low false negative is desirable as it represents the rate of missed attacks by the analysis engine.

Table 5.3: False Negative rate of several low interaction honeyclients in comparison to YALIH, on the "malicious dataset"

	Monkey-Spider	**HoneyC**	**Thug**	**YALIH**
False Negative	53.6%	100%	35.4%	19%

HoneyC's subpar performance might be due to the fact that Snort as an intrusion detection and prevention system primarily focuses on server-side attacks and less on client-side browser-based exploits and malware. Its open source rules are also not updated as frequently as those of popular antivirus engines.

5.3.2 Scanning Speed

Low interaction client honeypots and specifically those based on signature databases have relatively very low scanning time. The scanning time depends on the size of the signature database; however the differences are so insignificant they can be ignored. Table 5.4 shows the average scanning time for URL on the clean dataset. The experiment was run on a 3.0 GHz system with 8 GB of memory and 64 bit edition of Debian Linux.

Table 5.4: Average scanning time of each URL on the "clean dataset" (10,000 web sites)

	Thug	**Monkey-Spider**	**HoneyC**	**YALIH**	**Honeyware**
Time per URL (Seconds)	1.43	0.147	0.109	0.167	57

YALIH achieved a considerably lower scanning time than other low interaction client honeypots such as Honeyware [53]. Capture-HPC high interaction client honeypot achieved an average visiting time of 10 seconds and an average revert time of 15 seconds per URL. A hybrid system could utilise YALIH's fast scanning capabilities and a selection of broad pattern matching rules to detect suspicious content as the front detection system, and then pass them to a high interaction client honeypot for further analysis. The broad pattern matching rules of YALIH (e.g. hidden or small-sized iframes, <object>, <embed> tag variables) ensure an initial minimum false negative rate, while further scanning with the slower high

interaction client honeypot reduces the false positive rate within the initial batch of detected web sites.

5.4 Signature Generation

Yara signature generation is a capability which allows YALIH to improve the detection of attacks based on variations of a malicious code. This is achieved in conjunction with signature databases of antivirus engines that identify malware families with a very low false positive rate. Malicious files detected by antivirus engines that belong to the same family of malware or threats are isolated, placed into a directory and parsed. A set of common keywords/variables – present in all files within that family of malware – are extracted and used to create Yara rules (Figure 5.8). This is believed to increase the false positive rate for the client honeypot, but ultimately results in a lower false negative rate of detection as higher numbers of files are matched and selected for further analysis. Higher numbers of files detected in a single family of malware, will result in more accurate rules and consequently lower false positives. Users may also define the number of conditions and keywords to adjust accuracy.

Table 5.5 summarises the hazards and biases which could be introduced by the retrieval process, subjects of the study and intermediary devices, the possible consequences of those hazards and the action taken on our side to mitigate or minimise them. For instance an unreachable DNS service could produce a false negative alert represented by ”severity=3” which would potentially undermine the internal validity of our experiment. The likelihood of such a hazard is minimised significantly by using multiple high profile and reliable DNS servers (i.e. Google DNS, OpenDNS).

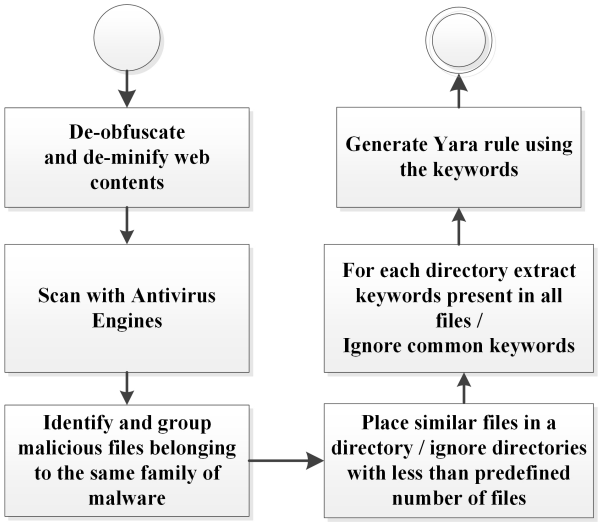


Figure 5.8: YALIH's signature generation procedures

Table 5.5: Hazards introduced by components of our experiments and intermediary devices

Study Node	Guide Word	Specific Deviation	Possible Cause	S	L	Possible Consequences	Required Action
HTTP Protocol	No	Referer value is not available, set, transmitted or is omitted	Referer value is not set by the host, transmitted or omitted by intermediary devices	3	2	Access is denied, Benign content is delivered resulting in false negative	Referer should be properly set in the honeyclient's setting and ensure its transmission to the remote server
	Other than	Referer value is different from the value malicious website is set to trigger attack	Referer value is not set correctly	2	2	Access is denied, Benign content or different malware is delivered resulting in false negative	Referer should be adjusted to the likely setting which may trigger attack (e.g. Google search results)
	Early, Late	Contents were retrieved earlier or after the traffic hours	Content retrieval is performed on incorrect times	1	1	Benign content is delivered resulting in false negative	Content retrieval should be adjusted to reflect the traffic hours of locations the website domains are located. The introduced risks are however trivial and therefore negligible
	Late	Website retrieval is timed-out	HTTP Server is busy. Connection is slow	3	1	Content retrieval is terminated	Increase the assigned connection time-out to a higher value per URL
	Biased	Higher or lower number of attacks are detected than expected	Configuration of operating system, browser, plugins and accepted language will bias detection toward malicious webpages which target those attributes	3	3	Threat to external validity, false negative	Document operating system, browser, plugins used and do not generalize beyond the experiment's configurations
DNS Protocol	No	Unable to translate address	DNS service is unavailable.			Address is not translated, Content cannot be retrieved and results in false negative	Mitigation through utilizing multiple public DNS servers
Intermediary Devices	No	URL content is not fetched	An intermediary device blocks access to the website	3	1	Malicious content cannot be fetched, resulting in false negative	Placing client honeypot behind firewalls and proxies should be avoided
	Less	Lower number of malicious websites are detected than expected	Traffic passes through an intermediary device, assigned a single IP address which is tracked by the malicious website			Benign content is delivered resulting in false negative	Placing client honeypot behind firewalls and proxies should be avoided
Website	No	Website cannot be reached and content be fetched	HTTP Server not available	3	2	Malicious content cannot be fetched, resulting in false negative	Multiple retries of remote website and logging unsuccessful retrieval attempts and error codes
	No	Malicious website is not detected	Malicious website detected the presence of a client honeypot			Benign content is delivered, resulting in false negative	Imitate realistic setup and behaviour of a general vulnerable user
	Unreliable	Website content vary each time the URL is visited	Web page denoted by URL is sourced from different web servers each time a request is made due to legitimate reasons (e.g. load balancing) or malicious activity	3	1	Threat to external validity, false negative	Determine if a website is a part of fast flux networks and repeat collection for each server
	Unreliable	Website content changes (malicious content is removed between each visit)	Webmaster removes the malicious content			Classification made may no longer be valid	Perform subsequent visits simultaneously from multiple servers and detect inconsistency. Save website content offline for later analysis

5.5 YALIH vs. Other Low Interaction Honeyclients

The YALIH client honeypot was developed to assist with retrieval and analysis of URL datasets in the upcoming experiments. Although there are several low and high interaction client honeypots available, the decision to design our own system was based on the following requirements:

1. **Usability**
    - Familiarity with the system, its components and capabilities
    - Design of a system to meet our specific requirements
2. **Modularity and development**
    - Knowledge of the underlying code and ability to detect and remove software bugs
    - A modular client honeypot to allow adding or replacing various queuer, retriever and analysis modules
3. **Multiple detection engines and analysis**
    - The honeypot had to allow offline analysis and support multiple detection engines
    - Detection engines to allow customised and configurable levels of detection
    - Capable of de-obfuscating JavaScript files
4. **Imitation capability**
    - A client honeypot capable of mimicking an average user's behaviour in visitation pattern, and allowing various configurable settings to overcome detection mitigation strategies by malicious web sites. These settings include: referer modification, cookies, redirection, refresh, session and robot handling

5. **Resources**

   - Due to the large number of experiments and URLs in our datasets, we required a lightweight client honeypot which could be installed on multiple remote locations with limited resources in terms of processing and storage.

The main motivation behind utilisation of low interaction client honeypots in large studies is their speed in retrieval and scanning. Retrieval and analysis of a single URL by a high interaction honeyclient may take from several seconds to a few minutes, depending on the content of the web site and available processing power [122, 31]. A large dataset (e.g. one million web sites) of input URLs will cause a significant delay between each retrieval phase, which results in inaccurate analysis and a lower number of detected malicious URLs due to web sites dropping offline between successive retrievals. However, a low interaction client honeypot's retrieval is only limited by the experiment setup's bandwidth while its analysis module can perform detection in fractions of a second per URL (e.g. 0.167 seconds per URL for YALIH).

The capabilities already mentioned were some of the initial requirements for the experiments performed in the following chapters. During the experiments for design of an architecture, several other capabilities of our client honeypot were used extensively and integrated into the final design. These include:

- YALIH's built-in crawler allowed our retrievers to mimic the behaviour of an average user or a search engine crawler through management of the time, number of retrievals per second and retrieval of embedded links.

- Integration of various reliability and notification mechanisms to minimise the cost of administrative overheads and reduce any analysis hazard due to a failed operation.

- Built-in compression and proxy support which were utilised in the design of our architecture.

Several other low interaction client honeypots such as HoneyC, MonkeySpider and PhoneyC were discussed in the literature review. The performance of some honeyclients (e.g. PhoneyC, HoneyC) were subpar in retrieval and detection capabilities while Thug, which was in development at the same time as YALIH, did not have the same reliability in retrieval of a large number of web sites due to high rate of crash during analysis.

In terms of detection and analysis, YALIH uses similar techniques to Thug for attack identification, including de-obfuscation of JavaScript, Yara pattern matching module and Libemu [137] for shellcode detection. YALIH, on the other hand, complements these by integrating two additional signature-based detection engines, while Thug supports VirusTotal analysis. Integration of VirusTotal was not deemed a requirement for YALIH since speedy offline analysis provides similar rates of detection. Table 5.6 summarises and compares the capabilities of YALIH and Thug client honeypots. While Thug has embedded vulnerability emulation, YALIH integrates various queuer features, signature generation and reliability components into the design.

Table 5.6: Comparison between features and capabilities of YALIH and Thug

Components and Capabilities	**Client Honeypot**	
	Thug	YALIH
Queuer		
Malicious database support	N	Y
Email/IMAP integration	N	Y
Search engine integration	N	Y
Bulk processing	N	Y
Visitor		
Browser personality emulation, redirection, referer modification, robot.txt, refresh and cookies management and handling capabilities	Y	Y
Proxy support	Y	Y
Integrated crawler	Y	Y
DOM event handling	Y	N
Flash, Java and Pdf plugin and vulnerability emulation	Y	N
JavaScript and executable file extraction	Y	Y
Minification and feature extraction	N	Y
Analysis		
JavaScript de-obfuscation	Y	Y
Integration of Yara malware detection library	Y	Y
Shellcode analysis	Y	N
Signature-based analysis	Online	Y
Signature generation	N	Y
Malicious pdf detection	Y	Y
Local analysis	Y	Y
Delayed execution	Y	N
Management and Reliability Modules		
Job status notification	N	Y
Process monitoring and failure recovery	N	Y

In order to provide grounds for utilisation of YALIH rather than other low interaction client honeypots in our experiments, we performed an experi-

ment involving a set of 10,000 suspicious web sites gathered from publicly available malicious domain lists. The intent of the experiment was to determine if Thug's [138] enhanced virtual browser would result in a higher rate of malware triggering and delivery. Thug's retriever was found capable of imitating several flash, adobe and JavaScript vulnerability modules and handling DOM events. These retriever capabilities are not implemented in the YALIH low interaction client honeypot. YALIH and Thug were installed on different machines on a network subnet with exactly the same underlying operating system and configurations. The setup for the experiment is shown in Table 5.7.

Table 5.7: Summary of YALIH vs. Thug experiment's setup

YALIH and Thug	
Underlying operating system	12.04
Imitated OS	Windows XP
Browser, version and language	Internet Explorer 8, en_NZ
Referer	https://www.google.co.nz/search ?q=&ie=utf-8&oe=utf-8&gws_rd=cr&ei=
Crawling	Disabled
Synchronised retrieval	Enabled
Redirection, referer modification, robot.txt, refresh and cookies management and handling capabilities	Enabled
Network Subnet	130.195.4.7*, 130.195.4.7*
Number of retrieved URLs	10,000
Thug only	
Acrobat reader	Version 11.0.06
Shockwave Flash	Version 10.0.64.0
Java engine	Version 1.6.0.32
Supported DOM events	

The malware delivered to Thug and YALIH were compared in two different ways:

1. Category of malware: the malware delivered to both the YALIH and Thug client honeypots was identical in type for common web sites.

2. Amount of delivered malware: Thug managed to detect 462 unique malicious web sites compared to 455 for YALIH. The malicious web sites detected in common also exhibited similar behaviour in terms of type and number of components belonging to each unique web site.

Overall insignificant variations were observed in the type and number of malicious components and web sites. Such behaviour and results could be due to the following reasons:

- Malicious web sites did not trigger malware delivery for specific plugin versions and vulnerability modules activated in Thug
- Malicious web sites employed techniques to detect the virtual environments in both client honeypots
- Malicious web sites served malware regardless of the client environment

5.6 Summary

We presented the design and analysis of a low interaction client honeypot that integrates a combination of multiple antivirus engines and pattern matching using string or regular expressions for detection. The honeyclient is capable of extracting embedded JavaScript files and performs de-obfuscation and de-minification of scripts. Subsequent signature and pattern matching analysis are performed on fetched contents and deobfuscated and de-minified scripts to detect signatures of attacks and static

and potentially malicious characteristics of a web site. Its embedded virtual browser reduces the possibility of detecting a virtual environment through handling cookies, redirection, sessions and imitating browser personality and referer settings. We demonstrated through experiments with benign and malicious datasets that our designed honeyclient was capable of achieving low false positive and false negative rate with very low scanning time compared to currently available client honeypot systems.

YALIH was designed and developed according to HAZOP methodology discussed in the previous chapter. Through HAZOP analysis and guide words, we identified shortcomings and potential flaws of our client honeypot's design and identified the possible deviations from the intended design; the reliable and efficient detection of malicious web sites. Subsequently, possible causes and consequences of each deviation were identified. We finally identified controls and procedures, and integrated them into the design of YALIH to reduce the effect and the associated consequence of each deviation.

Chapter 6

Experiments

In this chapter we perform large scale and real-world experiments and study of malicious web sites using various logical and physical attributes of a client honeypot. These attributes include HTTP referer, language, proxy information, network and IP addresses. We determine the effects of each attribute on malware delivery and how the behaviour of a malicious web site may change depending on these attributes. We also monitor the effects of tracking and blacklistings of IP addresses and subnets commonly found in browser exploit kits, by experiments on large datasets of potentially malicious web sites.

Datasets of potentially malicious web sites used throughout this chapter were gathered from public databases of malicious domains. Selection of suspicious URL dataset for retrieval can influence the number and type of delivered attacks. For instance a malicious Russian web site may only target users located within Russia or Russian speaking countries. These web sites' content are designed to lure specific users by serving appealing content in local language. Controls identified during the HAZOP analysis were therefore implemented to minimise any likely bias introduced into our study as the result of the input URL datasets. These controls were:

- The number of input URLs had to be high to represent a high external validity for the experiment. High external validity in an exper-

iment warrants a high degree of certainty about the generalisability of the findings.

- Input dataset had to contain random web sites from multiple domains unless specified in an experiment. For instance our referer experiment was performed on 500,000 potentially malicious web sites which included news, warez, gaming, shopping, adult and many country-specific top level domains. Language tracking experiment on the other hand was performed on a dataset of one million web sites from .ru top level domain. The dataset was randomly selected from a large dataset of 40 million .ru registered domains purchased from a web hosting service.

- Our data collection was performed over a course of one and half years. Very few malicious web sites would be online for such an extended period. A new set of input URL dataset was therefore gathered from multiple sources for every experiment. No identical datasets were used in two different experiments. Public sources of potentially web sites are listed in Appendix.

- IP tracking and cloaking is potentially a hazard which could result in a malicious web site serving benign content to our client honeypot and subsequent high rate of false negative. We implemented several controls identified in our HAZOP study of experimental setup to minimise such threat:

    1. Input dataset should not contain any duplicates as the 2nd retrieval may be served with a benign content.
    2. Malicious web sites may keep track of visiting IP addresses for an unknown period of time. The time frame between two experiments were therefore kept to a minimum of three months to minimise the probability of potential IP tracking effect. According to previous studies, malicious web sites have been observed

to have an active life of few days and no more than 18.5 days [162, 163].

3. Retrieving nodes were selected from low profile VPS services which had no affiliation with research or security organisation. Different VPS services were also selected for retrieving nodes of consecutive experiments. This setup allowed for varying IP addresses in case IP tracking was utilised by the target malicious web site.

The datasets used in the experiments discussed in this chapter are summarised in figure 6.1.

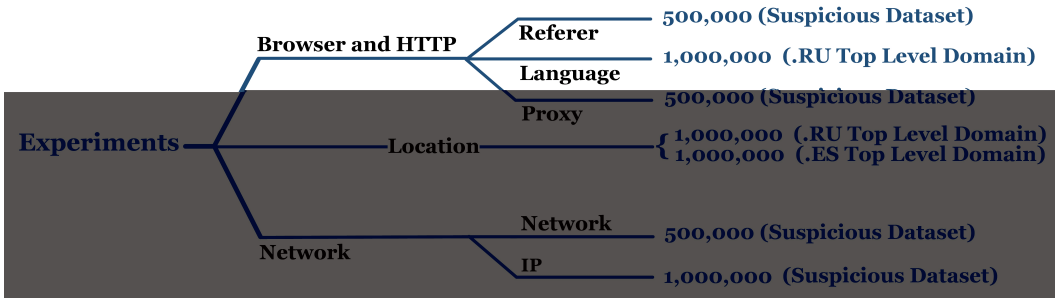


Figure 6.1: Summary of the datasets used in the experiments in this chapter

6.1 HTTP Referer Tracking

Referer, an optional attribute in the HTTP protocol header, identifies the origin of the request and the link used to direct users to the final resource. For instance, a user accessing resources on *"http://www.yahoo.com/"* by clicking a link located within the content of *"http://www.amazon.com"* would have *"http://www.amazon.com"* sent as the referer along with the request to the server. The Referer header allows a server to generate the lists of back-links to resources for various purposes (e.g. interest, marketing, logging and optimised caching) [91]. For example, referer is widely used in marketing and pay per click web sites to track incoming traffic. In addition, referer can be used to partially mitigate a cross site scripting attack by verifying that it originates from one's own domain [74, 75, 76] and as a means of weak authentication to allow access to resources by a selected list of gateway/login pages. However, referers can be modified and spoofed by an end user, applications and intermediary devices such as firewalls and proxies. Therefore, they are not a trustworthy authentication mechanism. Individual web sites hosted on a web server can employ server-side scripts to detect the referer attribute of a visiting request.

Server-Side Referer Cloaking involves the server checking the referer header field of an incoming HTTP request and delivering varying contents based on the value of the referer field [144]. Figure 6.2 shows the simplicity of server-side redirection.

```
$referrer = Request[HTTP_REFERER]
if (match("http://doorway-site.com",$referrer))
{
  Redirect to "malicious web site.com"
else
  Redirect to "Benign Web site"
}
```

Figure 6.2: Example of redirection based on client-side referer retrieval

Referer is also widely used in web spam cloaking and search engine optimisation [73]. Web spam cloaking is the practice of delivering different web site content based upon the type of visitor, for example whether it is a search engine crawler or a human visitor. The goal of the process is to obtain a higher ranking (and visibility) in search engines by placing significant number of keywords in the content delivered to the search engine crawlers and placing advertisements in the content delivered to the human visitor.

Content delivery based on referer attribute is not merely limited to web spam servers, and has also been observed in malicious web sites. A web site may examine the HTTP referer field of a request and depending on the rules set by the server or the exploit kit, redirect HTTP requests to either malware distribution web sites or deliver benign content based upon the value of the client-provided referer variable. Server-side referer cloaking techniques can, therefore, evade detection by a security tool (e.g. client honeypot) if its retrieval request does not include an expected referer value and is, thus, served with non-malicious content.

The challenge for web spam servers and malicious web sites is to distinguish an average user from security tools or search engine crawlers [145]. A frequently used technique by web spam pages is to examine if the request originates from a search engine result page. The server verifies if the referer URI contains the search engine's search keyword string. This could be an indication of a regular user searching for a keyword or a web site [72, 73]. A similar technique is used by malicious web sites [146], for example to identify whether visitors have arrived via a Google search because this makes it more likely the visitor is a human rather than a program such as a security tool. This can be achieved by matching the referer string against a known pattern for search requests generated by a given search engine. For example, many search requests contain alphanumeric sequences separated by multiple "&" characters which are indicative of delimited field value of search query strings [147, 80]. The following ref-

erer string is a concrete example of the referer string associated with a user directed to a given web site from a Google result's page (Figure 6.3).

```
http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&
    cad=rja&uact=8&ved=0CCsQFjAA&url
=http%3A%2F%2Fwww.whatismyreferer.com%2F&ei=_1dTU...
```

Figure 6.3: An example of a Google search results referer value

Malicious web sites may use a variety of referer attributes to target a specific visiting user. Determining the optimal referer which triggers the highest number of malware delivery can help with detection of malicious web sites and collection of malware samples. In this chapter, we study the malware delivery behavior (e.g. malware and drive-by download) based on referer setting using real-world datasets. We examine whether referer settings are exploited by malicious web sites. Based on logistic regression analysis of the experimental data, we evaluate and verify the optimal detection strategy based on the likely HTTP referer attributes of average users.

6.1.1 Background - Referer

It is known that the referer attribute has been exploited to facilitate spam delivery [71, 72, 73] and cross-site forgery attacks [74, 75, 76]. Referer validation and malware delivery have been observed in several browser exploit kits. They are a set of tools that contain a large number of vulnerabilities for popular browser, plugins and extensions. They facilitate the deployment and management process of malicious web sites on a large scale (e.g. Blackhole exploit kit) [6, 12]. These exploit kits allow the redirection of an unsuspecting user to a landing page by which malware will be delivered (or not) based on either a whitelist or a blacklist of various attributes such as referers, operating systems or network subnets.

Although the effects of referer on malicious content delivery have been mentioned in various literatures, we are not aware of any quantitative and scholarly study on the topic. A SophosLabs study of a SEO kit in 2010 identified Google.com as the top referer used by its victims [77]. Invernizzi et al. [78] attempt to identify the use of SEO (Search Engine Optimisation) which is used to improve the ranking of malicious pages by cybercriminals. If a cloaked malicious web site is located, the web site is then used as a "seed", in combination with other attributes, such as domain registration date, to locate the final landing page of the malicious domain. A landing page is the final page of a redirection chain which hosts the malicious file. The Strider Honeymonkey system sets the document referer to Microsoft's Live search engine, and performs queries of likely keywords targeted by spammers and visits the potential malicious web site [79]. This approach uses referer as one of the methods to detect cloaked web sites. In a study to detect web spam using Strider Honeymonkey, the authors concluded that that "some malicious web site operators are using search spamming techniques" to draw more traffic to exploit [72].

O'Dea [80] examines the evolution of rogue software and how they compare to other types of malware. The author provides an example of how current trends and one of the top search keywords was utilised in combination with SEO techniques to create high ranked search results in the Google search engine. The search engine returned results provided no information about the popular search term and directed users to malware web sites. In this case, referer tracking was applied to identify average users and evade search engine crawler detection.

6.1.2 Our Experiment Setup

Our experiment relies on the spoofing of HTTP referer header using the YALIH low interaction client honeypot. A low interaction client honeypot is a detection system comprising of a lightweight simulator which mimics

the behavior of a client browser and interacts with a remote web server. The response is subsequently examined with the client honeypot's detection engine to determine if any malicious code is present within the retrieved content.

YALIH allows the HTTP referer information to be manually specified within its simulated browser which is based on the Mechanize Library [127]. The Mechanize simulated browser can be considered the most capable simulated browser in low interaction client honeypots and can handle server-side redirections, user-agent imitation, cookies, robots and server-side refresh. Its built-in crawler is specifically used in this study to identify and retrieve executable files within a web site. YALIH was also selected as it allows addition of multiple detection engines to the fetched content which are saved locally. This includes multiple signature-based detection engines, pattern matching, anomaly detection and de-obfuscation of JavaScript through a built-in de-obfuscation and de-minification module. A high interaction client honeypot would rely on state changes for detection and requires a proxy setup for local content storage.

6.1.3 Network Profile and Input URL Dataset

Figure 6.4 shows an overview of our experiment's setup. Our retrieving clients are located within the same geographical location and experimental environments' attributes such a operating system, system and browser language and HTTP header information. Values of referer attribute explained in the next section represents the only independent variable in this experiment.

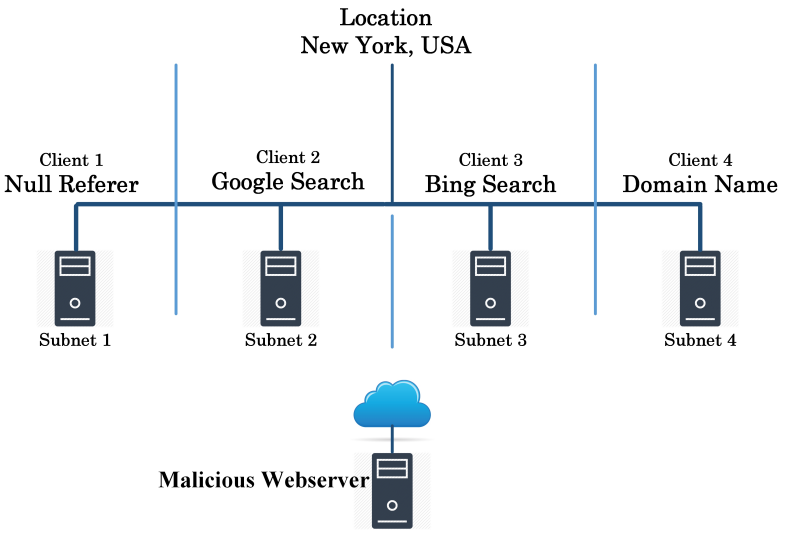


Figure 6.4: Our experiment's data collection setup

The datasets used in our experiment was collected from multiple sources and consisted of 500,000 web sites. We avoided collecting URLs via search engines to mitigate contamination of the results due to search engines applying their own filter to their results. Although our selection of web sites does not represent the entire Internet, we believe that 500,000 web sites is an adequate number to study the behavior of malicious web sites in the context of our study.

6.1.4 Referer Values

Similar to typical web spam experiments, several copies of a web site were retrieved from multiple retriever clients. Four different referer values were selected for this experiment. While there are many different ways of reaching a specific resource on a web site, the selected referer values represent the vast majority of values sent by browsers in HTTP requests. The specific classes of HTTP referer values which are used are:

1. **Null Referer (N):** A null referer value indicates that the web site visitor has either directly typed the URL address in the browser's ad-

dress bar, clicked a webpage saved in the brower's bookmarks or the referer information has been removed by intermediary devices.

2. **Domain Name (DN):** A domain name referer attribute indicates that the user arrived at the resource via either a subdomain or from the domain's main page. For instance, a user fetching the content from "http://ns1.drugstab.ru/drug s/topdrugs.asp" would have a referer set to "http://drugstab.ru".

3. **Google Search Result (G):** The referer indicates that the user arrives at the resources from Google's search result page.

4. **Bing Search Result (B):** The referer indicates that the user arrives at the resources from Bing's search result page.

In our analysis of the retrieved data, we examine the similarities and differences between the retrieved data from multiple locations. All client, protocol and environment variables are similar for all participating nodes therefore any differences in the maliciousness of the content can be associated with the varying referer value.

6.1.5 Number and Type of Detected Malware

Overall we detected 10,011 distinct malicious components belonging to 8,185 unique malicious web sites. Table 6.1 shows the number and type of malicious component detected, classified by the referer values. The types of malicious components were classified using YALIH's detection engine, which consists of a combination of two signature-based engines and an anomaly and pattern matching detection engine.

Table 6.1: Number and type of malware for each referer

Malware Type/Component	**Referer Values**			
	N	***DN***	***G***	***B***
Activex-Exploit	0	8	0	0
Adware	0	15	3	18
Adobe-Flash-Exploit	4	19	0	0
Embedded Shellcode/Exploit	144	488	424	391
Executable/Malware	319	899	939	932
JS-Trojan/Exploit	428	709	751	628
JS/Downloader.Agent	29	1823	1832	1781
JS/Obfuscated	9	51	23	24
Malicious-iframe	3247	3388	3999	3207
Phishing	1	5	15	15
PHP-Trojan/Exploit	3	4	6	6
Exploit.Kit Links/Scripts	663	1034	1304	1143
Script/PDF.Exploit	2	4	4	0
WebSpam	3	7	6	9
VBS	1	7	23	21
Total Malicious Components	4853	8461	9329	8175
Unique Malicious Web sites	4237	7199	7859	5391

Malware delivery based on Google referer seems to be the most widely used approach by attackers as shown in Table 6.1. This is, probably, due to the popularity of the search engine. Therefore, to maximise the number of visits, cybercriminals will attempt to ensure that their malicious web sites or their corresponding doorway web sites are indexed and highly ranked within the search engine's result. A doorway page is the initial page which is served to a visiting user and allows a web server to examine the properties of a visiting user system and direct him to appropriate malicious or benign page. A large portion of malicious components - belonging to the 8,185 unique malicious web sites - are also located in the other different

sets (Figure 6.5).

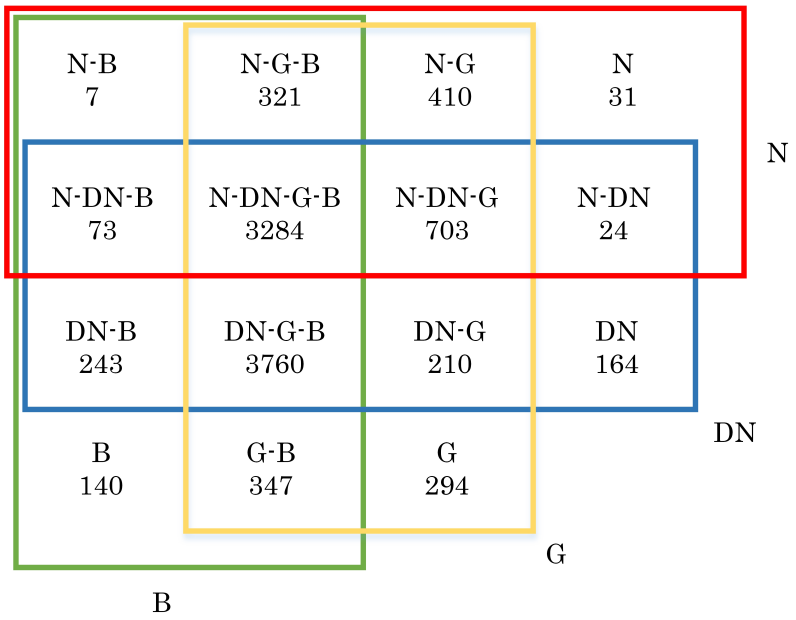


Figure 6.5: Number of malicious components common in null, domain name, Google and Bing referer settings

Based on the total number of malicious components, components unique and shared between sets, we derived a logistic regression model for repeated binary responses. Analysis of absolute numbers alone will not tell us reliably whether our independent variable (referer) really does have an effect. We therefore need a method that makes few assumptions about the distribution. Using a logistic regression model allows us to identify the relative impact of independent variables upon an outcome; in this case the impact of referer upon the likelihood of a page being malicious. We estimated the parameters in the model, and calculated the odds ratios and the corresponding 95% confidence interval (CI) using the Generalised Estimation Equation (GEE) [148] with the exchangeable working correlation structure (Table 6.2).

Table 6.2: Result of fitting a logistic regression model for repeated binary responses

Sets	Odds Ratio	95% CI for Odds Ratio
G/N	14.54	(13.40, 15.77)
DN/N	5.80	(5.42, 6.20)
B/N	4.73	(4.42, 5.07)
G/B	3.07	(2.81, 3.36)
G/DN	2.51	(2.29, 2.75)
DN/B	1.23	(1.15, 1.30)

```
We then conclude that:

The odds of "G" is 14.52 times the odds of "N".
The odds of "DN" is 5.80 times the odds of "N".
The odds of "B" is 4.73 times the odds of "N".
The odds of "G" is 3.07 times the odds of "B".
The odds of "G" is 2.51 times the odds of "DN".
The odds of "DN" is 1.23 times the odds of "B".
```

In summary, we determined that:

Odds of "G" > Odds of "DN" $\approx$ *Odds of "B" > Odds of "N"*

Hence,

Probability of "G" > Probability of "DN" $\approx$ *Probability of "B" > Probability of "N"*

This means that we are more likely to receive a malicious page from Google, equally from Domain name and Bing search engine and all greater than a null value referer. These results are in line with the design of many browser exploit kits which prevent access to landing pages if no referer or unique values are provided (Table 6.3) [149]. Landing pages are gateways which redirect users or point a browser to page which hosts the final

malicious content. The site with the final malware is called the Exploit Server [150].

Table 6.3: Styx exploit pack provides built-in APIs to block access to landing pages for traffic with unique or no referer value

/api/detBlockWithoutReferrer	Block access without referer
/api/setBlockUniqueReferrers	Block (first number of) access with unique referer

Our confidence in the accuracy of the results can be further supported by the number of malicious content generated by browser exploit kits and observed in different sets. We further analysed all datasets for signatures of three particular exploit tools; CrimeBoss, Blackhole and Neutrino browser exploit kits. CrimeBoss does not support traffic filtering based on referer header information and delivers malware regardless of the availability or specific referer field value. Blackhole and Neutrino on the other hand have built-in filtering and blacklisting capabilities for search engines and traffic with no referer header value. We observe that the number of CrimeBoss detected URLs is nearly identical in all datasets while there is a significant variation between Null-referer and other datasets for Blackhole and Neutrino exploit packs (Table 6.4).

Table 6.4: number of malicious components for Crimeboss versus Blackhole and Neutrino exploit kits

Exploit Kit	**Referer Datasets**			
	Null (N)	Domain Name (DN)	Google	Bing
Blackhole	254	371	370	328
Neutrino	70	143	138	128
CrimeBoss	18	19	20	18

6.1.6 Methods of Malware Delivery

Redirection is a popular technique used by attackers to target specific users in web spam and search engine optimisation attacks [151, 72, 73, 152] as well as avoiding detection by client honeypot crawlers [45, 153]. An attacker can retrieve referer information, browser type or operating system either at the client-side (e.g. JavaScript) or the server-side (e.g. referer information, user-agent) and redirect unsuspecting web site visitors to a compromised web site where an exploit targeting the vulnerability in the visitor's browser or operating system. Server-side redirections were logged in our experiments and analysis was performed to determine the ratio of redirections which resulted in malicious content delivery. Our findings illustrate that the vast majority of server-side redirections are designed to direct users from doorway pages to web sites that serve the actual malicious code. Figure 6.6 shows the number of server-side redirections for each referer and the percentage of those redirections that resulted in a malicious delivery. More than 80 percent of all server side redirections in our dataset using popular referer values, resulted in malware delivery while a null value representing a user who requests the malware directly resulted in only 33 percent successful malware retrieval.

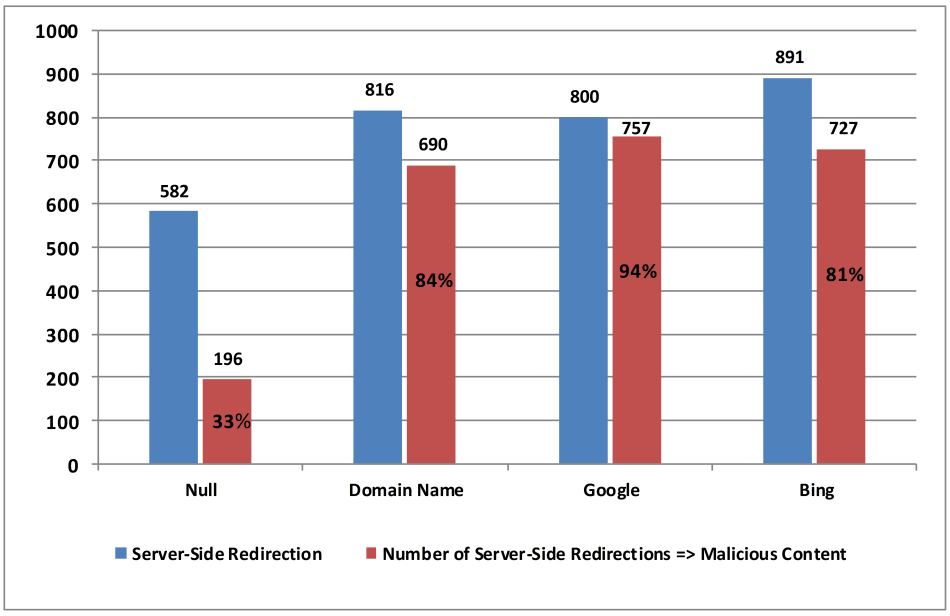


Figure 6.6: Number and ratio of redirections which resulted in malware delivery, for each referer

6.1.7 Analysis of Content and Status Codes

Our investigation shows that the behavior of malicious web sites is consistent across single domain. For instance, 1,612 malicious subdomains were detected within the 13900139**
0.com domain (e.g. http://1.1f93b.ke.13900139**0.com, http://1.1f93b.ke.13900139**0.com). These subdomains delivered malicious content only to visitors which had their referer variable set to Domain Name, Google and Bing values. Not a single case of malicious content within these subdomains was detected for Null referer. All malicious subdomains followed referer rules of their corresponding domain and showed consistent malware delivery behavior rule for similar referers. For instance, malicious contents were only detected in G, B referer settings for all subdomains of directxex.com and only DN, G and B for all subdomains within 13900139**0.com domain and no variations (e.g. DN, G for http://1.1f93b.

ke.13900139**0.com or G, B for http://1.1f93b.ke.13900139**0.com) were observed (Table 6.5).

Table 6.5: Most infected domains and their corresponding subdomains in our dataset, exhibiting similar malware delivery behavior

Domain	Sub-Domains	Referer Values
13900139**0.com	1612	DN, G, B
Clickbank.net	34	DN, G, B
crfebfj.com.cn	11	DN, G, B
directxex.com	8	G, B

The behaviour of a specific domain in terms of serving malware or content benign varied for a specific referer dataset. The number of malicious components within a specific domain (e.g. .js, .exe, .pdf) however, was consistent across different referers (Table 6.6). No malicious pdf document files were observed and a large percentage of malicious content were embedded JavaScript files.

Table 6.6: Examples of malicious domains serving exact number of malicious components to each referer

Referer Values	Parent Web site	Component
DN, G, B	http://herbu**e.pl	2 malicious components • (index.html) • swfobject.js
N, G, DN	http://dreameven**s.com.br	6 malicious components • scrollTo.js • jquery.validate.min.js • functions.js • jquery.anythingslider.min.js • jquery.min • jquery.metadata.js

Similar pattern of domain and subdomain naming were observed in most of datasets which suggest that these domains and subdomains are likely deployed by an attacker using an automated tool. Most browser exploit kits have unique URL naming patterns consisting of a combination of several integers and strings. These automated URLs can then be inserted into legitimate web site through iframes or used as gateway or landing pages. Such naming patterns allow detection engines such as Yara which rely on pattern matching techniques to find variations of such combinations [131, 154, 155]. Yara is one of the detection engines used by YALIH client honeypot in this research and utilises regular expression rules to identify such embedded malicious links [150]. The pattern in domain naming of exploit kits allows us to develop regular expression rules to detect them (Table 6.7).

Table 6.7: Example of similar domains following exact malware delivery rule for each referer

Similar Domains	Referer
• 12280.yu.13900139**0.com • 137be.oo.13900139**0.com • 15dd1.sl.13900139**0.com	DN, G, B
• w6b1ca**ube.info • w6b2ca**ube.info • w6b3ca**ube.info	DN, G, B
• Allfiles**0.com • Allfiles**1.com • Allfiles**2.com • Allfiles**6.com	DN, G, B

CrimePack exploit kit for instance, creates gateways using specific naming schemes:

- */0day.php
- */cb.php
- *.php?action=jv&h=*
- */rebots.php*

Examples of a few landing pages or URLs containing the payloads are provided below:

- */phedex/* - http://domain.com/phedex/java7.jar?r=8473618
- */java7.jar?r=* - http://domain.com/jex/java7.jar?r=7529012
- */jmx.jar?r=* - http://domain.com/app/jmx.jar?r=2265917

Such malicious content can be detected by a combination of the following Yara rules using our low interaction client honeypot (Figure 6.7). The rule looks for both the gateway and landing pages patterns which contain the above keywords and variations of seven-digit identifier number.

```
rule Crimeboss
{
  strings:
$match1 = /\/(\.php\?action=jv&h=|java7\.jar\?r=| jmx\.jar |
    phedex\/|jex\/|jhan\.jar|m11\.jar|)/
$match2 = /\/[a-z0-9]{3,4}\.jar\?r\=\d{7}\s/
  condition:
    $match1 and $match2
}
```

Figure 6.7: Regular expression rules by Yara to detect naming variations of CrimeBoss exploit kit's links

The behaviour of malicious web sites and error messages were returned varied between referer values. For instance, the malicious web site http://5thaveappliance.com returned an "error 111 – Connection Refused" message for null referer value, while it returned JS-Trojan/Exploit for DN, G and B referer values. Web sites which returned "error 111 – connection refused", and "HTTP Error 403: Forbidden" for a component within

a web site (e.g. .js, .exe) were of interest, to observe if any of these error messages returned for a particular referer would result in different messages or potentially malicious content for other referer settings. Web sites which refused connection for a component had a high rate of malicious content (Table 6.8).

Table 6.8: Number of malicious components for domains refusing connection and forbidden error messages.

Referer	**Error Messages**	
	Connection Refused / Malicious	***Forbidden / Malicious***
N	13 / 10	34 / 6
DN	16 / 8	37 / 17
G	16 / 15	15 / 14
B	12 / 7	15 / 13

6.1.8 Lessons Learnt

Cloaking poses a challenge to detection of malicious web sites as it requires additional copies to be fetched that match the required setting by the exploit kit. Referer is an optional HTTP field and can be used for cloaking purposes. For example, referers are widely used in web spam and SEO practices to deliver spam or malicious content to selected web site visitors and avoid detection by search engine crawlers. Previous studies on HTTP referer variable have focused on web spam and cross-site scripting attacks and no prior measurement study on the importance of referer variable on the type and behaviour of malicious domains has been done.

We observed that web sites which refused or denied connection for a component within the web site exhibited a high potential of being malicious in all datasets. There is also a strong connection between the maliciousness of a subdomain and presence of other subdomains within that particular domain. These patterns can be used as strong indication of mali-

ciousness of a web site and used in conjunction with other attributes in the creation of signature and machine learning detection signatures and engines. Machine learning algorithms for instance, use various extracted attributes from the URL, domain, content or behaviour of a web server (.e.g. "eval" function, hidden or size 0 iframes) to classify a web site [140, 156]. Moreover, identification of such behaviours can provide a filter by which a URL is marked for further analysis or multiple retrievals with various referer or client-side settings (e.g. browser type, version, location). Applying such filters can in turn reduce the required in-depth analysis and resources –through multiple retrievals– for a large URL dataset.

Overall, our findings illustrated that web servers actively scan, log and utilise referer information in the delivery of malicious payload to end users. The practice of removing referer from HTTP header by intermediary devices might restrict access to few web sites but reduce the risk of malware infection to the average users. We believe such experiments are regularly required to explore popular online services and identify the optimal referer values which could trigger malware delivery by the malicious web sites. Identifying such variables helps reduce the cost of detection and malware collection by those involved in malware collection and analysis through minimizing the number of required retrievals.

6.2 Measurement of IP and Network Tracking

Web sites collect a wealth of information about visiting users such as their IP address, network provider, web browser and operating system configuration information. This information is gathered from the user's IP address, HTTP header values and from code embedded within the web site that is downloaded and executed within the browser. Geolocation services such as MaxMind [157] can be used to further infer user location, spoken language, economic position and cultural information from the visiting user's IP address. Web sites may also monitor the activities of a particular visitor over time by using the visitor's IP address. For example, the pattern of requests by a visitor can be recorded simply by logging IP addresses with the timestamp of the visit and record of what content was delivered at that time. Monitoring what and how content is retrieved, a web site can associate a behaviour to a certain IP address. This information can be used to decide what content to provide to the visitor on future visits. The process by which a web site monitors the behaviour of visiting users using their IP addresses is known as IP tracking. IP tracking is the initial reconnaissance process which allows the web site to provide or deny visitors specific services or content.

It is known that IP tracking is supported by various Crimeware kits (e.g. Blackhole exploit kit) for determining what malware to deliver to web site visitors, however there has been few empirical research into the extent. Additionally, it is also believed that these kits can be configured to track visitors not just at the individual level but at the level of an organisation identified by IP subnet (we term this Network tracking). These would be used to ensure that any visitor from say an anti-virus organisation or a visitor with suspicious activity is identified as potential bots crawling the web to find malware. The contribution of this experiment is to empirically investigate the behaviour of web sites in terms of malware delivery based on the experiments where we deliberately control the visitation pattern

and number of retrievals by web crawlers collecting malware from web sites. We analyse the data collected to determine the extent to which IP tracking and Network tracking is actually used.

The process by which specific content is delivered as a result of prior IP tracking activity is known as IP cloaking. The process is not necessarily illicit in nature as it is widely used by benign web sites to deliver specific content to users based on their IP and corresponding information gathered from a user. For example, a 2013 inquiry by the Australian Government House of Representatives Standing Committee on Infrastructure and Communications, for example, found that the 'geo-blocking' practices (i.e. methods adopted by businesses to differentiate between regions and keep customers separate) resulted in Australian consumers and businesses "pay[ing] much more for their IT products than their counterparts in comparable economies". In many cases Australians pay 50 to 100 percent more for the same product [158].

IP Cloaking is a defensive approach used by malicious web sites to lower the risk of detection. The lower risk of detection is a direct result of anti-forensic practice in which a malicious web site subjects its code to a limited and few selected visitors. The ideal targets of malicious web sites are content-requesting IP addresses which do not present any suspicious activity and possess the required environment in terms of operating system and browser vulnerabilities. The suspicious activity and profile presented by an IP address can be in the form of 1) Suspicious information about the owner of the IP address 2) The pattern of interaction with the web site. The primary challenge is distinguishing average users from automated security tools.

Information about an IP address may be associated with detection tools or point to a security organisation. In such a scenario, the web site can mitigate detection by serving benign content to the requesting client. Association of IP addresses with security organisations is performed through matching the IP address against a database of known and active secu-

rity companies. Such databases and services are freely and commercially available online. Such a defensive approach requires few resources to implement and is utilised by many Crimeware and browser exploit kits (e.g. Red Kit, Eleanore, CrimePack, Blackhole exploit kit) [126, 159]. Effectiveness of such a simple mitigation strategy has contributed greatly to its popularity with malicious web sites. According to a Rajab *et. al.*[160]:

"IP cloaking contributes significantly to the overall number of malicious web sites found by our system".

Malware delivery may also vary based on pattern of visit by visiting users. Malware delivery changes depending on the history of visits by users identified by their IP addresses. Generally a malware is delivered to a client if it detects an IP address retrieving content for the first time. This is achieved by keeping a list of IP addresses accessing the content. A subsequent visit is matched against the list and results in redirection to a benign web site. This is to avoid detection by security tools which tend to access a malware delivery web sites on regular basis to either determine if it is still malicious or gather different and up-to-date version of malware served by the web site [84].

There are various approaches security vendors use to detect malicious web sites using IP tracking and IP cloaking evasion techniques. Some of these counter-responses may include: varying IP ranges, varying time intervals between each visit to avoid giving the impression of a crawler and introducing randomisation into the visiting algorithm [160, 84]. We perform our experiment using varying IP addresses and retrieval time.

Similar to IP tracking, tracking network and cloaking is the process of monitoring the behaviour and profile of visiting users' IP addresses by a web site and serving different or specific content accordingly. Network tracking and cloaking allows malicious domains to mitigate detection through tracking and serving benign content to IP addresses sus-

pected of belonging to known security or research firms [160]. A malicious web site could also potentially block malware delivery to all IP addresses belonging to a certain subnet with suspicious activity (i.e. crawlers) or on suspicion of hosting client honeypots or detection systems [150]. Network banning and cloaking feature has been observed in multiple browser exploit kits [126] and can be easily implemented by malicious web sites.

The primary purpose of network and subnet tracking is detection mitigation. Magnitude Exploit Kit for instance keeps a database of 1400 IP range records of high profile companies, security and law agencies [125]. The inclusion of range of IP addresses and subnets allows for easier updatability compared to single unique IP address lists. Our experiment in network tracking does not include analysis of attacks on high profile subnets as access to an IP address in a large security organisation was not available. Our experiment however focused on determining if suspicious activity by an IP address would result in varying web site behaviour toward adjacent IP addresses which most likely belonged to the same network.

6.2.1 Background - IP and Network Tracking

IP and network tracking have been mentioned by many researchers as a technique employed by malicious web sites to mitigate detection by security tools, organisations and search engine companies. These studies investigated IP and network tracking using IP addresses and network subnets of high profile security and search engine companies [80, 81, 82]. O'Dea [80] studied the behaviour of rogue malware disguised as antimalware software and their evolution. Target selection and malware delivery relied on monitoring a visiting user's browser, referer and IP address. Visitors with a Microsoft IP address for instance were redirected to a benign web site. IP tracking can also be performed at an individual visitor level in which, only the behaviour and not the public profile of the visiting user

is monitored for suspicious activity and succeeding visits.

There are few scholarly studies on the prevalence of IP and network tracking being used at an individual visitor level. Erete [81] proposed and designed a distributed honeyclient which uses idle processing cycles of participating users' desktops to retrieve and detect malicious web sites. Allowing average users to participate in malware retrieval and analysis enables such a design to easily mitigate IP cloaking techniques as varying IP addresses are used for each retrieval. Their system is in principal similar to honey@home server honeypot which is deployed for detection of server-side attacks [89]. Their small-scale study on 35,000 URLs revealed a minority of detected malicious subset (i.e. 1294/33) utilised IP tracking infrequently. Stringhini *et. al.* [82] proposed a system relying on analysis of the chain of redirections by malicious domains rather than static or dynamic analysis of the URL content. Redirections point average visitors to the final domain that hosts the appropriate malware for the visiting user's system and browser environment. Their system similarly uses the log files of browser-to-web server interaction of a large number of participating users. Their proposed system is capable of detecting IP and network cloaked web site but relies on several distinct redirections to achieve accurate detection.

Yi-Min *et. al.* [30] and Rajab *et. al.* [83] performed a large scale study of malicious web sites and investigated several aspects including IP tracking behaviour over a long period. Their IP tracking experiments however focused on IP and network profile tracking of malicious web sites in which IP addresses from subnets belonging to well-known security or search engine companies are monitored and served with benign content. Our experiment on the other hand, focuses on the relationship between malware delivery and the history and pattern of visits by single or adjacent nodes in a network identified by their IP address. A study by Zeeuwen *et. al.* [84] on update patterns of malware download centres (including IP tracking), and their relative prevalence focused on designing a re-scheduling algo-

rithm to effectively download malicious binaries from malware download centres. Their study in 2010-2011 periods did not result in conclusive evidence of IP tracking on a small set of 5000 malicious web sites. Their research is similar to ours in terms of monitoring IP tracking and blacklisting behaviour, however on a very small set of URLs and different experimental setup, properties and visitation pattern and does not cover network tracking.

6.2.2 Experiment Setup – IP Tracking

This section explains the setup and retrieval process for IP tracking experiment.

Location:

Three nodes located within a small geographical location (i.e. City). Geographically located nodes were a measure to ensure nodes would not be served varying content based on their location.

Network Profile and Input URL Dataset:

Nodes were selected from low profile VPS services which had no affiliation with research or security organisation (Table 6.9).

Table 6.9: IP, network and locational properties of the retrieving nodes.

IP Tracking Experiment			
	Nodes		
	IP1	**IP2**	**IP3**
IP Address	107.161.157.***	199.48.68.***	155.94.216.***
Network profile	VPS Nodes	PaulVPS	Sshvm
Physical Location	Los Angeles, USA		

To maximise the external validity of the results, an input set of 1,000,000 suspicious URLs were gathered from multiple sources (e.g. hpHosts [161]). High external validity in an experiment warrants a high degree of certainty about the generalisability of the findings.

Retrieval process:

Node-1 (IP1) in the experiment's setup initiates three retrievals to malicious web sites:

1. An initial connection to potentially register its IP address.
2. A second retrieval in conjunction with IP2.
3. Third retrieval in conjunction with nodes IP2 and IP3. The third retrieval is to examine the impact of multiple visits on malware delivery of malicious web sites.

The sequence of actions based on three nodes and multiple retrieval process of web sites is depicted in figure 6.8. The twelve hours delay between retrievals was selected to minimise the risk of malicious web sites going offline yet allow the IP addresses of visiting nodes to be registered with the web sites for the next retrieval. Malicious web sites utilizing fast flux networks for instance, have been observed to have an active life of few days and no more than 18.5 days [162, 163].

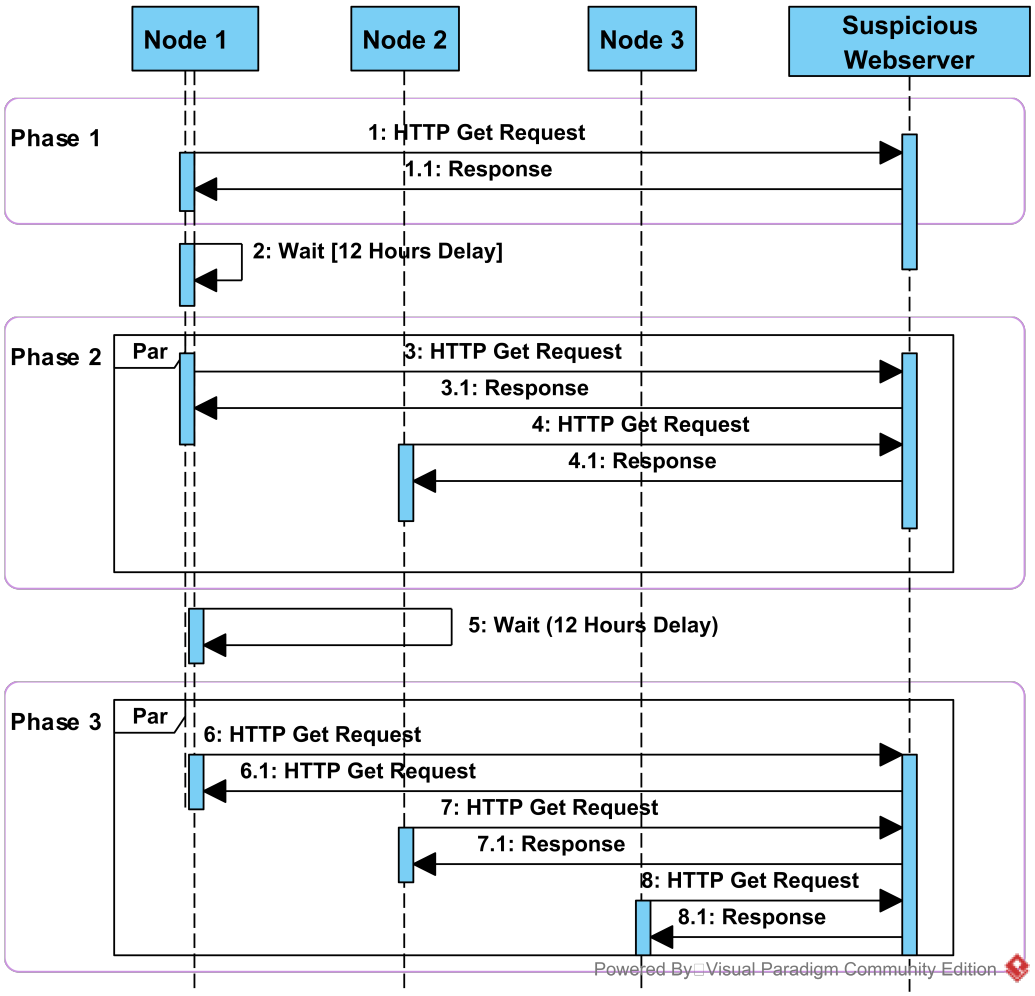


Figure 6.8: Retrieval process of our experiment based on three nodes and three phases

Crawling behaviour was avoided in IP tracking experiment to minimise the possibility of result contamination which may arise from a web site's detection of crawling activity. Cookies were handled and stored for subsequent visits by our retriever component (i.e. Mechanize simulated browser and its built-in *"cookielib library"* [127]) to inform a web site of previous visits.

6.2.3 Results - IP Tracking

In our repeated measurement study, one million web sites were retrieved using our low interaction client honeypot YALIH on three different nodes. Initial retrieval by hosts show a near similar number of malicious web sites fetched; IP1=4969, IP2=4961 and IP3=4959. The minor difference between initial retrieval by nodes could be associated with the time difference between retrievals which would result in some web sites going offline (i.e. 24 hours between all nodes). Malicious web sites utilizing fast flux networks for instance, have been observed to have an active life of few days and no more than 18.5 days [162, 163]. Table 6.10 shows the number of unique malicious web sites fetched in retrieval by each node. The delay between each retrieval phase is 12 hours. Second retrieval by nodes shows a relatively large drop in the number of unique malicious web sites compared to the initial retrievals. Initial retrievals have an overall standard deviation of 5.29 (10 web sites between retrieval 1 by IP1 and IP3) while the standard deviation between first, second and third retrievals is > 224 for IP1 (Table 6.11).

The similarities between the numbers of detected malicious domains in phase 1 of retrieval by all nodes and phase two by IP1 and IP2 illustrate the reliability of our study. It confirms that if hazards can be controlled and minimised in an experimental environment, the results should be consistent and can be replicated and predicted. Nodes participating in the study with similar attributes and controlled hazards -apart from IP address- resulting in large decrease in the number of detected attacks also support the internal validity of our experimental design where a direct relationship between number of visits and malware delivery behaviour can be observed. Neglecting to control nodes' other attributes or experimental variables could introduce significant hazard into the study. For instance, in this study, all nodes' referer settings were set to Google's search query to represent a user who visits the web site from Google's result page. If the referer setting of IP1's retrieval phase 1 were to change to blank/null and

retrieval phase 2 to Google's referer setting, it would result in significantly higher detected attacks in phase 2. Google's referer has been confirmed by studies to mimic a regular user more accurately and result in higher number of attacks and confirmed by our experiment.[73, 74, 75, 76].

Overall, 4969 unique malicious web sites were detected in phase 1 retrieval by IP1. Initial retrieval by other nodes (i.e. IP2 and IP3) also resulted in very similar number of detected domains (i.e. 4961 and 4959) despite 24 hours total delay. Second retrieval phases by IP1 and IP2 nodes followed similar pattern of comparable number of detected domains (i.e. 4867 and 4863) (Table 6.10). Evidence of IP tracking behaviour can be observed in subsequent retrievals by nodes IP1 and IP2. Retrieval phase 2 and phase 3 by node IP1 and phase 2 by node IP2 for instance results in relatively much lower number of detected domains. The results illustrate that in our dataset, malicious domains were more likely to served malicious content to nodes with no history of prior interaction [164].

Table 6.10: Number of malicious web sites fetched in each phase by each node

Retrieval Phase	**Retrieval Nodes (Number of malicious web sites)**		
Phases	IP1	IP2	IP3
	(Phase 1) 4969	[12 hour delay]	[12 hour delay]
	(Phase 2) 4867	(Phase 1) 4961	[12 hour delay]
	(Phase 3) 4539	(Phase 2) 4863	(Phase 1) 4959

Table 6.11: Statistics on IP1 and IP2 datasets

Descriptive Statistics							
	N	**Min**	**Max**	**Mean**		**Std. Dev.**	**Variance**
	Statistics	Stat.	Stat.	Stat.	Std. Error	Stat.	Stat.
IP1	3	4539	4969	4791.6	129.71	224.6	50481
IP2	2	4863	4961	4912.0	49.000	69.29	4802.0

The decline in the succeeding retrieval by nodes in Table 6.12 and 6.13 illustrate that many web sites rely on initial visit for malware delivery but subsequent visits also trigger IP tracking and cloaking behaviour. 4068 domains were shared between phase 1, phase 2 and phase 3 retrievals of node IP1. A large set of domains (i.e. 270) only delivered malicious code during retrieval phase 1 while 130 domains triggered malware delivery only in phase 2 of retrieval. Phase 3 resulted in only 23 unique web sites being detected.

Table 6.12: Number of common and unique malicious domains for each retrieval phase (IP1 and IP2)

IP1				
Retrieval Phases	**Common Domains**	**1^{st} retrieval only***	**2^{nd} retrieval only**	**3^{rd} retrieval only**
1,2,3	4068	270	130	23

*This table excludes common domains between various phases.

IP2			
Retrieval Phases	**Common Domains**	**1^{st} retrieval only**	**2^{nd} retrieval only**
1,2	4515	446	348 (2^{nd})

A large number of malicious web sites served identical type of malicious content not only for different retrieval phases of a single node but also between different nodes. For instance, in table 6, initial retrieval by IP1 and IP2 resulted in 4417 common malicious domains being detected. From which only 18 domains served varying malicious content (i.e. type of malicious code: malicious iframe, shell-code etc.). Domains serving varying malicious content for different nodes were a small subset of overall detected malicious domains (e.g. 25/4457 and 26/4354 for phase 1 retrieval

of IP1-IP3 and IP2-IP3 respectively) (Table 6.13). There are two possibilities for such behaviour: 1) These domains directed users to their own specific Exploit Server which served a single exploit for all visitors or, 2) Our client honeypots managed to emulate regular users effectively and had consistency in their emulation across all three nodes which triggered identical malware delivery. We believe the latter assumption is valid since various attributes of a visitor browser and system are analysed by the landing page before redirecting to an exploit server which delivers a specific malware for a visitor's system configuration. Landing pages are gateways which redirect users or point a browser to page which hosts the final malicious content. The site with the final malware is called the Exploit Server [150].

Table 6.13: Number of common malicious domains between retrieval phases and those serving varying content upon consecutive visits

Node	Retrieval Phase	Common Domains	Web sites Serving Exact Malicious Content/Varying Content
IP1-IP2	1	4417	4399/18
IP1-IP3	1	4482	4457/25
IP2-IP3	1	4380	4354/26
IP1-IP2	2	4291	4273/18
IP1	1-2	4486	4459/27

6.2.4 Experiment Setup – Network Tracking

This section explains the setup and retrieval process for network tracking experiment. In this experiment, similar to IP tracking, the idea was to determine if retrieval from nodes within the same subnet or adjacent IP addresses (e.g. 104.168.32.**2, 104.168.32.**3) would result in lower number of attacks for the node initiating the 2^{nd} connection (i.e. 104.168.32.**3). A client residing on a different but similarly geolocated network (i.e. 162.221.176.*7) would be used as a controlled host.

Location:

Three nodes located within a small geographical location (i.e. City). Geographically located nodes were a measure to ensure nodes would not be served varying content based on their location.

Network Profile and Input URL Dataset:

Nodes were selected from low profile VPS services which had no affiliation with research or security organisation.

Table 6.14: Properties of the participating nodes

Network Tracking Experiment		
	NT1 Subnet	**NT2 Subnet**
IP Address	**NT1-2** = 104.168.32.**2 **NT1-3** = 104.168.32.**3	**NT2-7** = 162.221.176.*7
Network profile	OpenVirtuals	VPSAce

An input set of 500,000 suspicious URLs were gathered from multiple sources including public database of suspicious domains (e.g. hpHosts [161]). High external validity in an experiment warrants a high degree of certainty about the generalisability of the findings.

Retrieval process:

Several steps were taken to represent the behaviour of initial retrieval by node NT1-2 as a crawling activity:

- Crawling activity was enabled for NT1-2 through removal of delay between the retrieval of input domain components (e.g. .exe files)
- Performing rapid retrieval of random links from the domain.

- Accessing and retrieving Robot.txt file to officially represent the node as a crawler.

These actions were performed to make the initial retrieval from the first node (i.e. NT1-2) suspicious to a web site. This experiment only focused on analysis of malicious web site behaviour toward adjacent IP addresses suspected of being from the same web site. The experiment did not include retrieval from high profile security or research organisations. In our experiment, similar number of malicious web sites in NT1-2 and NT1-3 and lower number of malicious web sites in NT2-7 – which is performed simultaneously with NT1-3 – would indicate network tracking by the web site. Figure 6.9 shows an overview of our experiment setup and the sequence of retrieval process between the participating clients and our input dataset of suspicious web sites.

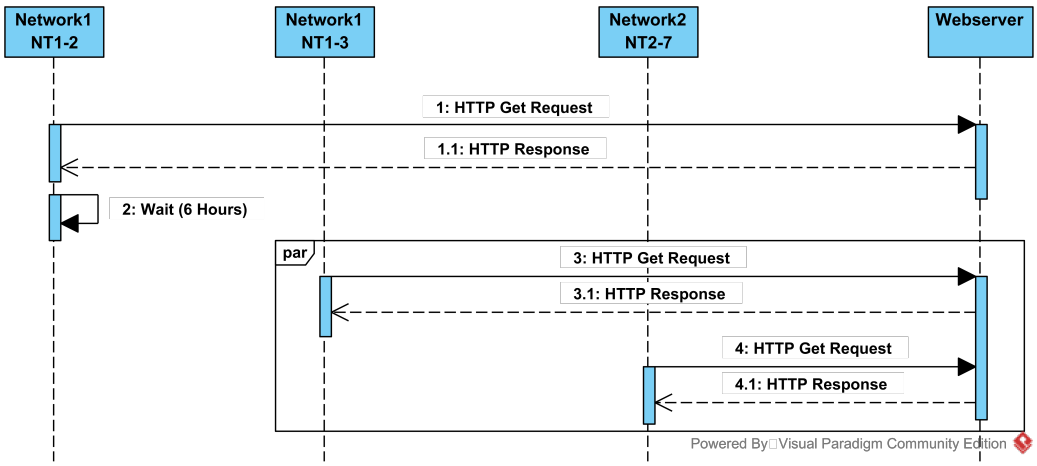


Figure 6.9: Simple view of our network tracking experiment's setup

6.2.5 Results - Network Tracking

Retrievals between network NT1-2, NT1-3 and node NT2-7 in the different network indicate a minor difference which could potentially be due to

timing of the retrieval (i.e. 6 hours). Retrieval 2 from network NT1 which had a similar IP address in the same subnet however resulted in slight difference in the number of unique domains and embedded components compared to the initial retrievals from NT1-2 and NT2-7. This shows that within our experiment and dataset containing more than 500,000 unique malicious domains, network tracking was employed by a very small subset of the domains. This is significantly smaller than deviation observed in IP tracking. A reason behind such a result could be due to the fact that tracking or blocking a subnet may result in blocking a large subset of IP addresses which would subsequently result in less number of potential infected hosts. We therefore assume that while network tracking for high profile networks (e.g. Google, Microsoft and Symantec Security) is widely used, implementation of network tracking based on one-time suspicious behaviour from a node is not commonly used.

Table 6.15 shows a summary of detected attacks in our experiment of network tracking performed on 500,000 URL dataset. The number of unique domains and components is very similar in all phases of the experiment. 1041 and 1037 unique domains were detected for NT1-2 and subsequent retrieval by NT1-3 respectively. These results are very similar to the number of detected domains for NT2 (i.e. 1029) which retrieved the content in concurrency with NT1-3. These results illustrate that remote suspicious web sites treated NT1 nodes as unique independent nodes by serving very similar content, confirming our hypothesis that network tracking and cloaking was not used in our input dataset.

Table 6.15: Number of detected domains in network tracking experiment

Network Tracking Dataset		
	NT1	**NT2**
Unique Domains/Components, NT1-2	1041/1428	-
Unique Domains/Components, NT1-3	1037/1421	1029/1423
Common Domains between NT1-2/ NT1-3	1011	
Common Domains between all	985	
Common Component between all	1352	

There are however differences in actual domain names in common and their behaviours. Thirty malicious domains were unique to NT1-2 of which few examples of the difference in their behaviours compared to NT1-3 is listed in Table 6.16. Similarly 26 unique domains unique to NT1-3 mostly resulted in connection time-out messages on NT1-2 while serving malicious components to NT1-3. Similar behaviour was observed between all retrieval nodes and phases. The types of delivered malware between all common components of three phases (i.e. 1036 between NT1-2, NT1-3 and NT2-7) were identical (e.g. executable file, malicious iframe etc.).

Table 6.16: Variation in the behaviour of malicious web sites between two retrieving nodes in network N1

Website	**N1-2**	**N1-3**
eiqikan.com	JS.Agent-156/JS/Downloader.Agent	timed out
eastfilm.net	Exploit.Iframe-1/HTML/Framer	timed out
caomeimiao.com.cn	HTML.Iframe-63/HTML/Framer	timed out
gulsproductions.com	JS.Trojan.Blackhole-1/HTML/Framer	timed out
xpath.syncrvprodist.com	Adware AdLoad.K/Adware Generic_r.TH/Adware Generic5.BNDB	timed out
feenode.net	Exploit.MS05-013/HTML/Framer.AG	timed out
www.ahwydljt.com	Heuristic.HTML.Dropper/VBS/Dropper	Failure in name resolution
yantushi.cn	JS.Agent-156/JS/Downloader.Agent	timed out
gnaa-members.on.nimp.org	Exploit.MS05-013/HTML/Framer.AG	timed out
carvingstudio935.com	HTML/Framer	Forbidden
turism-portal.ro	Exploit.HTML.iframe-6	timed out
chinese.ahzh-pv.com	Heuristic.HTML.Dropper/VBS/Dropper	timed out
yz-sl.cn	VBS.Agent-62	timed out
thedeadpit.com	HTML/Framer	Redirection
indochinanetwork.com	HTML/Framer	timed out
qshdch.com	HTML/Framer	timed out
combidata.internetdsl.pl	HTML/Framer	Not found 404
hentelpower.com	VBS/Dropper	timed out
down.hit020.com	Win.Trojan.11350378	Forbidden
beroepsperformancescan.nl	JS/Redir	timed out
china-zhenao.com	VBS/Dropper	timed out
dtstesting.com	JS/Exploit	timed out
xn--80aaagge2acs2agf3bgi.xn--p1ai	HTML/Framer	timed out
zjkxlzxw.com	HTML.Iframe-63	timed out
laspfxb.com	VBS/Dropper	timed out
centralwestwater.com.au	HTML/Framer	timed out
web1.51.la	JS/Agent	Connection reset by peer
5.charliemask.bugs3.com	HTML/Framer	Bad Gateway
e-cte.cn	VBS/Dropper	timed out

Based on our understanding from our measurement experiment we believe simple network tracking for an average user may not pose a threat or vary considerably. Simple network tracking implies a tracking activity to identify. This experiment was however performed using VPS services from low profile companies who are not engaged in security related ac-

tivities. To accurately measure the behaviour of malicious web sites, an experiment to compare the behaviour between hosts located at a low profile subnet with a host belonging to a large recognised organisation (i.e. McAfee, Google) should be conducted.

6.3 Language Tracking

The language preference of a user, set either within the browser, operating system or user agent, may be used by companies to provide language and region specific services for end users. Although language alone, due to diversity of communities, cannot reliably determine a user's location, in combination with other factors such as IP, network address and so on, a near accurate estimate of user location can be determined. Identifying a user's language through browser and HTTP header is a simple process which requires parsing the transmitted HTTP values by an end user's browser for specific strings (e.g. ru-RU, en-US) as discussed in chapter 6, section 3.3. The intention of our experiment was to measure the effect of user language on malware and malicious code delivery and validate whether user language settings taken in isolation, triggers malware delivery regardless of other factors.

6.3.1 Experiment Setup – Language Tracking

This section explains the setup and retrieval process for language tracking experiment.

Location:

Three nodes located within a small geographical location (i.e. City). Geographically located nodes were a measure to ensure nodes would not be served varying content based on their location.

Network Profile and Input URL Dataset:

Nodes were selected from low profile VPS services which had no affiliation with research or security organisation.

The user language setting in our experiment included the language

of the browser set within the user-agent string. Following the HAZOP approach, the experiment was conducted so that all other variables were controlled and multiple copies of a web site were retrieved from similar environments and locations using exact nodes and varying language settings. A set of URLs containing one million web sites from Russian top level domain was input for retrieval. It was assumed that these web sites had been designed for Russian language users 6.10.

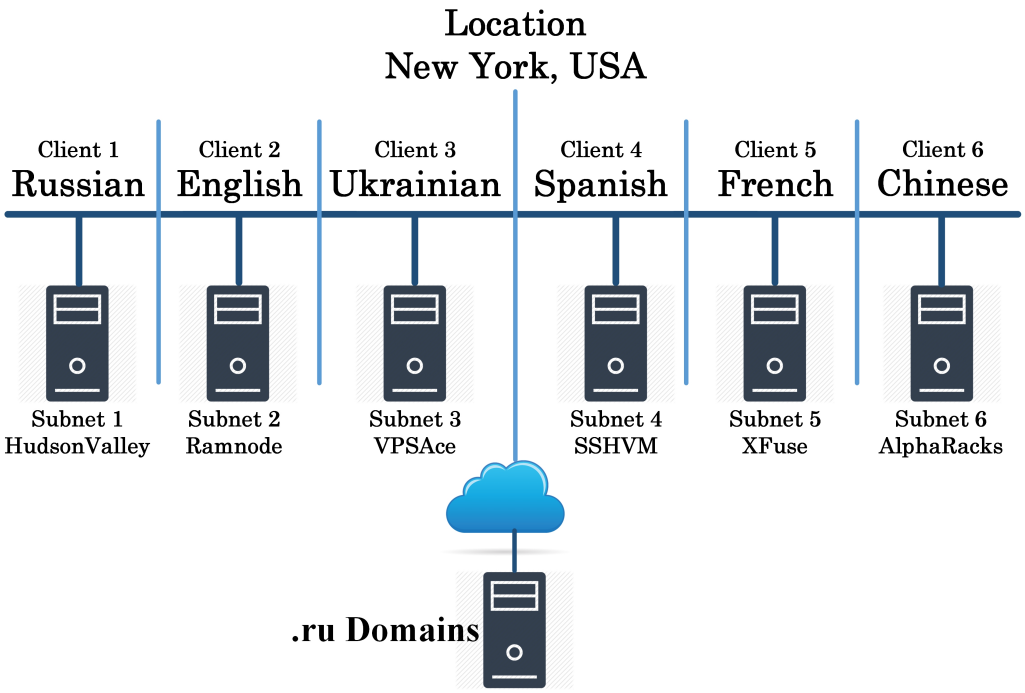


Figure 6.10: Simple view of our language experiment's setup

Table 6.17 shows the result of our experiment in which language settings set for a specific top level domain resulted in significantly higher number of detected web sites. Figure 6.11 shows a six set Venn diagram corresponding to six languages evaluated in this experiment. Each region (dataset) of the diagram is assigned a unique binary representation corresponding to union and intersection between each region and populated with the number of attacks detected for that specific region. The Venn di-

agram overall shows the number of attacks unique to each dataset and shared between others. This representation is then used to calculate confidence interval and odds using the Generalized Estimation Equation (GEE). For instance:

- 010000 represents English language for which no attacks were detected unique to that region, hence 0.
- Four attacks were unique to Russian language represented by 100000 which were not observed in any other regions.
- 110000 represents the dataset resulted from the union of English language dataset (010000) and Russian language dataset (100000). There were 28 attacks unique to a union of these datasets which were not observed in any other regions/datasets.

We calculated the odds ratios and the corresponding 95% confidence interval (CI) using the Generalised Estimation Equation (GEE) with the exchangeable working correlation structure for individual Accept-Language values. We derived the logistic regression model for repeated binary responses using components unique and shared between sets. Based on our findings we conclude that, a node located in Russia retrieving a web site from .ru domain with language preferences set to Russian is on average 14.2 times more likely to be targeted versus a node with fr-FR, UK, zh-CH and es-ES language preferences. Data also shows that English language settings is on average 2 times more likely to subject a user to an attack than fr-FR, UK, zh-CH and es-ES language preferences.

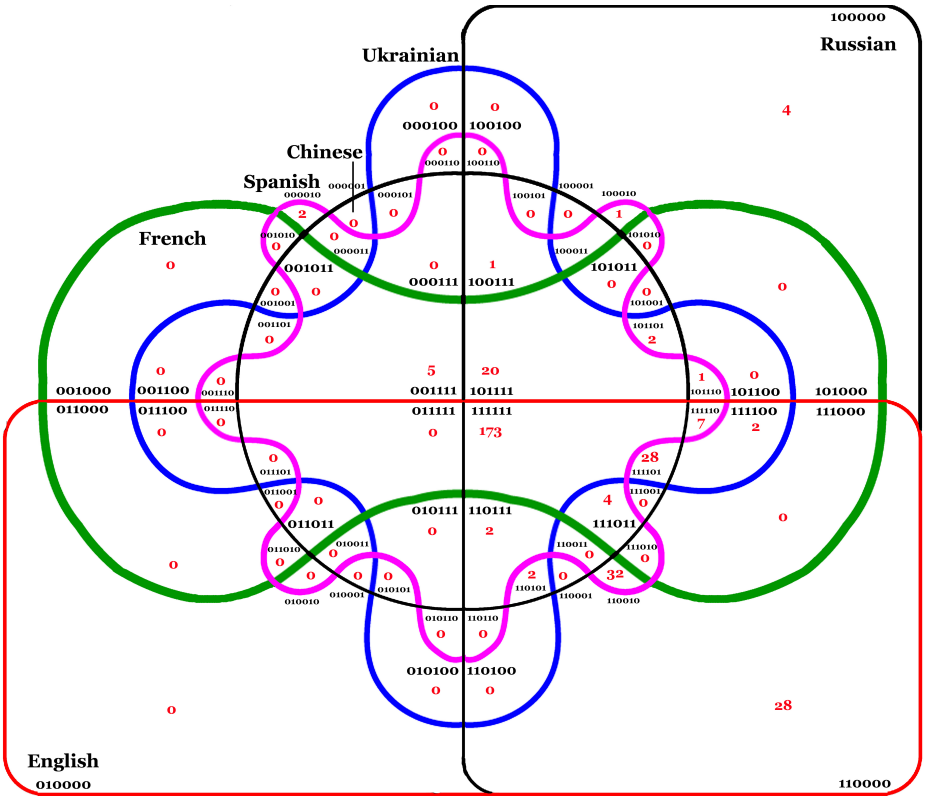


Figure 6.11: Venn diagram representing unique and shared malicious components between datasets

In summary, we determined that:

Odds of "RU" > Odds of "EN" > odds of FR ≈ Odds of "UK" ≈ Odds of ZH ≈ Odds of ES

Hence,

Probability of "RU" > Probability of "EN" > Probability of "ZH" ≈ Probability of "UK" ≈ Probability of "FR" ≈ Probability of "ES"

Table 6.17: Probability of malware delivery based on browser language preference

Sets	Odds Ratio	95% CI for Odds Ratio	
RU/ZH	14.2	6.5	31.4
RU/FR	13.0	5.9	28.7
RU/UK	12.8	5.8	28.2
RU/ES	11.7	5.2	26.4
RU/EN	5.7	2.9	11.2
EN/CH	2.5	1.6	3.9
EN/FR	2.3	1.5	3.6
EN/UK	2.3	1.5	3.5
EN/ES	2.1	1.3	3.2
ES/CH	1.2	0.9	1.7
ES/FR	1.1	0.8	1.5
UK/ZH	1.1	1.0	1.3
FR/ZH	1.1	1.0	1.2
FR/UK	1.0	0.9	1.1

6.4 IP Geolocation

Malware delivery based on geographical location, either through the IP address or other factors, is a complex scenario dependent on various social, economic or political situations. Social and economic factors contributing to targeted attacks have been discussed in previous chapters. An attack campaign using political or social events can only be detected with constant monitoring of the malicious domains. There are cases in which publicised events such as world cups, valentine days and Brexit referendum have been used to lure users into an attack. Luring techniques based on social or political events are widely used in spam and phishing campaigns [170, 171, 172].

The following sections provide results of a real-world experiment to verify the presence and prevalence of IP geolocation attacks. We collect and analyse data gathered from multiple nodes in various geographical locations.

6.4.1 Experiment Setup – IP Geolocation

This section explains the setup and retrieval process for IP geolocation experiment.

Location:

In this experiments, we focused on a snapshot of a large set of URLs, including multiple regions and various language settings. Accordingly, two languages, two domains and six regions were selected. Table 6.18 below shows the list of retrieval agents located in several countries, used in the experiment setup.

Table 6.18: list of countries used for retrieval in the experiment setup

Countries	Countries and Groups of Participating Nodes					
	Group A		**Group B**		**Group C**	
	Russia	**Ukraine**	US	France	Hong Kong	Chile
	Group A		**Group B**		**Group C**	
	Spain	**Chile**	US	France	Hong Kong	Russia

Two countries had common social and cultural similarities, including but not limited to identical official languages and a large minority of each within their population. The purpose of the experiment was to determine if web attacks hosted on each top level domain (i.e. .ru and .es) URLs, targeting their associated country's population, were also more likely to be observed in the other country within the same group (i.e. Ukraine and Chile). The results would subsequently be compared to countries which shared fewer social or cultural similarities. Group A consisted of the following countries which shared social and cultural similarities:

- Russia – Ukraine
- Spain – Chile

Group B consisted of the following countries 1) France and 2) United States. France is located within the same region as the main nodes but does not share many sociocultural similarities. France was selected to determine if geographical proximity affects the number of detected web sites. Finally a copy of all URLs were retrieved from a node in US to observe and reconfirm that similar to other attacks, host systems located in US are the prime target of browser based attacks.

- France
- United States

Group C countries were selected on the basis that they did not share any social or cultural similarities with the first group. We anticipate that nodes

located within these countries would be significantly less targeted than others.

- {HongKong and Chile} for Russian (R1) Dataset
- {HongKong and Russia} for Spanish (S1) Dataset

Selected countries also had to meet certain criteria to participate in the experiment. All countries had to have high availability of resources to perform the experiment. These resources included hosting companies with provided virtual private servers running operating systems with full remote privileges. They also needed to have high Internet penetration rate similar to US. High Internet penetration rate is a reliable indication of the availability of the infrastructure which support online services and consequently make the clients a prime target for attackers. Countries on the list also had to have a large population to justify return of investment (ROI) of attack, from an attacker's point of view.

We therefore selected two top level domains, .ru and .es. The Russian top-level domain .ru was selected due to its recognised history of malware distribution and malicious activity, and the Spanish TLD was chosen as it spans two continents including countries with large populations and high internet penetration rate. Table 6.19 provides a summary of attributes related to the main nodes and their companion nodes.

Table 6.19: Common attributes about countries where nodes are located

	Population	**Language**	**Ethnicity**	**Internet Penetration Rate**	**Economic State**
Russia	143 Million	Russian,	Russian, Tatar, Ukrainian	61%	High income*
Ukraine	44 Million	Ukrainian, Russian	Ukrainian, Russian (17%)	41%	Lower middle income

* data.worldbank.org

	Population	**Language**	**Ethnicity**	**Internet Penetration Rate**	**Economic State**
Spain	46 Million	Spanish	Spanish	71%	High income
Chile	18 Million	Spanish	20% Spanish Descendant	66%	High income

Network Profile and Input URL Dataset:

Nodes were selected from low profile VPS services which had no affiliation with research or security organisation. The nodes shared similar configurations in terms of operating system, retriever module, user-agent information, browser and system language, proxy (X-Forwarded-For, Via) and detection engines. The nodes, however, varied by IP address, system and system time, representing their different physical locations. The decision to place a physical node in each location was to minimise the hazard which could be introduced into the study and results arising from possible detection of proxies. We assumed that system time has no effect on the behaviour of malware delivery for a region, and language settings have been shown to have very little to no effect on the behaviour of malicious web sites. Considering the similarity between all the nodes, excluding the IP address, we therefore assumed any similarity between malware retrieved from the main node and its companion node in a different country was a result of IP-geolocation performed by the malicious web server.

We perform experiments on a two sets of suspicious dataset. These datasets included:

- Dataset of 1,000,000 web sites with TLDs corresponding to countries where Russian is the native or first official language. This dataset and associated experiments are referred to as ”R1” in the following sections.

- Dataset of 1,000,000 web sites with TLDs corresponding to countries where Spanish is the native or official language. This dataset and associated experiments are referred to as ”S1” in the following sections.

Retrieval process:

The experiment setup of both R1 and S1 datasets involved near simultaneous retrievals from participating nodes located in the selected regions identified in table 6.18. The S1 experiment was conducted four months after R1 to minimise the likelihood of IP tracking.

6.5 R1 Experiment Results

In this we provide a summary of findings for each experiment. The results indicated a pattern of geo-location across multiple nodes and regions in our R1 dataset. Common detected web sites between all countries resulted in delivery of identical malware for all components. Table 6.20 below provides a summary of the detected domains and their corresponding components.

Table 6.20: Detected web sites in R1 experiment

	Russia	**Ukraine**	**US**	**Hong Kong**	**Chile**	**France**
Domain	2696	2700	2671	2120	1209	2535
Components	5076	5071	4983	4140	1926	4735

Similar to our experiment on referer attribute in 6.2, we derived a logistic regression model for repeated binary responses to reliably determine the impact of IP-to-location variable on the number of attacks. We calculated the odds ratios and the corresponding 95% confidence interval (CI) using the Generalised Estimation Equation (GEE) with the exchangeable working correlation structure. Logistic regression model allows us to identify the relative impact of independent variables upon an outcome; in this case the impact of IP-to-location upon the likelihood of a triggered attack. Our model using R1 (i.e. Russia Dataset) shows the likelihood of an attack is

significantly higher if retrieving nodes are located in Russia and Ukraine. The likelihood of an attack decreases considerably as nodes are located farther from the locations which share fewer geographical and socioeconomic similarities with the estimated target base of the malicious web sites. We also predicted that nodes (i.e. users) located within United States would be subjected to high number of attacks. Our model determined the following:

```
The odds of "RU" is 15.3 times the odds of "CH".
The odds of "RU" is 4.1 times the odds of "HK".
The odds of "RU" is 1.1 and 1.7 times the odds of "US" and "FR".
The odds of "UKR" is 15.6 times the odds of "CH".
The odds of "UKR" is 4.24 times the odds of "HK".
The odds of "UKR" is 1 and 1.77 times the odds of "US" and "FR".
The odds of "US" is 13.8 times the odds of "CH"
The odds of "US" is 3.76 times the odds of "HK"
The odds of "FR" is 8.8 and 2.4 times the odds of "CH'' and "HK".
```

In summary, we determined:

- *Probability of attacks from nodes located in Russia* $\approx$ *Ukraine, US and* $>$ *Probability of attacks from nodes located in "FR"*
- *Probability of attacks from nodes located in "France"* $>$ *Probability of Hong Kong and Chile*

Table 6.21 shows the calculated odds ratio and Confidence Interval for our R1, Russia dataset

Table 6.21: Odds ratio and CI of R1 dataset

Sets	Odds Ratio	95% Confidence Interval for Odds Ratio	
RU/HK	4.17	3.65	4.76
RU/US	1.11	0.98	1.26
RU/FR	1.74	1.52	1.99
RU/CH	15.34	13.36	17.61
UKR/HK	4.24	3.71	4.85
UKR/RU	1.02	0.90	1.15
UKR/US	1.13	0.99	1.28
UKR/FR	1.77	1.57	2.00
UKR/CH	15.61	13.59	17.93
US/HK	3.76	3.33	4.24
US/FR	1.57	1.38	1.78
US/CH	13.83	12.09	15.81
FR/HK	2.39	2.13	2.69
FR/CH	8.81	7.80	9.96
HK/CH	3.68	3.35	4.04

6.6 S1 Experiment Results

The setup and configuration of nodes in S1 followed the design of the R1 experiment with nodes located in Spain, US, Hong Kong, Chile, France and Russia. Spain and Chile were companion nodes with similar results expected. Table 6.22 shows a summary of the detected domains.

Table 6.22: Detected web sites in S1 experiment

	Spain	Chile	US	France	Russia	Hong Kong
Domain	874	868	870	856	748	592
Components	1555	1457	1342	1278	1200	993

Our odds ratio and 95% Confidence Interval calculation for S1 (i.e. Spain) dataset also matched the result of the R1 dataset in which .es top level

domains had a significantly higher probability of attacks against nodes located in Spain and Chile which share a common language. Similar to R1 dataset, retrieval from the node located in US yielded similar number of attacks compared to the node located in the top level domain of the dataset (Table 6.23).

```
The odds of "SP" is 10.9 times the odds of "HK".
The odds of "SP" is 4.51 times the odds of "RU".
The odds of "SP" is 1.1 and 1.44 times the odds of "US" and "FR".
The odds of "CH" is 9.55 times the odds of "HK".
The odds of "CH" is 3.95 times the odds of "RU".
The odds of "US" is 9.98 and 4.1 times the odds of "HK" and "RU".
The odds of "FR" is 7.6 and 3.14 times the odds of "HK" and "RU".
```

Therefore:

- *Probability of attacks from nodes located in Spain* $\approx$ *Chile, US and* $>$ *Probability of attacks from nodes located in France*
- *Probability of attacks from nodes located in France* $>$ *Probability of Russia and Hong Kong*

Table 6.23: Odds ratio and CI of S1 dataset

Sets	Odds Ratio	95% Confidence Interval for Odds Ratio	
Sp/Ch	1.14	0.82	1.60
Sp/US	1.10	0.83	1.45
Sp/Fr	1.44	1.09	1.90
Sp/Ru	4.51	3.32	6.14
Sp/HK	10.9	8.01	14.95
Ch/Fr	1.26	0.90	1.76
Ch/Ru	3.95	2.86	5.44
Ch/HK	9.55	7.06	12.94
US/Ch	1.04	0.74	1.46
US/Fr	1.31	1.04	1.66
US/Ru	4.12	3.12	5.43
US/HK	9.98	7.41	13.45
Fr/Ru	3.14	2.42	4.06
Fr/HK	7.60	5.77	10.03
Ru/HK	2.42	1.99	2.95

6.7 Browser Exploit Kits in R1 and S1 Datasets

In this section we will discuss several popular kits and whether geo-targeting is integrated within each. We will then analyse the data captured in our-geolocation datasets and identify the usage ration of these kits within our dataset.

Blackhole exploit kit

Our dataset was collected in 2014, a year after the prime peak of Blackhole exploit kit. We therefore expect a large number of malicious web sites to be detected in our dataset. Blackhole exploit kit has embedded geolocation, IP, network and referer tracking.

Neutrino exploit kit

Neutrino Exploit kit emerged in 2012 and replaced a previously widely used Angler exploit kit. Neutrino mainly relies on iframe injection, redirecting visitors to landing pages which exploit vulnerabilities in Adobe flash player by loading a SWF exploit or through vulnerabilities in Java Runtime Environment (i.e. CVE-2012-1723, CVE-2013-0431). Landing pages deliver a variety of malware payloads such as Tofsee, Gamarue/Andromeda, Panda Banker, and various ransomware including CryptXXX [173]. Neutrino features IP tracking, geolocation, user-agent and referer traffic filtering capabilities [174]. Tofsee malware served by Neutrino exploit kit is a malware which targets the gaming community in China stealing credentials of gaming web sites.

Styx exploit kit

Styx exploit kit emerged in early 2013 and became a popular tool in automated deployment of malicious web sites. The kit provided visitor targeting functionalities through referral (i.e. unique referer, null referer), IP address and tracking, search engine bots (i.e. user agent, IP address) and specific operating system filters. Styx targeted vulnerabilities in Java, Windows font parser, Adobe reader through drive-by download or custom malware delivery with direct user download. Infections initiated from redirecting users of legitimate gateway pages to landing sites with hidden and obfuscated iframes. Embedded iframes either attempted to exploit the users vulnerabilities or request user to install the malware directly. Styx does not provide built-in geo-location targeting functionality, based on country or language, however it allows the attacker to upload targeted malware to be downloaded by a visitor.

Redkit exploit kit

Redkit exploit kit, similar to Blackhole and Neutrino offers traffic filtering based on referer, location and user-agent. Figure 6.12 shows the distribution of browser exploit kits' infected domains in R1 and S1 datasets. There are significant variations in the number of detected domains with browser exploit kits which support IP geolocation (e.g. Blackhole exploit kit, Neutrino, Redkit) while the number of infected domains with Styx, CrimeBoss remain near constant across all countries.

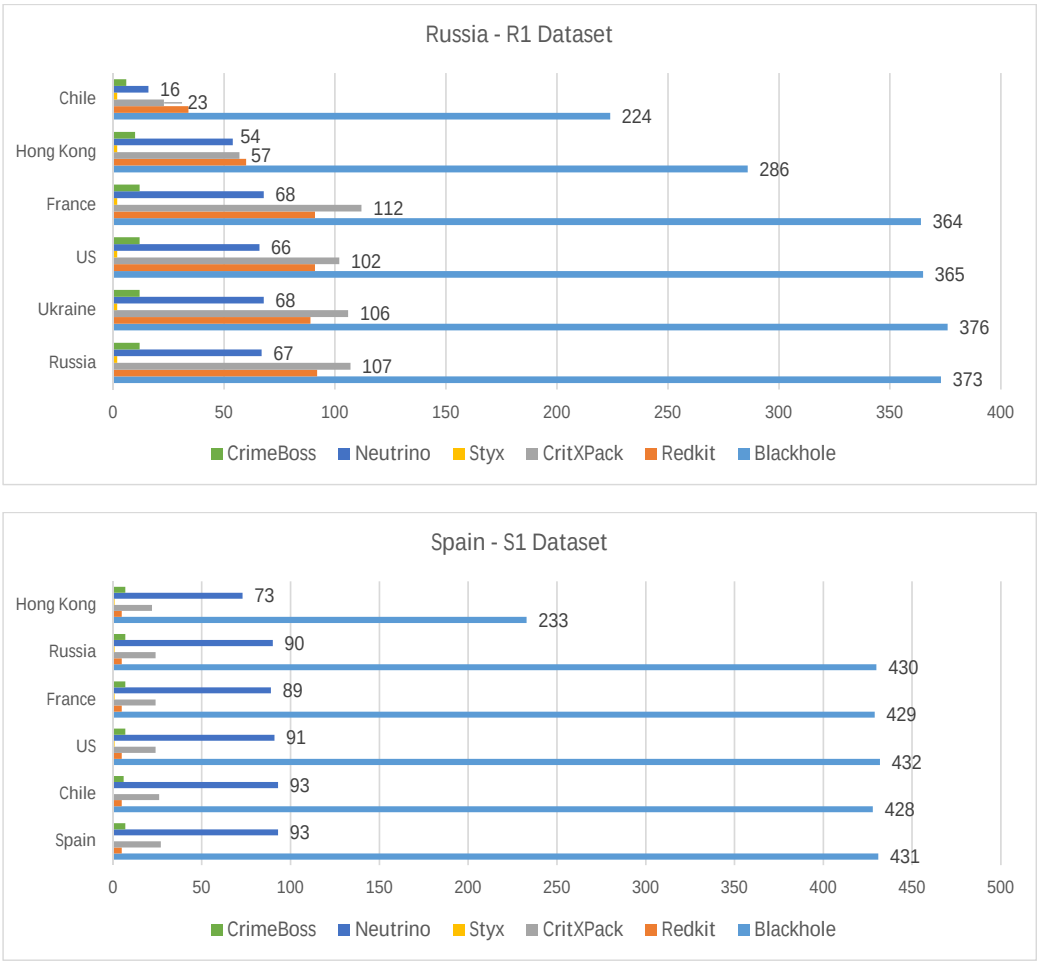


Figure 6.12: Number of web sites infected with various browser exploit packs in R1 and S1 datasets

6.8 Proxy Detection

Detection of variables within the HTTP protocol indicating the presence of a middle proxy or relay can potentially affect the behaviour of a web site. Similarly to the user agent string, X-Forwarded-For can be parsed using simple code and consequent action be taken by the web server. The following figure shows the setup of our experiment performed on one million web sites. Two clients in Netherlands we selected for retrieval of the input URL dataset. The URLs were a selection of hp-hosts dataset and .br TLD.

One of the clients simulated a proxy server fetching remote content on behalf of a client in United States. The profile of a client in United States was selected to maximise the possibility of malware delivery since hosts located in the US seem to be the de-facto higher-profile targets in various attacks. Simulation of proxy was performed by setting the appropriate HTTP header information of the retrieval request (i.e Via, X-Forward-For and From) (Figure 6.13).

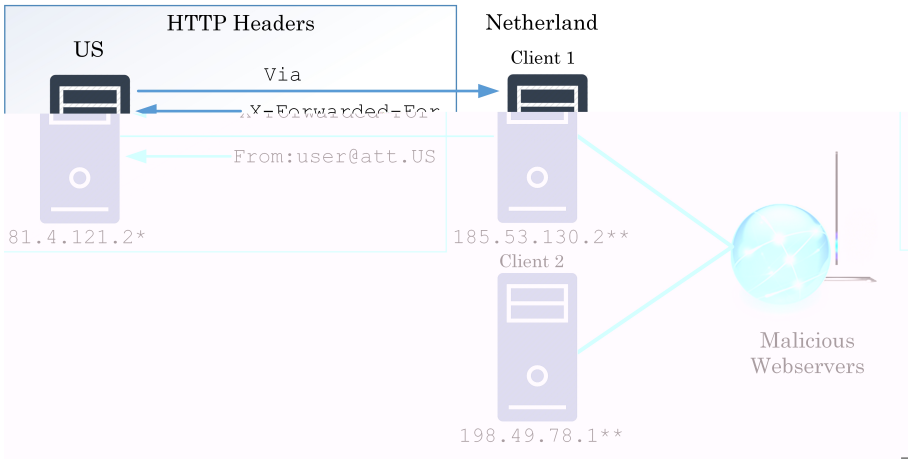


Figure 6.13: Setup of our proxy experiment

Near-simultaneous retrieval from two nodes located in the Netherlands resulted in different numbers and types of attacks. As expected retrieval

from simulated US hosts resulted in higher numbers of attacks (Table 6.24). Domains and components which were shared between two nodes however exhibited exactly the same behaviour in terms of types of malware delivered.

Table 6.24: Number of detected domains and components in our proxy experiment

	Domains	**Components**
Netherlands (Direct retrieval)	1155	1428
US (Via NL Proxy)	1273	1571
Common Domains	1029	
Common Components		1263

Measurement of the impact of the attributes which could suggest the presence of a proxy or relay server on final detection rate is crucial in the design of a client honeypot and real-world measurement experiment. Fundamentally, a client honeypot capable of detecting geo-located attacks should be able to utilise relay proxies to retrieve content and simulate multiple geo-located users without significance impact on the number of attacks. Our experiment however shows that a small subset of malicious web sites in our dataset detected and parsed proxy information within a HTTP request's header, resulting in higher number of attacks subjected at the simulated US client. Client honeypots should therefore rely on anonymous relay proxies in which no information about the real requesting client is forwarded to the receiving web server.

6.9 Summary of Findings

Successful detection of a malicious web site is not merely a function of its analysis engine but all modules involved in the process of interaction with the web site, user simulation and retrieval. Each of these modules in turn comprise of components and attributes which can potentially affect

the result of the analysis in spite of the geolocation properties of an attack. Through analysis of browser exploit kits and features provided to an attacker, we identified few of these attributes and processes which could potentially be used to bypass detection by cloaking techniques. We identified referer, language settings set within the browser and transmitted through HTTP protocol to have notable effects on the number of triggered attacks. By performing experiments on large datasets of potentially malicious web sites and varying attributes, we determined that referer value set to Google's search engine results triggers highest number of attacks as it simulates an average user activity, having reached the web site through search engine result. We also confirmed that malicious web sites target certain population through monitoring Accepted-language attribute of a client browser. Such an approach increases the probability of an attack since the content of the web site is designed to lure that specific population, using their local language. Language cloaking also provides a level of protection against detection engines as requesting clients which do not meet the language requirement of the malicious web site will be served with benign content. Language cloaking is a trivial task to implement for an attacker on either client or server side. If server-side language cloaking is implemented, it forces a detection engine to request multiple copies of a web site to identify the cloaking behaviour. Proxy identification through via, from and x-forwarded header fields of HTTP protocol however did not result in any variations in malware behaviour. Such a finding indicates that free proxies and HTTP relay services can ultimately be used to retrieve a web site content without the need for physical presence of a client honeypot in a target location, given other logical attributes of a client honeypot (e.g. referer) settings are relayed and transmitted without alteration. Utilisation of open and free proxies can significantly reduce the retrieval cost of a distributed or semi-distributed system where multiple retrievals from multiple nodes are required.

Another popular feature of browser exploit kits is tracking and moni-

toring of IP addresses and subnets from which retrieval requests are initiated. IP and subnet tracking are solely cloaking features by which a malicious content is delivered to retrieving clients if and only if they match a retrieval pattern set by the attacker, identified through the client's IP address and network subnet. This retrieval pattern could be in the forms of malware delivery to single IP address in a specific time frame, serving benign contents to clients whose retrieval pattern indicate multiple requests and subsequent blacklisting of the requesting client, or blocking an entire subnet which might be suspected of hosting client honeypots or belonging to high profile security organisations. We demonstrated that IP tracking is used with malware delivery in a portion of URLs in our suspicious dataset while no indication of network tracking through monitoring adjacent IP addresses were observed. We therefore conclude that utilisation of different IP addresses within a single subnet can be used to mimic different requesting clients with little to no effect on malware delivery and threat of undesired hazard into the study. Network tracking from high profile organisations was however not included in our study due to lack of resources.

We also performed IP geolocation experiment on two large sets of URLs. The purpose of the experiments was to identify if IP addresses are attributes used to target specific visitors. Our assumption was that IP geolocation targeting, available in various exploit kits, and IP to location databases are widely used by by malicious web sites to target specific users and server benign content to visitors who are not located in the targeted geographical location. Two datasets corresponding to two different regions were analysed to identify the impact of location on the number of detected web sites. Our experiments identified a strong correlation between top level domain (TLD) and the number of attacks targeted at users who were located in the TLD specific region. We also observed a high number of attacks in the regions and countries which shared sociocultural and official language matching the local language of the TLD input datasets.

Part III – How Do We Deal with Geolocation Attacks?

Chapter 7

Geolocation Retrieval System

In this chapter, we discuss the design and implementation of a prototype system used in identification of location targeting web sites and discuss several components and techniques which would result in efficient multi-retrieval and detection of said web sites. We initially identify the functional requirements which all client honeypots and retrieval systems should meet in order to allow for simulating diversely located nodes using logical and physical attributes of a retrieval node. We utilise our findings in chapter 6 to embed logical and physical attributes with high impact on malware delivery into our prototype design. We also discuss techniques by which the cost of multiretrieval can be reduced through integration of techniques such as compression, minification and content extraction and deletion. Experiments are performed to identify their impact on performance and reliability of detection. Non-functional requirements of the proposed system such as reliability components in place are then discussed in section 7.3. Non-functional requirements ensure lower hazards in large scale and real-world experiments by introducing methods by which effects of hazards introduced by failure of an system component can be minimised. We also demonstrate the overall design of our system and provide detail on few modules of the prototype system.

7.1 Adversarial Information Retrieval Systems

Widespread use of web spam and SEO techniques to achieve economic gain has paved the way for significant research in the area of Adversarial Information Retrieval Systems (AIRs). AIR systems are designed to retrieve, index and rank information from a subset in which a smaller subset may have been maliciously manipulated. AIR systems have focused mainly on the area of search engines. There is a financial incentive for an attacker to manipulate online information to alter search engine rankings. Imagine a search engine which frequently redirects its users to spam or malware web site; such a scenario would result in enormous damage to the search engine's reputation for reliability and credibility, and consequently lower market share and profit. Therefore search engine developers and researchers have worked on methods to detect the ever-changing techniques used by spammers in spam design, delivery and detection mitigation. Malicious web sites, their behaviour toward users and search engines and malware delivery share many similarities with web spam.

- Both manipulate a set of information maliciously for various reasons
- They employ similar techniques to avoid or minimise detection
- Both stand to gain from higher rankings in search engine results
- Their effect on the reliability of a search engine is immense

There are various similarities between adversarial information systems for web spam and malicious content. Essentially many of the techniques used to deliver web spam and mitigate detection by search engine bots can be used in the delivery of malicious code. Server-side and client-side techniques such as multiple redirections, referer cloaking and IP address cloaking are a few examples of such techniques. Differences between the two exist however in their detection methods. Web spam detection is based on content or structural comparisons between multiple copies of a web site

(e.g. link-based, bag of words, tag and structure based), while malicious content requires signature, anomaly or execution based techniques to detect an attack. Yet the methods used to mimic an average user and fool a web server into delivering web spam can be turned around to defeat cloaking and detection mitigation techniques by malicious web servers.

In this chapter a system for efficient retrieval and detection of geolocation and cloaked malware is proposed. The improved system is designed using the data gathered from chapter 6 which identified the likelihood and effect of different attributes in malware delivery. Figure 7.1 summarises the criteria for a multi-retrieval honeyclient with geolocation capabilities.

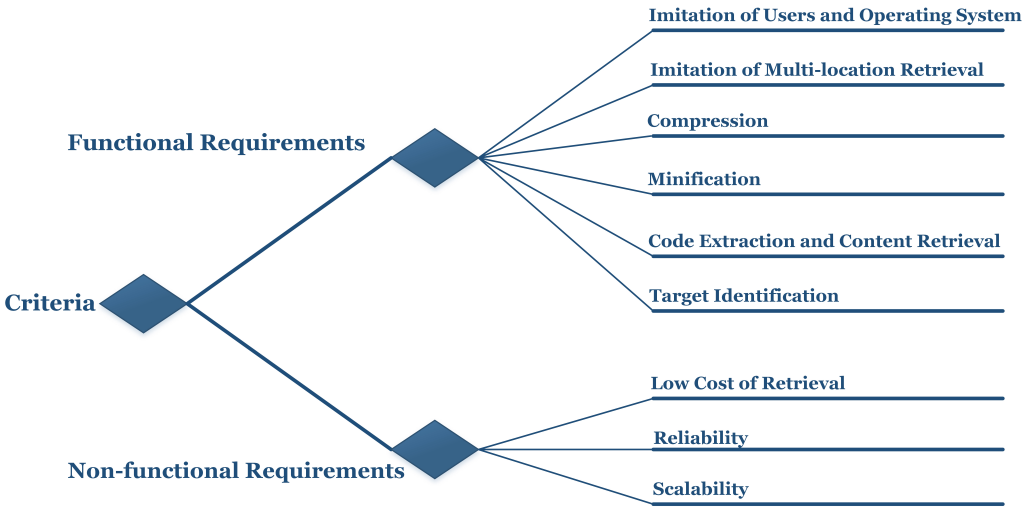


Figure 7.1: Functional and non-functional requirements of our system

7.2 Functional Requirements

Functional requirements are defined as behaviour or functions which a system should support and provide. The functional requirements of our system are support for behaviours which allows a client honeypot to imitate the behaviour of a regular user and a specific operating system and browser, thereby mimicking an average user browsing a web site. These

functions and behaviours should provide the capability to fool a remote web server into delivering malicious content. We divide these functionalities into the following categories:

7.2.1 Support for Imitation of User Behaviour and Operating Systems

We know from the research literature and review of our own experiment (i.e. referer experiment) that certain attributes and functionalities within the browser and operating system have profound effects on incidence of malware delivery. These are variables which cannot be translated to a physical location, yet they may be combined with other location variables to select and trigger an attack for a visiting user. Most of these attributes have been discussed in previous chapters but we will summarise them into functionalities which the proposed system should provide.

First and foremost, the variables which the system should support relate to the browser used in the retrieval process. The browser is the module which interacts with the web server, therefore its behaviour should represent that of an average user. Imitating the average user requires the retriever module to avoid behaviours that are unique to crawlers, either from search engines or security agencies. These behaviours occur in relation to how fast web sites are retrieved in a specific period of time, what elements should be extracted and fetched from a web site, what browser personality should be presented to the remote web server, and how the retriever should handle attacker strategies such as redirections and cookies. The system design should also support the various personalities of operating systems. These could be in the form of a virtual retriever specifying a particular client operating system, or the utilisation of real operating systems and browsers for retrieval.

7.2.2 Support for Imitation of Multi-location Retrieval

Retrieval variables are the various main attributes transmitted to a remote server during the request phase. Regardless of the architectural pattern in use, our system should be able to support designs which offer simulation of various retrieval variables. The variables by which a targeted geolocation attack can potentially be performed can be divided into two main categories – logical and physical variables:

- Logical variables in our system are attributes independent of a physical location or resource. This includes any attribute which can be set by the experimenter within the software environment. These attributes includes variables contained within the browser, operating system or communication protocol.
- Physical variables are attributes set by the hardware or physical location of the instrument of retrieval. These variables are harder to mimic or spoof (Figure 7.2).

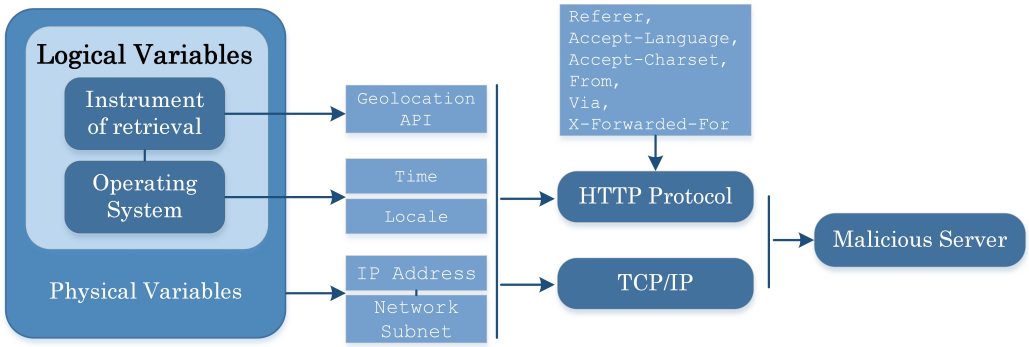


Figure 7.2: Logical and physical variables which may expose the location of a user

Physical variables such as IP addresses are difficult to spoof compared to logical variables which can be set within the browser, operating system or transmission protocol. An IP address may be spoofed by a host engaged

in a Denial of Service (DOS) attack; these attacks only require a destination address to act as a target and no response is expected nor required. TCP protocol works by a three-way handshake starting a session between two nodes. Therefore a spoofed IP address would not be able to complete the handshake and receive the web site content. Accordingly, designs based on on-location virtual or physical systems, distributed designs or proxy-based designs are required.

7.2.3 Support for Multiple Detection Engines and Uniform Analysis

Analysis engines depend on various techniques to detect an attack. These techniques range from using signatures of a previously detected attack, to examining the state of the system upon execution of a code, to looking for anomalies or elements indicative of a malicious behaviour. No single detection system is capable of achieving a 100 percent detection rate with a minimum of false negatives. We therefore suggest that the proposed system should be capable of accommodating multiple detection engines to allow a user to enhance their detection capabilities. Ideally the detection module should be capable of adjusting the level of detection based on a user's preference. Uniform analysis is another issue in deployment of geographically diverse or distributed honeyclients. While a centralised system mitigates the overhead cost of varying detection engines in different locations, it requires fetched content to be transferred back to a single location for analysis resulting in higher bandwidth utilisation and subsequently higher cost.

7.2.4 Support for Compression

Compression of content provides a reliable method for decreasing bandwidth consumption without major undesired effects. While removal of content may result in non-executable malicious content and a subsequent

false negative, compression keeps the order and structure of content in place. HTML documents are prime candidates for compression since various content such as text, JavaScripts and HTML tags include ASCII values containing highly repetitive characters, and hence are highly compressible [165]. Compression should only be applied to non-compressed components such as ASCII elements. Compression of already compressed content, such as executable and image files (e.g. JPEG) files, will only result in unnecessary processing and may result in a larger output file than the input.

7.2.5 Optional Support for Minification

Minification of HTML and JavaScript is another approach for reducing the size of the content required to be transmitted. Minification refers to the process of removing whitespaces and unnecessary elements (e.g. comments) within the HTML and JavaScript code. Minification can be considered a lossy compression since white spaces are permanently removed and the input and output files do not match bit-by-bit, although they are rendered and execute similarly. Minification improves performance for multiple reasons:

- Reduced compression time since the input file is smaller [166]
- Reduced transfer time since the input compressed file is considerably smaller
- Reduced file-size resulting in lower script load time (e.g. for analysis)
- It is parsed faster, since comments and white-spaces don't have to be explicitly ignored
- Minified content does not require de-minification at the client level

7.2.6 Optional Support for Code Extraction and Content Deletion

Code extraction and content deletion are other methods by which the amount of bandwidth can be reduced, resulting in reduced cost of operation. Code extraction refers to the practice of extracting the dynamic and executable code embedded within a web site (e.g. JavaScript). Content deletion on the other hand removes the static elements such as images or text while keeping the structure of the html file intact. Deletion of content to minimise bandwidth usage requires removal of elements deemed unnecessary in the detection of malware behaviour. Minification and code extraction are not essential requirements of the system. They can however be implemented if additional bandwidth reduction is required. Figure 7.3 shows functional requirements met through each client honeypot module.

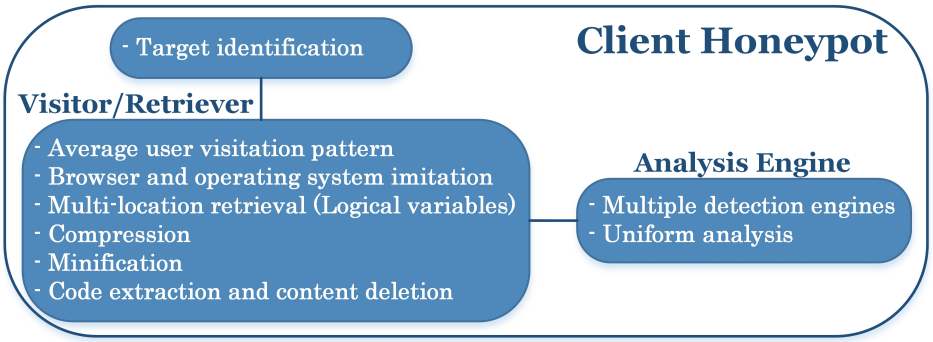


Figure 7.3: Client Honeypot modules and their respective functional requirements

7.2.7 Target Identification

The experiments performed in previous chapter revealed that there is a pattern in malware targeting users based on their geographical locations and associated attributes. IP address, language and subsequent geolocation of visiting hosts seemed to be the major factor in the delivery of ma-

licious content. Similarities between the numbers of malware delivered to countries with similar language indicates a targeting pattern.

7.3 Non-Functional Requirements

The proposed platform should be designed to meet certain criteria to effectively and efficiently detect geolocation attacks. The criteria recognised as essential in the design are:

7.3.1 Low Cost of Retrieval

Bandwidth utilisation poses one of the biggest challenges in detection of browser geolocation attacks and contributes vastly to the overall cost of a design. Similar to web spam analysis, multiple copies from nodes physically or virtually located are required to detect an attack. Multiple retrievals are a basic requirement of the detection. The challenge is to reduce the cost of detection per URL by reducing the number of retrievals and maintaining an acceptable rate of false negative. We attempt to address bandwidth utilisation through compression, minification, code extraction and content removal.

The cost of administration includes supervision by an individual who manages the process of installation and deployment, and ensures consistency of all components within the system. The cost of administration is significantly reduced in our design through reliability components discussed in the next sections.

7.3.2 Reliability

Malicious web site detection generally involves scanning a large number of web sites over long periods of time; therefore a certain level of automation is required to reduce the overhead costs of management. Automation

requires agents to monitor the behaviour of the system process, detect abnormalities and recover from a failure. Overall the proposed system should be able to automate the process of content retrieval, storage and analysis without significant administrative overheads.

7.3.3 Scalability

The system should allow the addition of a large number of nodes with various system configurations. This includes nodes with simulated or real operating systems or browsers. Scalability allows for higher accuracy as nodes can be placed in more regions. Scalability also allows for removal or addition of nodes in cases where a retrieving node is detected by a web server, therefore increasing resilience against detection.

7.4 Geolocation Attack Detection System Prototype Design and Implementation

In this section we provide general guideline for the main components and their interactions in order to set the honeyclient and retrieval variables to ideal attributes and maximise the possibility of malware retrieval.

7.4.1 Imitation of User Behaviour and Operating System

Instruments of retrieval are modules responsible for simulating the capabilities of user browsers, interacting with potentially malicious web servers and fetching content. These modules can be standalone browser applications or simulated and real retrieval modules as part of a low or high interaction client honeypot. We decided on utilising a virtualised browser for reasons of security and scalability. Mechanize browser was selected due to the simplicity of its code and since it met the requirements of the design. We compare our virtual browser against the functional and non-functional requirements set in section 7.3. Mechanize is a virtual browser library which provides user behaviour and OS imitation capabilities through the following embedded features:

- Browser and OS imitation capability through user-agent
- robot.txt, redirection, refresh, authentication, cookies and session handling
- Embedded crawling, link parsing and following capabilities
- HTML form filling and submission

7.4.2 Imitation of Multi-location Retrieval

Mechanize allows for logical geolocation identifiers to be manipulated by altering the browser and HTTP header values. Mechanize provides:

- HTTP referer header management
- Proxy integration: Native proxy capability allowed our system to integrate advantages of free proxy services into the design.
- Http header manipulation (e.g. accept-language)

7.4.3 Multiple Detection Engines and Uniform Analysis

Detection modules are methods of analysis performed on the retrieved content in order to detect malicious code. The detection mechanism in the design of our system required a detection engine capable of analysing content saved offline. These analysis engines can be signature, state or anomaly-based. Our implementation relies detection engines integrated into YALIH low interaction client honeypot. YALIH's detection engine relies on:

- Two signature-based antivirus engines (i.e. Clam and AVG Antivirus) with capability to add additional engines
- YARA's signature and pattern matching engine
- De-obfuscation and de-minification of JavaScripts
- Portable Document File (.pdf) analysis

Imitation of multi-location, user behaviour and operating system are mandatory requirements of a client honeypot capable of detecting geolocation attacks. Other functionalities such as compression, minification and content removal are optional functionalities that result in higher performance in terms of retrieval and analysis time through reducing the size of the retrieved and analysed content. These functionalities may however affect the output of the analysis by altering the content of the input malicious

web site into a client honeypot analysis engine. We therefore initially determine the impact of these processes on the reduction of the size of the content in retrieval process and subsequently measure their effects on the reliability and accuracy of detection.

In order to evaluate the efficiency of compression, minification and content deletion algorithm on the size of the transmitted content subsequent performance gains and their effects on the detection rate, the following experiments were performed:

1. Retrieval of .html and .js was conducted on 150,000 random web sites to determine the percentage of web sites using compression. This dataset is refered to as **"RandomCompression"** in the following sections.

2. Retrieval of web sites which used compression in the previous dataset (i.e. RandomCompression) to determine the impact on the size and retrieval time if an uncompressed content were to be requested from the web server. This dataset is referred to as **"PostCompression"** in the following sections.

3. 1000 malicious .html and .js files to evaluate the effect of minification and content removal on the number of detected content in our client honeypot.

7.4.4 Low Cost of Retrieval

Due to the nature of geolocation attacks, several copies of a single web sites from multiple locations are required to adequately detect an attack. The cost of detection per URL is therefore significantly higher in geolocation attacks than a traditional attack. In this section we analyse compression, minification, content deletion and extraction and determine if these approaches can help minimise the cost of detection per URL through reducing the size of the content delivered or analysed. We will also deter-

mine if these approaches have any negative effect on the detection rate of our geolocation retrieval system.

7.4.5 Support for Compression

Current implementation of HTTP/1.1 supports compression through content-coding values, or on a hop-by hop basis through transfer of coding values between a client and web server. The Accept-Encoding and Content-Encoding headers in HTTP/1.1 provide a mechanism for the client to inform the server which encoding format it can support. Content encoding in HTTP/1.1 occurs through standard deflate or de-facto "Gzip" algorithms. Deflate and Gzip are lossless algorithms in which the compressed output should be capable of producing an identical copy of the input without losing any data in the process. A lossless compression algorithm is a requirement in HTML/JavaScript compression, as loss of a single bit may alter the markup of the page, or make a JavaScript file un-executable.

The Gzip standard provides a file format consisting of a series of compressed data sets using the "DEFLATE" compression method. Gzip however is the de-facto standard used by Apache web servers and is considered a more reliable and stable approach. The Gzip file structure format contains a header which identifies it as a Gzip file format, and additional flags which provide information about the data content [167, 168].

All popular browsers support compression and decompression, and the effectiveness of these mechanisms depends on the behaviour of the web server. A web server may transfer uncompressed content for various reasons. A web server may either not support the compression, or bypass compression, if load on the server passes a specific threshold. Although compression algorithms require minimum CPU time, in relation to the high processing power of today's web servers, the load on the web server may be high if the number of simultaneous connections is very large. Studies on web server performance have illustrated insignificant

differences in reply rate using uncompressed versus deflate and Gzip compression of content [165]. Compression of content is applied using popular libraries such as gzib. Compression in our semi-distributed prototype system can be performed between:

- **Web server → Remote Retriever** – Mechanize natively supports compression and decompression of content between web server and client which avoids the need for integration of third party compression APIs. Web server to remote retriever however depends on compression support by the web server.
- **Remote Retriever → Centralised Node** – The retrieved web site content can be compressed using Gzip library API by the remote retriever node if the web server serves uncompressed content and forward it to the centralised node for decompression and analysis. The remote retriever may also directly forward the payload if the content is already compressed.

In order to evaluate the utilisation and efficiency of compression algorithm on data transmission, an experiment involving retrieval of .html and .js was conducted on our "RandCompression" dataset. Application of top 1000 web sites was avoided as they all utilise compression (Table 7.1). The experiment resulted in the identification of 73,527 web sites which utilised compression representing nearly 50 percent of all web sites.

Table 7.1: Experiment on 150,000 web sites revealed only half use compression

Number of Web sites	Percentage of web sites used compression	Compression rate achieved (Supported Web sites)
150,000	49%	61%

Effect of Compression on Bandwidth Utilisation

A follow up study of web sites which utilised compression in the previous experiment (i.e. 73,527) was conducted to illustrate the time and bandwidth usage of these URLs with and without compression (Figure 7.4). As mentioned earlier, this dataset is referred to as **"PostCompression"** in the following sections.

The results illustrate that bandwidth utilisation and amount of data transferred is significantly lower (i.e. 61 percent) while the time it takes for the web server to compress the content and web browser to un-compress it results in 30 percent higher retrieval and processing time. The effects of compression on the remote web server serving a single copy of a URL is minimal since 30 percent obtained in our experiment represents a combination of the time it took for all web servers to compress content, transmit and for a local node to fetch and decompress 10 simultaneous URLs per second.

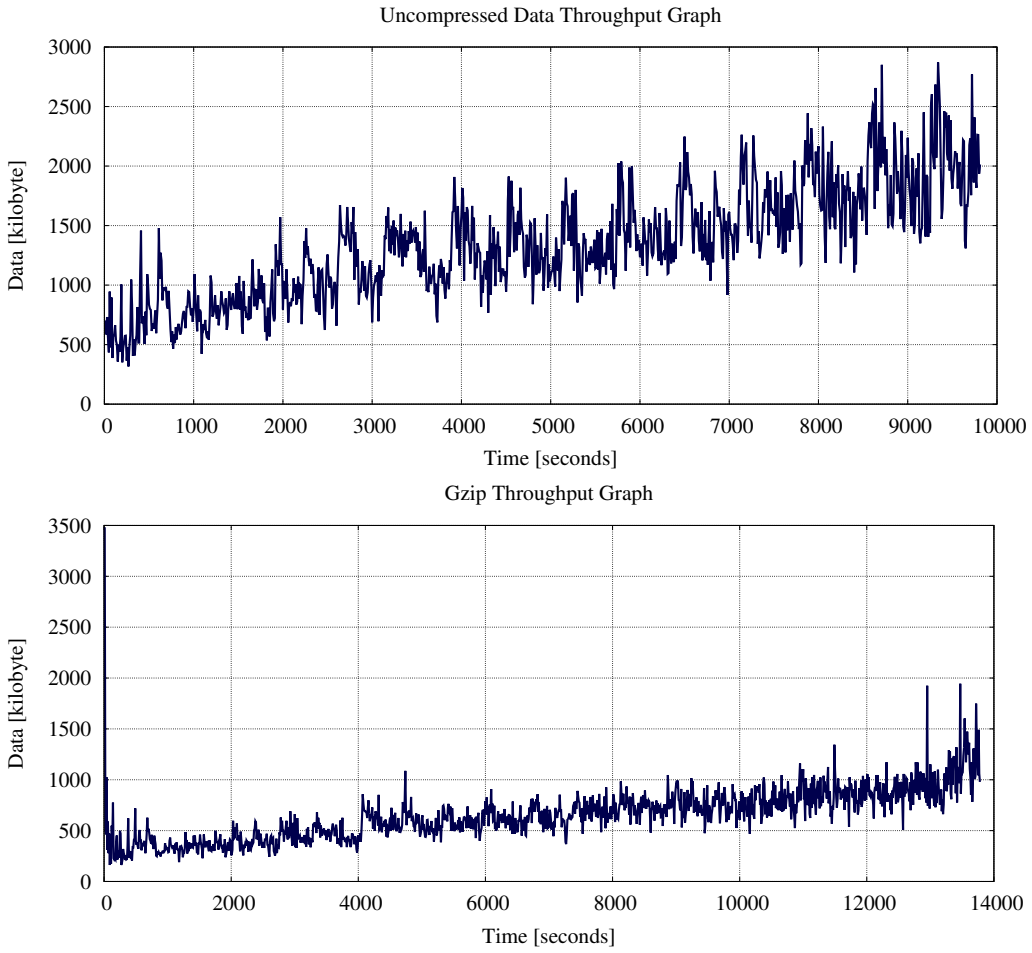


Figure 7.4: Compression results in lower bandwidth utilisation in the expense of retrieval time

Effect of Compression on the Size of the Content

Compression on all html and associated js files of our PostCompression dataset (i.e. 73,527 URL dataset) in a controlled environment using the highest compression level of Gzip achieved a significant average of 73 and 62 percent compression for .html and .js files respectively. There is however a threshold at which Gzip compression is advantageous to an un-

compressed content. The general rule is that the larger the input file is; a higher compression rate can be achieved. Compression of retrieved html and JavaScript files less than 140 Bytes in our experiment resulted in negative compression ratio, indicating a larger file size for compressed files than input content.

7.4.6 Support for Minification

Minification changes the representation of the code and static signature of web site by removing the white spaces in the code, while not changing the actual code functions. Minification can be applied to the fetched content and transmit between the remote retriever and central node. Minification of HTML and JavaScript was performed using HtmlCompressor library [169] [169]. The library allows preservation of content within `<pre>`, `<textarea>`, `<script>` and `<style>` tags will be preserved and remain untouched (with the exception of `<script type="text/x-jquery-tmpl">` tags which are compressed as HTML), while comments, multiple spaces unnecessary spaces inside, around and between tags are removed.

Minification performed on our PostCompression dataset resulted in an average 13$\pm$ percent size reduction on retrieved content. Its effect was minor when combined with compression which does not justify additional time and processing overhead. Unlike compression however, no relationship or threshold between the size of files and minification rate were observed (Figure 7.5). Looking at Figure 7.5, it can be observed that beyond a certain threshold, there is a clear similarity and pattern between the rate of compression and minification. The similarity can be associated with the number of whitespaces the file holds, which is removed by minification and compression algorithms.

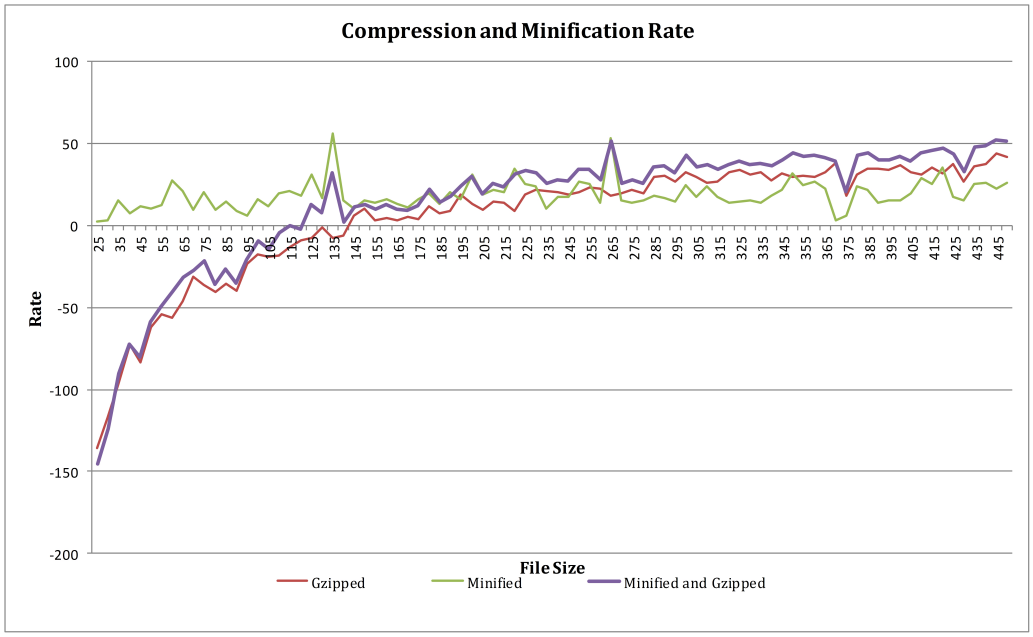


Figure 7.5: illustrate the relationship between file size with compression and minification rate

7.4.7 Support for Code Extraction and Content Deletion

Content deletion is another method by which the amount of bandwidth can be reduced, resulting in reduced cost of operation. Content deletion on the other hand removes the static elements such as images or text while keeping the structure of the html file intact.

The deletion approach performs well for web sites which are deterministic. That means they follow the same execution path in all environments and do not produce varying output as a result of certain triggers, such as a generated random key. Malicious web sites may trigger the execution of web sites through attributes unique to each system, or at a particular point in time [39]. This significantly reduces the probability of detection as the required triggers may not be available or met. Certain detection techniques such as signature-based detection, however, are capable of detection triggered attacks as they rely on signatures rather than execution of

the script. Signature detection would however fail in a situation in which an already detected attack is obfuscated using a random key or attribute generated upon execution. The deletion approach is also only applicable to html files.

Structural-based detection of web spam provided a starting point for content selection in our design, and subsequent detection of malicious behaviour. We hypothesised that standard text and images within a web site are not necessary in the detection of malicious content and can be considered as noise and omitted. Content deletion to minimise bandwidth usage requires removal of elements deemed unnecessary in the detection of malware behaviour. We divide these elements into essential and non-essential.

- The essential group consists of codes, elements and variables which could be used in a malicious attack. This group also covers hyperlinks which may reference a malicious component on a different domain. These can be any elements in the various HTML element categories (e.g. heading content, phrasing content, and embedded content).

- Non-essential elements make up the text or the presentation of the text. Examples of such elements are <p>
. Heading elements such as <h1> and <hgroup> fall under this category. Table 7.2 shows essential variables which should not be omitted from a web site structure. Some of the variables may belong to one or more categories.

Table 7.2: List of essential elements which should not be removed from a fetched HTML

Category	**Elements**				
Metadata content	<base>	<link>	<script>		
Flow content	<a>	<code>	<embed>	<iframe>	<keygen>
	<object>	<var>			
Interactive content	<button>	<input>			

Extraction or deletion of such elements at the retriever side can be performed using various libraries. BeautifulSoup Python library for instance removes all the text from an html file but keeps the structure of the web site intact. References to objects, links, JavaScript code and all tags are retained. Since these elements and their associated information are preserved, the processed file will create a similar outcome in the copy received by the retriever. The Python HTML parser library, lxml, used in our design also allows removal or extraction of individual attributes. Minification and content deletion or extraction are not essential requirements of the system. They can however be implemented if additional bandwidth reduction is required.

Table 7.3 illustrates the result of performing, minification, content removal and compression on our PostCompression dataset. Similar to minification, deletion of text content achieves insignificant results in reducing the size of the outcome html files. These techniques can however be used if minimizing the cost through reducing retrieval size is essential.

Table 7.3: Compression and minification on web site components achieves significant reduction in size

Files	**Number**	**Compressed (Average)**	**Minified (Average)**	**Minified and Compressed (Average)**	**Deletion (Average)**
HTML	73635	73%	11%	75%	16%
.JS	451250	62%	15.5%	70%	-

7.4.8 Target User Identification

Target identification attempts to reduce the number of retrievals required to detect geolocation attacks, resulting in more efficient system by lowering the retrieval and storage cost per URL. The idea relies on the notion that malicious web sites lure specific users by integrating related contents. Therefore if we are able to identify the content of a web site we can determine the likely target user based for whom the content has been specifically designed. We assume that users who do not fit the required geographical attributes set by the attacker will be served or redirected to benign content. Determining the specific target of a web site could be a challenge requiring a large number of retrievals to cover all possible combination of browsers, operating systems or http variables' combinations. We however believe that based on certain properties of a URL or web server, it is possible to minimise the number of retrievals and identify the likely targets. These values include:

Top Level Domain

Top level domain provides a simple method of determining the likely target of a web site. Generally specific top level domains are used to host web sites for a particular country, region or category. For instance it is expected that a large percentage of .br domains would have a target base of Brazilians or Portuguese speaking users. Therefore parsing the URL before retrieval for TLD attributes provides an indication of targeted or expected visitors. Extraction of TLD can be achieved through various libraries in different programming languages. In Python libraries such as tld, tldextract provide the capability to divide a URL into multiple parts and compare with a list of predefined country TLDs.

WHOIS Data

WHOIS data as discussed in previous chapters contains information about the registrant of the domain. Although the accuracy of the information can be disputed since there are no mechanisms in place to accurately verify the information, certain attributes such as the email address of the admin – which is required to activate the service – may expose the location of the registrant. WHOIS also contains "country" field which represents the country of the registrant of the domain. The combination of these fields could be used to identify the location of registrant and subsequently estimate the targeted user base. WHOIS library in python provides a simple way of retrieving information for a URL. Example below shows a web site mimicking the popular WinRAR compression utility (Figure 7.6). The web site is designed in English language and serves two malicious executable files.

```
>>> w=whois.whois("www.wiinrar.info")
>>> w
{
...
'name': 'Domain Administrator', 'city': 'Phoenix', '
   expiration_date':
datetime.datetime(2015, 10, 23, 15, 43, 47),'zipcode': '85016',
'domain_name': 'WIINRAR.INFO', 'country': 'US', 'whois_server':
None, 'state': 'AZ', 'registrar': 'Namesilo, LLC (R620-LRMS)',
'referral_url': None, 'address':['1928 E. Highland Ave. Ste F104
   ','PMB# 255'],
...
'emails': 'pw-426c862434a11a057ee1da6bfb090503@privacyguardian.
   org'}

>>> w.country
US
```

Figure 7.6: Estimating the target users of a web site through WHOIS information

Country information may not be available for a large number of smaller web sites or those purchased in bulk. These web sites are sometime pur-

chased from a third-party retailer and may hide the location of the true customer. Email address or the Registrar information may help provide required geolocation information in such cases (Figure 7.7).

```
>>> w=whois.whois("http://www.drugzdravia.ru/")
>>> w.country
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'WhoisRu' object has no attribute 'country'

>>> w
{'status': 'REGISTERED, DELEGATED, VERIFIED', 'expiration_date':
datetime.datetime(2016, 3, 31, 0, 0), 'domain_name': 'DRUGZDRAVIA
    .RU',
'creation_date': datetime.datetime(2012, 3, 31, 0, 0), 'registrar
    ': 'REGTIME-RU',
'name_servers': ['ns5.ukrdomain.net.', 'ns6.ukrdomain.net.'], '
    org': None, 'emails': None}

>>> w.registrar
REGTIME-RU
```

Figure 7.7: Extraction of Top Level Domain (TLD) information from a URL through Registrar attributes of WHOIS data

Web site Language

The language in which the web site content has been designed can be used to determine its target user base. The content of a malicious web site can be designed to meet the language and regional attributes of a visiting users and lure the visitors into malicious download and malware execution. Retrieval of a single copy of a web site may help reveal the target visitors of a specific URL through analysing the content and identifying the embedded content's language. Although the web site language is a reliable indication of target users, the location by which the initial copy is retrieved can result in unreliable detection. For instance initial retrieval of a web site retrieved from US may result in the web server redirecting users to English version

of a web site. Google's language detection library (i.e. langdetect) is highly effective in determining the language of an input text. Graph below illustrates the attributes we utilise to determine the likely target user-base of a malicious web site. Figure 7.8 shows the retrieved information for each URL to determine the likely user target.

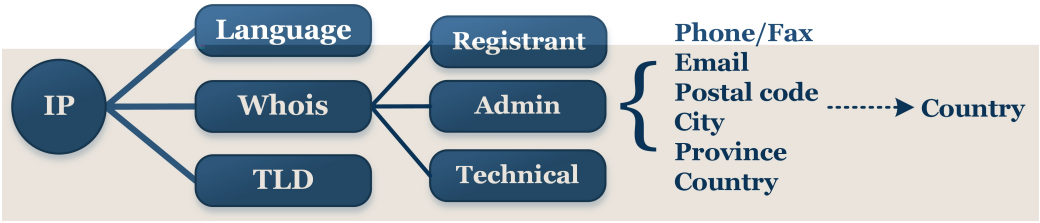


Figure 7.8: Retrieved information for each URL to determine the likely user target

We follow a simple algorithm to achieve target user identification (Figure 7.9). It is assumed that web sites belonging to a country code Top Level Domain (ccTLD) are much more likely to be designed with content to lure users (i.Target) specific to the country by which the TLD is registered (i.TLD). Therefore any web site with a certain country code TLD should be retrieved once from a node residing in the corresponding country (RNode1) or set of countries with similar attributes (Region1). Retrieval from a node located in country which shares no sociocultural similarity with the country associated with the URL's ccTLD (RNode1) can reliably determine if the web site employs geolocation cloaking (RNode2). Detection of attacks based on geolocation or sociocultural attributes require an Additional retrieval from a node within the set in which the RNode1 resides. Web sites belonging to generic TLDs (e.g. .com, .net, .info) however require additional information to identify their likely target users. We rely on information in the DNS registry to determine the location and properties of the retrieving nodes from which a web site content should be retrieved. WHOIS data contains the country, email address and registrar information which could provide an indication of the likely attacker. Re-

trieval of WHOIS information does not trigger IP tracking at the remote malicious web server as the data is retrieved from DNS registries. Registry information however might be partially available depending on the requirements of each registry. If WHOIS information does not indicate a certain geographical location, we therefore rely on a single retrieval to determine the web site language. We follow the algorithm below for URL retrieval:

```
Sets ccTLD(), Region1(), Region2(), Region3(), Regionn:
SEQUENCE i in w(1...n)
  i.tld=i.extract(Top Level Domain)
  If i.tld is in generic[.com, .net. org, .info]
    Extract CountryCode in (i.whois(country, registrar, email))
    for CountryCode in i.whois:
      i.Target=country > registrar > email
  ELSEIF i.tld is in ccTLD():
    i.Target=i.tld
  ELSE
    i.target=RNode-US.retrieve().detect_language(i)
  RNode1=Region(i)
  RNode2= RNode in union(Region1, Region2...Regionn).difference(
     Region(i))
  RNode3=Region(i).difference(RNode1)
```

Figure 7.9: Retrieval algorithm to detect geolocation and targeted attacks

This dataset was utilised to determine if our target identification through WHOIS, top level domain information and language of the suspicious web site could reliably determine the likely audience of the malicious web site and result in a higher detection rate compared to a basic retrieval.

A large number of malicious web sites are registered within Generic Top Level Domains (gTLDs). We attempt to identify the likely user targets of those web sites through information within public DNS registries and the web sites language using our retrieval algorithm. We tested our algorithm on a small dataset of 10,000 generic top level domains (.e.g. .com, .net, .org, .info, .biz).

Initially DNS information was parsed for any embedded geolocation information such as country of registration, email address corresponding to a country (e.g. user@domain.co.uk). Each URL in the dataset was subsequently retrieved from the country of registered domain, the country associated with the web site language, and was subsequently retrieved from another country serving as the control node. The language of the retrieved content from each node was detected using Googles language detection library (i.e. langdetect).

Our dataset resulted in detection of 189 malicious domains from which 125 domains were registered in China and served malware only to users in China and Hong Kong. Our algorithm managed to determine the likely target of these web sites from WHOIS information (i.e. 86 Web sites) and Language information (i.e. 39 web sites) as shown in table 7.4. Our algorithm also detected 27 malicious web sites primarily registered in United States which neutrally served malware to all visiting users regardless of the location of the retrieving node and therefore did not perform any IP or location cloaking. Overall, our experiment revealed that domain registration information and language detection can be used to identify the likely targets of a malicious web sites and limit the number of retrievals for detection.

Table 7.4: The result of our experiment on a small set of generic TLDs to evaluate our target user identification algorithm

Generic Top Level Domains			
Number of Detected Web Sites	**WHOIS**	**Language**	**Country**
125	86	39	China
37	31	6	United States
27	9	18	Others

7.5 Reliability

Reliability of our design can be measured using two distinct criteria: 1) reliability in detection 2) reliability in operation

7.5.1 Reliability in Detection

In this section we determine if the process of minifying the content of a web site including .html and .js files result in higher rate of false positive and negative alerts. The evaluation will focus on minification and content removal only, as content are compressed using a lossless compression and are decompressed prior to detection and analysis by the client honeypot.

Minification

Minification can affect a web site content by removing the whitespaces and in the process alters the sequence by which an attack signature has been created. We performed minification and content removal on a dataset of 1000 malicious .html and .js files we had previously collected to determine their effect on detection with our low interaction client honeypot. Number of detected signatures were 284 versus 281 for original and minified files respectively, resulting in one percent reduction in the number of detected files as the result of minification. Minification of the files resulted in six percent overall lower file sizes (Table 7.5).

Table 7.5: Number of detected items, pre and post minification

Number of files	Pre-Minification	Post-Minification	Avg. File Size Reduced
100	284	281	6%

Content Removal or Code Extraction

There are multiple techniques to remove unwanted content or extract the features from a web site. Our experiment relied on two different techniques (i.e. BeautifulSoup library and lxml library) to remove pre-determined tags and content. Our aim was to measure the effect of content removal on detection and the average reduction of the dataset. Default feature removal in BeautifulSoup library resulted in 52 percent reduction in detection at the expense of 36 percent decline in the required bandwidth. Such a high rate of false negative is undesirable in a large scale study. Python lxml library on the other hand provides the capability to specify the tags and specific content to be removed from a file. This flexibility allows the user to balance desired detection rate and rate of size reduction for a dataset. Low number of detected items for lxml (default setting) is due to the removal of all high risk tags such as iframes, (Java) scripts, embed and <src> tags. The detected items (i.e. 8) correspond to the web sites which had embedded regular hyperlink pointing directly to malicious web site. These web sites were therefore used as gateway web sites to lure visitors. Inclusion of <script> tag contributed significantly to the number of malicious content in our database while frame-related tags and inclusion of embedded "src" files resulted in detection of two additional malicious files (Table 7.6).

Table 7.6: Number of detected items, pre and post content removal

	Allowed Tags	**Pre- Removal**	**Post-Removal**	**Average File Size Reduced**
BeautifulSoup	-	284	134	36%
lxml	Structure, Style, links	284	8	69%
lxml	Structure, Style, links, Scripts	284	164	35%
lxml	Structure, Style, links, Scripts, (i)frames	284	166	34%

7.5.2 Reliability in Operation

Certain failure scenarios can be mitigated using redundancy in the overall platform. For instance, using multiple reliable and public DNS servers reduces or eliminates the risk of errors in address translation. Monitoring

the operation of geo-nodes and associated running processes requires an active mechanism.

Notification regarding the status and failure of modules in our platform relies on the SMTP protocol library native in Python. SMTP allows secured email delivery of notification through SSL and TLS standards. The notification shows the failure, the action taken and the IP address of the remote host in which the client honeypot is located. We identified several components within our prototype system which could result in failure of different modules.

Failure of Management Module

Management module and monitor module are placed on the same host system and therefore oversight by an administrator is expected. Failure of the management is handled by the monitor module, yet failure of the monitor component results in total shut-down of the host system. It is assumed that in a management system run on trusted hardware, failures are unlikely to occur.

The monitor module monitors the operation of the management component through the process and system utility module (e.g psutil). The monitor component checks if the process is running at intervals set by the user. If the management process seems stopped or killed for any reason, the monitor will attempt to restart the process and notify the administrator by email accordingly.

Failure of the Storage or Detection Modules

Storage is a function of the node and main storage system. Nodes located in various locations have their own internal storage systems which upon completion of a retrieval per URL set, host the retrieved content. Content is then compressed and transferred back to the main storage for analysis.

It is assumed that the storage system of any host is directly linked with its main function; that is the storage does not fail unless the host system fails. Any error in writing the data to the disk, due to various reasons such as a full disk, generates an error which causes the retriever to fail. Failure of the retriever triggers the monitor agent to attempt to restart the retriever for a limited number of times and notify the administrator by email accordingly.

The detection engine's operation is similarly monitored by the monitor component with a similar error-handling and alert on error mechanism. Analysis also happens at the central storage unit where we assume administration oversight is available to handle any critical system fault.

Failure of Communication Protocols

Communication protocols are the most difficult aspect of the system to monitor and control, as they rely on intermediary devices beyond the direct control of the user. The communication protocols which might introduce fault into the normal operation of the components may happen at various levels. We assume, however, that such devices are from trusted sources and operate on trusted hardware. Our implementation uses redundancy and multiple retries to minimise the possibility of an error. Redundancy is used especially in communication between geo-nodes and DNS servers in which multiple public but reliable DNS servers are utilised for address translation. The system also utilises multiple retries of communication in order to minimise the chances of an unavailable content due to an offline server or network congestion, in the case of geo-node to web server interaction for instance (Table 7.7).

Table 7.7: Reliability modules embedded in our design to reduce the risk of failure on remote nodes

Communication	**Reliability Control**
Management Geo-Nodes	Heart Beat
Geo-Nodes DNS Server	Redundancy, Multiple retries, Logging errors
Geo-Nodes Web server	Multiple retries, Logging errors
Geo-Nodes Storage System	Transfer protocol (i.e. rsync) reliability features including retries

Failure of Geo-Nodes

Geo-nodes are located on various servers and platforms across the world, which can have various rules and restrictions in terms of bandwidth or process utilisation. Any such restriction could result in a node being shut down or process killed or stopped. Reliable communication between nodes and management module is therefore needed. There are two different scenarios where an operation of a node could stop:

1. **The Process is killed:** The monitor module monitors the operation of the management component through the process and system utility module. The monitor component checks if the process is running on intervals set by the user. If the management process seems to have been killed for any reason, the monitor will attempt to restart the process and notify the administrator by email accordingly. There are various ways to check if a process or service is running in a Linux environment. Bash for instance allows several different methods to monitor and restart a process in case it is killed. The following code illustrates a simple bash script which monitors the honeypot.py process every 5 seconds and logs an exit message. Our implementation however relies on a cross-platform library called psutil. Psutil allows retrieval of information about running processes and utilisation of system resources. This allows further management of processes, per-

mitting not only the monitoring the state of a processes but also the extent of resources used by the specific process. Our implementation uses this library for various monitoring tasks due to the reliability of this technique compared to bash or other alternatives. The following figure illustrates an excerpt of our monitoring function which checks if a specific process identified by PID is running on the system on specific intervals set by a user within the honeypots config file (i.e. config.checkfrequency).

2. **The Process is Stopped**

   There are also scenarios in which a process may be running on the system yet halted and stopped due to various causes. For instance during our experiments running on virtual private systems (VPS), instances of process halt were observed due to high utilisation of bandwidth by a single process (i.e. the retriever module honeypot.py). Monitoring the running process on the system using its Process ID (PID) only, fails to detect such an abnormality since the process is still running on the system and with a unique PID. Therefore the monitoring module should not solely rely on a process PID, but also consider utilisation of resource usage monitoring to ensure the process is running normally.

   Our implementation of resource usage monitoring relies on bandwidth consumed by the retriever process. Psutil provides the capability to detect the bandwidth usage of a process and determine if a process is deemed as halted or stopped. It is assumed that a task which includes the retrieval of large number of URLs should utilise bandwidth continually until the task is finished. A long delay in bandwidth utilisation defined by the user is hence considered an abnormality, and detected by the monitor agent. The monitor agent subsequently attempts to kill the process, restart and resume the task. A notification is then sent to the administrator with the

details.

Failure handling and recovery should be capable of resuming a task from the point at which the failure occurred. This is essential since restarting the retriever process of a large URL set, for instance, would result in re-fetching the entire set which is undesirable for several reasons (e.g. bandwidth, time difference, IP tracking). Our implementation of task recommence relies on a file based approach in which the monitor keeps track of the retrieval process status through monitoring three variables: first URL in the list, last URL successfully retrieved and last URL in the list. By comparing these variables, the monitor keeps track of a task's progress.

3. **System Shutdown**

   A total system shutdown/restart of a node is detected through a heartbeat mechanism where notifications on pre-defined intervals are sent to the monitor module. If the monitor does not receive "alive" notifications from a node in a predefined period of time, the node is deemed shutdown and a notification is sent to the administrator. Our implementation does not currently handle recovery from a system shutdown/restart scenario beyond the notification as the design is reliant on a functional monitor module for fault recovery. A total system shutdown could potentially be due to various reasons (e.g. power loss at remote location) which requires an administrative assessment.

7.6 Scalability

The system should allow the addition of a large number of nodes with various system configurations. Scalability allows for higher accuracy as nodes can be placed in more regions. The client honeypot used in our design supports proxy servers which allows a user to simulate retrieval from

a different geographical location. Our low interaction client honeypot also requires few resources to be installed and operated on a client machine and allows simulation of various operating systems and browsers. Simulated browser also ensures that the host machine will not be at risk of attack from malicious web sites, as opposed to a high interaction client honeypot which runs a real browser to interact with a malicious web site.

7.7 Summary

In this chapter we identified the functional and non-functional requirements of a client honeypot to detect geolocation targeting attacks. Functional requirements of such client honeypots require capabilities to mimic system and network properties of remote users through logical and physical attributes such as browser language settings. Non-functional requirements specified, ensure the bias and hazards are minimised in a design and experimental study.

A prototype, meeting the requirements specified, was designed and implemented. We identified that compression of content is a highly effective technique in reducing the bandwidth utilisation of client honeypots in retrieval of large sets of URLs. Minification is another viable solution in reducing the size of retrieved content per URL in a distributed client honeypot architecture. Content retrieval and extraction approach however resulted in high rate of false negatives.

Validation of a system involves assessments to examine the efficiency and reliability of a system and its components. Using monitoring agents such as heart-beat systems, process and network monitors and subsequent recommence and notification, we illustrated that distributed retrievals can be managed reliably to recover from failure of various components (e.g. management, agent nodes) and minimise variations in retrieval time and subsequent introduction of hazards into the experiments.

Part IV – Drawing Conclusion

Chapter 8

Conclusion

Web based attacks have largely seen a shift toward targeting end users and away from vulnerabilities in server side services. This shift is largely due to factors such as popularity of todays web services and lower complexity of attacks on client services. Organisations deploy several layers of security to protect their systems and data against unauthorised access while a large fraction of end users do not utilise and/or are not familiar with any security tools. End users unfamiliarity with security products contribute vastly to the number of online DDoS attacks, malware and Spam distribution.

In this thesis we primarily focused on understanding the motives behind targeted attacks from an attacker's perspective and identifying attributes on the client side which could be used to target a specific user or group of users through real-world experiments and empirical analysis. The following contributions are summarised according to the goals set in Chapter 1, section 2.

8.1 Contributions

- **Characterising the nature of the attacks –** In this research we discussed economical and legal reasons for an attacker to target specific

group of end users rather than a large scale blind attack. Social factors such as a popular activity, an interest or a popular service used by a large portion of population were few examples of social factors which made a population a target and susceptible to targeted attacks through social engineering. We identified that economic conditions of the target country plays a major role in the selection of its population for various types of attacks. Countries with higher GDP, developed infrastructure and high internet penetration are more likely to be targeted for attacks to obtain information which could be exchanged for a currency in an accessible and secure underground market. Users in other countries are more likely to be targeted spam and used as bots for distributed denial of service attacks. We also discussed how political, legal and extradition laws can be a major factor in an attack scenario, significantly influencing the source and target of an attack.

- **Identify the user attributes used in targeted attacks –** One of the outcomes of this research was identifying the client attributes contributing to geo-information and targeted attacks. We discussed logical attributes of browser and operating system and physical attributes of an end user which could be associated with location of a user and therefore used to target the user with specific and customised attack. End users browser attributes such as user-agent information, accepted language, via and from were few attributes identified as potential variables in an attack. The logical variables despite lower accuracy, require few resources on the attacker's side with potential to be spoofed by sensors associated with detection tools. Physical attributes such as IP and subnet information can also be associated with geographical location through active and passive measurements. Active measurements although more accurate, generally depend on network topology information and packet transmission delay to calculate an estimate location. These techniques require

landmarks and significantly higher detection speed. An approach believed to be widely used by attackers is through passive IP to location database services which relay geolocation detection to other entities while providing an adequate level of accuracy for a large scale attack. Integration of geolocation databases was also observed in few browser exploit kits.

- **Review and understanding the extent of the threat –** Campaigns of targeted attacks have revealed a shift toward targeted attacks which have a significantly higher rate of success compared to a generic attack. Attacks detected on server side services captured by server honeypots have shown a pattern of targeted attacks based on geographical location of users and organisational profile of the targets. Attacks on client browsers however, although mentioned in many papers and reports by security vendors are not thoroughly investigated and lack a measurement study to determine their prevalence. There are various browser exploit kits with embedded IP-tracking and geolocation services, however no large scale study had been done to confirm and measure the rate by which these techniques were being used. This thesis set to perform large scale study of malicious web sites using multiple end user attributes to determine if those attributes had an impact in triggering an attack. We performed experiments on one million dataset of URLs with varying HTTP header values such as referer, browser language and HTTP protocol attributes. We identified that referer attributes had significant impact on malware delivery as it mimicked an end user following a search engine result. Our findings revealed that referer settings which simulate users reaching a web site via a search engine or via internal links receive significantly higher number of malicious content compared to users with null referer variable. The retrieved malware in the context of different referer header values varied not only in their numbers but also type of malicious payload delivered. We

used a logistic regression model to identify the relative impact of referer on the likelihood of a page being malicious. Using odds ratios and the corresponding 95% confidence interval (CI) using the Generalised Estimation Equation (GEE) with the exchangeable working correlation structure we calculated the Google referer string is 14.52 times more likely to trigger malware delivery than a null value referer. We also performed a large scale study on the effects of browser language attributes on malware delivery. Our experiment with Russian top level domain and multiple visitor agents simulating multiple browser language settings revealed a large impact on the number of malware delivered. The behaviour of malicious web sites in terms of types of delivered malware was however remained nearly identical in all instances. Language associated with the top level domain of the input URL was shown to be nearly 14 times more likely to trigger an attack than other languages. A challenge facing security researchers and organisation active in the field of browser based malware detection is the simplicity by which IP and network tracking can be implemented. Similar to referer tracking, IP and network tracking are features of several high profile browser exploit kits. IP and network cloaking require low resources to implement on the attacker's side however they can significantly affect the result of an experiment by monitoring the visitation pattern of a retrieving client or clients located in a subnet. Subsequent visits to the web server will not trigger malware delivery which imposes additional cost in terms of IP allocation to any detection agents relying on multi-retrieval process. Failure to address IP tracking capabilities into malicious web site detection can introduce inaccurate analysis regardless of the detection engines as benign contents are delivered on subsequent visits. Our experiments revealed that malware delivery based on IP tracking is used in a small subset of malicious web sites. Inclusion of IP tracking features in browser exploit kits are however trends we will

likely see in all automated browser exploit tools. The experiment on network tracking for adjacent IP addresses however did not show any indication of tracking. HTTP header values such as via and from and x-forwarded had no impact on the total number of detected attacks. We therefore concluded that any retrieving agent used in the identification of malicious web sites should have the capability to simulate a local language and set proper referer to reflect a popular activity of an end user. Visiting agents may however take advantage of proxy capabilities to retrieve content to simulate end users in a specific location without introducing large bias into studies.

- **Design of a low interaction client honeypot –** We presented the design and analysis of a low interaction client honeypot that integrates a combination of multiple signature based antivirus engines and pattern matching using string or regular expressions. The designed honeypot is extensively used in our experiments and throughout this research. The motive behind design of our client honeypot was to have a modular honeyclient with logical geolocation and enhanced end user simulation capabilities while integrating various detection engines. We selected two signature based detection engines to provide detection with very low false positive and a pattern matching and regular expression engine to allow identification of potentially malicious characteristics of a web site. This allowed us to write custom rules to detect browser exploit kits and modify false positive and false negative rates accordingly.

- **Hazard and Operability study of measurement studies using a low interaction client honeypot –** Real-world experiments with large number of confounding variables are subjected to a high degree of hazards in terms of inaccurate analysis if those variables are not controlled or their effects are not properly measured. This research focused on measurement study of logical and physical variables within

a user's host system which may be used to subject an end user to a targeted attack. In a dynamic environment where many physical and logical variables can affect the outcome of the study, there needs to be a method by which every risk in analysis introduced by components, processes or subjects of the study can be controlled. The approach also should allow identifying the likelihood and severity of each uncontrolled variable and determine its likelihood of occurrence and identify mitigation strategies to minimise the risk or mitigate them entirely. Hazard and Operability study which is widely used in chemical engineering to analyse the processes and hazards in HAZOP approach was applied throughout this thesis to the design of a repeated measurement studies of malicious web sites. The hazards in the honeyclient's design, web protocols, environmental factors and intermediary devices which could affect the results of the experiments were defined. Mitigation strategies were implemented in our low interaction client honeypot's design such as multiple detection engines, reliability components and browser module with enhanced simulation capabilities and throughout our experiments in which the effects of referer attribute were measured and steps were taken to minimise the impact of various attributes such as IP, network and organisational profile tracking, timing on the outcome of the each experiment.

- **Design and implementation of a Geolocation Information Retrieval System –** We proposed the design of an Information Retrieval System for detection of geolocation and cloaked malware. The system design was the result of data gathered from our experiments which identified the likelihood and effects of different attributes in malware delivery. We identified that any retrieval system for detection of geo-located malware should be able to meet several functional requirements such as imitation of browser language, referer and multi-location retrieval. Our language experiment revealed that language

setting plays a role in determining the target users of malicious web sites and therefore should reflect the likely user for which the web site content has been catered for. We therefore proposed identifying the likely target base of a web site prior to retrieval. Our algorithm relies on geographical information in public Domain Name System registries and a single retrieval from a node. Identifying the target user allows for reduced number of required retrievals per URL which results in significantly lower cost of retrieval for a large dataset. We also experimented with various techniques to determine the effects of compression, minification and content deletion on the performance and reliability of detection. We identified compression and minification to be viable approaches which should be considered in the design of any retrieval system that requires multiple copies for analysis.

8.2 Future Work

This thesis covered the goals and objectives set in the first chapter. In this section we discuss areas in which this work can be improved. Data collection – Our experiments relied on datasets containing one million web sites. The size of our dataset was constrained due to time limitation of our data collection phase and limited resources in terms of retrieving agents in locations where IP geolocation attributes were to be analysed. The external validity of our experiments are therefore limited to our datasets and specific domains (e.g. .ru) and may not represent the behaviour of all malicious web sites on the internet.

8.2.1 Larger Measurement Studies

Future work could expand our work by running geolocation experiments and measurement studies on larger datasets and within more regions. In

order to maximise the internal validity of our experiments we relied on reliable commercial virtual private servers in multiple countries which offered similar performance, reliability and bandwidth. Such a decision was taken to minimise the bias and threats introduced by factors such as difference in retrieval times between multiple sensors or an offline sensor. This however limited us in terms of the number of sociocultural regions we could investigate. For instance, we initially intended to investigate geolocation attacks between Portugal and Brazil. Brazil was an ideal research subject with large population and many sociocultural similarities with another country in a different continent. Although a large number of sensors could be obtained in Portugal, finding reliable, affordable and similarly configured sensors proved to be a significant challenge in Brazil. Future work could extend our experiments to cover more regions and domains. Our experiments also focused on geolocation attributes such as IP and language in the context of sociocultural similarities. Future studies with larger resources can extend our studies to cover the impact of a significant event (social or political) in a specific region and analyse its effect on others in terms of the behaviour of malicious web sites, social engineering techniques and type of malware delivered.

8.2.2 Client Honeypot Improvements

The client honeypot used throughout this research was an in-house low interaction client honeypot which focused on integration of multiple detection engines, browser simulation capabilities to allow varying attributes used in geolocation attacks and performance in mind. The secondary focus of our client honeypot was modularity and reliability in operation for bulk processing of URLs. There are however areas in which our low interaction client honeypot can be improved to allow for improved emulation and increased usability.

1. **Visitor agent** – A low interaction client honeypot inherently has lim-

itations in terms of emulating many functions of a real browser and fails to adequately simulate browser plugins and extensions. Future work could focus on an open source browser with enhanced browser and plugins simulation, native JavaScript support, sandboxing and improved logging capabilities.

2. **Reporting agent** – Crawling a dataset of multi-million URLs can result in the generation of a large sum of textual data. An embedded reporting agent would greatly facilitate the analysis of a multi-location and retrieval experiment which generates log files in the range of a few Gigabytes.

Overall, level of browser emulation remains one of the biggest shortcomings in the design of current low and high interaction client honeypots. Virtual browsers in low Interaction honeyclients are very limited in terms of simulating the personality and functionalities of a real browser which may expose them as client honeypots. They also have very limited support for extensions and plugins which result in missed attacks targeting those vulnerabilities. Browsers on high interaction honeyclients on the other hand are real browsers which are designed for in-depth analysis and specifically detection of zero-day attacks and less on bulk processing of a large number of web sites. Therefore, similar to server honeypots, a medium interaction client honeypot with improved simulation capabilities is the next step in the evolution of honeypot systems.

Bibliography

[1] A. L. Zain, "El estado actual de la industria digital y las tendencias que estn modelando el futuro," report, ComScore, 2013.

[2] Maxmind, "Industry leading ip intelligence - "ip address to country database," 2002.

[3] B. Medler, "Player dossiers: Analyzing gameplay data as a reward," *Game Studies Journal*, vol. 11, no. 1, 2011.

[4] Z. Jianwei, G. Liang, and D. Haixin, "Investigating chinas online underground economy," *University of California Institute on Global Conflict and Cooperation*, 2012.

[5] J. Mauro C. Hernandez and J. Afonso Mazzon, "Adoption of internet banking: proposition and implementation of an integrated methodology approach," *International Journal of Bank Marketing*, vol. 25, no. 2, pp. 72–88, 2007.

[6] C. Herley and D. Florłncio, "A profitless endeavor: phishing as tragedy of the commons," in *Proceedings of the 2008 workshop on New security paradigms*, pp. 59–70, ACM, 2009.

[7] C. Kanich, C. Kreibich, K. Levchenko, B. Enright, G. M. Voelker, V. Paxson, and S. Savage, "Spamalytics: An empirical analysis of spam marketing conversion," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 3–14, ACM, 2008.

[8] M. Goncharov, "Russian underground revisited," *Cybercriminal Underground Economy Series*, 2014.

[9] FCA, "Interim report: Annex 11 international comparisons," report, Financial Conduct Authority, 2015.

[10] KPMG, "Payment developments in africa," report, 2015.

[11] A. Keren and K. Elazari, "Internet as a cii-a framework to measure awareness in the cyber sphere," in *Cyber Conflict (CYCON), 2012 4th International Conference on*, pp. 1–13, IEEE, 2012.

[12] M. Fossi, E. Johnson, D. Turner, T. Mack, and J. Blackbird, "Symantec report on the underground economy," report, 2008.

[13] M.-A. Ester and R. Benzmller, "Underground economy," report, 2008.

[14] W. Kim, O.-R. Jeong, C. Kim, and J. So, "The dark side of the internet: Attacks, costs and responses," *Information Systems*, vol. 36, no. 3, pp. 675–705, 2011.

[15] R. Geromel, "Hackers stole $ 1billion in brazil, the worst prepared nation to adopt cloud technology," *Forbes Magazine*, 2012.

[16] A. Shull, "Global cybercrime: The interplay of politics and law," 2014.

[17] H. Coroiu, "Disorderly conduct: localized malware impersonates the police," report, 2011.

[18] D. Alyias, D. Batchelder, J. Blackbird, J. Faulhaber, and D. Felstead, "Microsoft security intelligence report - an in-depth perspective on software vulnerabilities and exploits, malware, potentially unwanted software, and malicious websites," report, Microsoft, 2012.

[19] Microsoft, "Easy money: Program:win32/pameseg," 7-2-2014 2011.

[20] J. Baltazar, J. Costoya, and R. Flores, "Infiltrating waledac botnet's covert operations," *TREND MICRO*, 2009.

[21] H. L. Ragragio and M. Radu, "The cloud or the mist?," in *Virus Bulletin (VB2009) Conference*, 2009.

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

[23] 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*, (1855682), pp. 6–6, USENIX Association, 2009.

[24] Y. Alosefer and O. Rana, "Honeyware: A web-based low interaction client honeypot," in *Proceedings of the 2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, (1799609), pp. 410–417, IEEE Computer Society, 2010.

[25] A. Ikinci, T. Holz, and F. C. Freiling, "Monkey-spider: Detecting malicious websites with low-interaction honeyclients," in *In Proceedings of Sicherheit, Schutz und Zuverlssigkeit*, 2008.

[26] C. Seifert, I. Welch, and P. Komisarczuk, "Honeyc - the low-interaction client honeypot," in *New Zealand Computer Science Research Student Conference NZCSRCS*, 2007.

[27] N. Provos, "Spybye," 2007.

[28] J. R. Rocaspana, "Shelia: A client honeypot for client-side attack detection." 2009.

[29] S. Xiaoyan, W. Yang, R. Jie, Z. Yuefei, and L. Shengli, "Collecting internet malware based on client-side honeypot," in *Young Com-*

puter Scientists, 2008. ICYCS 2008. The 9th International Conference for, pp. 1493–1498, 2008.

[30] Y.-M. Wang, D. Beck, X. Jiang, and R. Roussev, "Automated web patrol with strider honeymonkey," in *Proceedings of the Network and Distributed System Security Symposium*, pp. 35–49, 2006.

[31] C. Seifert, "Capture honeypot client (capture-hpc)," 2006.

[32] P. Kijewski, C. Overes, and R. Spoor, "The honeyspider network fighting client-side threats," in *In First.org (Vancouver, 2008)*, 2008.

[33] M. Roesch, "Snort - light weight intrusion detection system," in *In Proceedings of the USENIX Conference on System Administration*, 1999.

[34] A. Dellaera, "Python low-interaction honeyclient," 2013.

[35] C. Guarnieri, M. Schloesser, J. Bremer, and A. Tanasi, "Cuckoo sandbox-open source automated malware analysis," *Black Hat USA*, 2013.

[36] A. Dewald and F. C. Freiling, "Adsandbox: Sandboxing javascript to fight malicious websites," in *In Proc. of ACM Symposium on Applied Computing*, ACM, 2010.

[37] K. Rieck, T. Krueger, and A. Dewald, "Cujo: Efficient detection and prevention of drive-by-download attacks," in *In Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2010.

[38] M. Cova and C. Kruegel, "Detection and analysis of drive-by-download attacks and malicious javascript code," in *In Proceedings of the 19th International World Wide Web Conference (WWW)*, 2010.

[39] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy, "Spyproxy: Execution-based detection of malicious web content," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, ACM, 2007.

[40] O. Hallaraker and G. Vigna, "Detecting malicious javascript code in mozilla," in *Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems*, IEEE, 2005.

[41] C. Reis, J. Dunagan, H. J. Wang, O. Dubrovsky, and S. Esmeir, "Browsershield: Vulnerability-driven filtering of dynamic html," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 3, 2007.

[42] M. Silbey and P. Brundrett, "Understanding and working in protected mode internet explorer," 2006.

[43] T. Wadlow and V. Gorelik, "Security in the browser," *Communications of the ACM*, vol. 52, no. 5, pp. 40–45, 2009.

[44] M. T. Qassrawi and H. Zhang, "Detecting malicious web servers with honeyclients," *Journal of Networks*, vol. 6, no. 1, pp. 145–152, 2011.

[45] M. T. Qassrawi and H. Zhang, "Client honeypots: Approaches and challenges," in *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, pp. 19–25, IEEE, 2010.

[46] M. Wood, "Turning scareware devious distribution tactics into practical protection mechanisms," 2011.

[47] J. C. Knight and N. G. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," *IEEE Transactions on Software Engineering*, no. 1, pp. 96–109, 1986.

[48] K. Killourhy and R. Maxion, "The effect of clock resolution on keystroke dynamics," in *Recent Advances in Intrusion Detection*, pp. 331–350, Springer, 2008.

[49] P. A. Fishwick, "Neural network models in simulation: a comparison with traditional modeling approaches," in *Proceedings of the 21st conference on Winter simulation*, pp. 702–709, ACM, 1989.

[50] W. F. Tichy, "Should computer scientists experiment more?," *Computer*, vol. 31, no. 5, pp. 32–40, 1998.

[51] M. V. Zelkowitz and D. R. Wallace, "Experimental models for validating technology," *Computer*, vol. 31, no. 5, pp. 23–31, 1998.

[52] R. A. Maxion and R. R. Roberts, "Methodological foundations: Enabling the next generation of security," *Security & Privacy, IEEE*, vol. 3, no. 2, pp. 54–57, 2005.

[53] V. R. Basili and M. V. Zelkowitz, "Empirical studies to build a science of computer science," *Communications of the ACM*, vol. 50, no. 11, pp. 33–37, 2007.

[54] D. I. Sjoberg, T. Dyba, and M. Jorgensen, "The future of empirical methods in software engineering research," in *Future of Software Engineering, 2007. FOSE'07*, pp. 358–378, IEEE, 2007.

[55] R. A. Maxion, T. A. Longstaff, and J. McHugh, "Why is there no science in cyber science? a panel discussion at nspw 2010," in *Proceedings of the 2010 workshop on New security paradigms*, pp. 1–6, ACM, 2010.

[56] T. D. Cook, D. T. Campbell, and A. Day, *Quasi-experimentation: Design & analysis issues for field settings*. Houghton Mifflin Boston, 1979.

[57] P. Lukowicz and S. Intille, "Experimental methodology in pervasive computing," *Pervasive Computing, IEEE*, vol. 10, no. 2, pp. 94–96, 2011.

[58] F. Massicotte and M. Couture, "Blueprints of a lightweight automated experimentation system: a building block towards experimental cyber security," in *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*, pp. 19–28, ACM, 2011.

[59] T. E. Carroll, D. Manz, T. Edgar, and F. L. Greitzer, "Realizing scientific methods for cyber security," in *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*, pp. 19–24, ACM, 2012.

[60] N. H. Ab Rahman and K.-K. R. Choo, "A survey of information security incident handling in the cloud," *Computers & Security*, vol. 49, pp. 45–69, 2015.

[61] T. Benzel, "The science of cyber security experimentation: the deter project," in *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 137–148, ACM, 2011.

[62] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Experience with deter: a testbed for security research," in *Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006. 2nd International Conference on*, pp. 10 pp.–388, IEEE, 2006.

[63] R. Maxion, *Making experiments dependable*. Springer, 2011.

[64] K. S. Killourhy and R. A. Maxion, "Should security researchers experiment more and draw more inferences?," in *CSET*, 2011.

[65] C. Seifert, "Cost-effective detection of drive-by-download attacks with hybrid client honeypots," 2010.

[66] T. Srivatanakul, J. A. Clark, and F. Polack, *Effective security requirements analysis: Hazop and use cases*, pp. 416–427. Springer, 2004.

[67] K. Lano, D. Clark, and K. Androutsopoulos, *Safety and security analysis of object-oriented models*, pp. 82–93. Springer, 2002.

[68] B. Daruwala, S. Mandujano, N. K. Mangipudi, and H.-c. Wong, "Threat analysis for hardware and software products using hazop,"

in *Proceedings of the international Conference on Computational and information Science*, pp. 446–453, 2009.

[69] R. Winther, O.-A. Johnsen, and B. A. Gran, *Security assessments of safety critical systems using HAZOPs*, pp. 14–24. Springer, 2001.

[70] F. Patern and C. Santoro, "Preventing user errors by systematic analysis of deviations from the system task model," *International Journal of Human-Computer Studies*, vol. 56, no. 2, pp. 225–245, 2002.

[71] V. Anupam, A. Mayer, K. Nissim, B. Pinkas, and M. K. Reiter, "On the security of pay-per-click and other web advertising schemes," *Journal of Computer Networks*, vol. 31, pp. 1091–1100, 1999.

[72] Y.-M. Wang and M. Ma, "Detecting stealth web pages that use click-through cloaking," report, Microsoft Research Technical Report, MSR-TR-2006-178, 2006.

[73] D. Y. Wang, S. Savage, and G. M. Voelker, "Cloak and dagger: dynamics of web search cloaking," in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 477–490, ACM, 2011.

[74] E. Kirda, N. Jovanovic, C. Kruegel, and G. Vigna, "Client-side cross-site scripting protection," *computers & security*, vol. 28, pp. 592–604, 2009.

[75] F. Kerschbaum, "Simple cross-site attack prevention," in *SecureComm 2007. Third International Conference on Security and Privacy in Communications Networks and the Workshops.*, pp. 464–472, IEEE, 2007.

[76] Z. Mao, N. Li, and I. Molloy, *Defeating cross-site request forgery attacks with browser-enforced authenticity protection*, pp. 238–255. Springer, 2009.

[77] F. Howard, "Malware with your mocha? obfuscation and antiemulation tricks in malicious javascript." 2010.

[78] L. Invernizzi, P. M. Comparetti, S. Benvenuti, C. Kruegel, M. Cova, and G. Vigna, "Evilseed: A guided approach to finding malicious web pages," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 428–442, IEEE, 2012.

[79] Y.-M. Wang, Y. Niu, H. Chen, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King, "Strider honeymonkeys: Active, client-side honeypots for finding malicious websites," *See http://research. microsoft. com/users/shuochen/HM. PDF*, 2007.

[80] H. ODea, "The modern rogue - malware with a face," in *Proceedings of the Virus Bulletin Conference*, 2009.

[81] I. Erete, V. Yegneswaran, and P. Porras, "Alice@ home: Distributed framework for detecting malicious sites," in *Recent Advances in Intrusion Detection*, pp. 362–364, Springer, 2009.

[82] G. Stringhini, C. Kruegel, and G. Vigna, "Shady paths: Leveraging surfing crowds to detect malicious web pages," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 133–144, ACM, 2013.

[83] M. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt, "Trends in circumventing web-malware detection," *Google, Google Technical Report*, 2011.

[84] K. Zeeuwen, M. Ripeanu, and K. Beznosov, "Improving malicious url re-evaluation scheduling through an empirical study of malware download centers," in *Proceedings of the 2011 Joint WICOW/AIRWeb Workshop on Web Quality*, pp. 42–49, ACM, 2011.

[85] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social phishing," *Communications of the ACM*, vol. 50, no. 10, pp. 94–100, 2007.

[86] D. Alyias, P. Henry, T. Rains, D. Batchelder, J. Jones, and V. Sekhar, "An in-depth perspective on software vulnerabilities and exploits, malware, potentially unwanted software, and malicious websites," report, 2012.

[87] H. Asghari, M. Ciere, and M. J. Van Eeten, "Post-mortem of a zombie: conficker cleanup after six years," in *24th USENIX Security Symposium (USENIX Security 15), Washington, DC*, 2015.

[88] C. Zhang, S. Zhou, and B. M. Chain, "Hybrid epidemicsa case study on computer worm conficker," 2015.

[89] W. MaxMind, "Ma, usa,maxmind: Open source ip address to country database,."

[90] F. Leder and T. Werner, "Know your enemy: Containing conficker," *The Honeynet Project, University of Bonn, Germany, Tech. Rep*, 2009.

[91] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocolhttp/1.1," 1999.

[92] H. Van de Sompel, M. L. Nelson, and R. Sanderson, "Http framework for time-based access to resource states–memento," 2012.

[93] J. Nylander, "Automatic timezone detection using javascript," 2010-12-23.

[94] S. J. Vaughan-Nichols, "Will mobile computing's future be location, location, location?," *Computer*, vol. 42, no. 2, pp. 14–17, 2009.

[95] A. Popescu, "Geolocation api specification," *World Wide Web Consortium, Candidate Recommendation CR-geolocation-API-20100907*, 2010.

[96] B. Pejic, A. Pejic, and Z. Covic, "Uses of w3c's geolocation api," in *Computational Intelligence and Informatics (CINTI), 2010 11th International Symposium on*, pp. 319–322, IEEE, 2010.

[97] V. N. Padmanabhan and L. Subramanian, "Determining the geographic location of internet hosts," in *SIGMETRICS/Performance*, pp. 324–325, 2001.

[98] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards ip geolocation using delay and topology measurements," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 71–84, ACM, 2006.

[99] V. N. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," in *ACM SIGCOMM Computer Communication Review*, vol. 31, pp. 173–185, ACM, 2001.

[100] M. J. Arif, S. Karunasekera, S. Kulkarni, A. Gunatilaka, and B. Ristic, "Internet host geolocation using maximum likelihood estimation technique," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 422–429, IEEE, 2010.

[101] D. N. Karthi Bojja and N. Reddy, "The internet host geo-location using delay and topology measures," 2015.

[102] P. Gill, Y. Ganjali, B. Wong, and D. Lie, "Dude, wheres that ip?: circumventing measurement-based ip geolocation," in *Proceedings of the 19th USENIX conference on Security*, pp. 16–16, USENIX Association, 2010.

[103] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: A comprehensive framework for the geolocalization of internet hosts," in *NSDI*, 2007.

[104] S. Siwpersad, B. Gueye, and S. Uhlig, *Assessing the geographic resolution of exhaustive tabulation for geolocating internet hosts*, pp. 11–20. Springer, 2008.

[105] J. Taylor, J. Devlin, and K. Curran, "Bringing location to ip addresses withip geolocation," *Journal of Emerging Technologies in Web Intelligence*, vol. 4, no. 3, pp. 273–277, 2012.

[106] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "Ip geolocation databases: Unreliable?," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 53–56, 2011.

[107] A. Dahnert, "Hawkeyes: An advanced ip geolocation approach: Ip geolocation using semantic and measurement based techniques," in *Cybersecurity Summit (WCS), 2011 Second Worldwide*, pp. 1–3, IEEE, 2011.

[108] ICANN, "Internet assigned numbers authority (iana) policy for allocation of ipv6 blocks to regional internet registries," 2006.

[109] ICANN, "Beginners guide to internet protocol (ip) addresses," report, 2011.

[110] J. A. Muir and P. van Oorschot, "Internet geolocation and evasion," *Apr*, vol. 10, pp. 1–22, 2006.

[111] ICANN, "Whois accuracy program specification," 2013.

[112] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," Report 2070-1721, 2005.

[113] P. T. Endo and D. F. H. Sadok, "Whois based geolocation: A strategy to geolocate internet hosts," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 408–413, IEEE, 2010.

[114] D. Moore, R. Periakaruppan, J. Donohoe, and K. Claffy, "Where in the world is netgeo. caida. org," in *INET 2000*, 2000.

[115] S.-H. Yook, H. Jeong, and A.-L. Barabsi, "Modeling the internet's large-scale topology," *Proceedings of the National Academy of Sciences*, vol. 99, no. 21, pp. 13382–13386, 2002.

[116] Y. Shavitt and N. Zilberman, "A geolocation databases study," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 10, pp. 2044–2056, 2011.

[117] K. TA, "Hazop and hazan: identifying and assessing process industry hazards," *Rugby, UK: Institution of Chemical Engineers*, pp. 1–223, 1999.

[118] N. G. Leveson, "Software safety in embedded computer systems," *Communications of the ACM*, vol. 34, no. 2, pp. 34–46, 1991.

[119] P. Baybutt, "A critique of the hazard and operability (hazop) study," *Journal of Loss Prevention in the Process Industries*, vol. 33, pp. 52–58, 2015.

[120] P. Roberts, "Failure modes, effects & criticality analysis," 2007.

[121] M. Mansoori, I. Welch, and Q. Fu, "Yalih, yet another low interaction honeyclient," 2014.

[122] H.-G. Kim, D.-J. Kim, S.-J. Cho, and M. Park, "Efficient detection of malicious web pages using high-interaction client honeypots," *Journal of Information Science and Engineering*, vol. 28, no. 5, pp. 911–924, 2012.

[123] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, "Rozzle: Decloaking internet malware," in *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 443–457, IEEE, 2012.

[124] Bitdefender, "Multi-language ransomware mentions police to enforce payment," 2012.

[125] SpiderLabsResearch, "A peek into the lion's den the magnitude [aka popads] exploit kit," 19-11-2014 2014.

[126] V. Kotov and F. Massacci, *Anatomy of Exploit Kits*, vol. 7781 of *Lecture Notes in Computer Science*, pp. 181–196. Springer, 2013.

[127] J. J. Lee, "Mechanize: Stateful programmatic web browsing in python," 2010.

[128] B. Feinstein and D. Peck, "Caffeinemonkey: Automated collection, detection and analysis of malicious javascript," in *Black Hat USA*, 2007.

[129] K. Heyman, "New attack tricks antivirus software," *Computer*, vol. 40, no. 5, pp. 18–20, 2007.

[130] Kaspersky, "Kaspersky security bulletin 2012. the overall statistics for 2012," report, 2013.

[131] V. M. Alvarez, "Yara in a nutshell," *yara-project: A Malware Identification and Classification Tool*, 2012.

[132] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, and K. Nakao, "Malware behavior analysis in isolated miniature network for revealing malware's network activity," in *ICC'08. IEEE International Conference on Communications*, pp. 1715–1721, IEEE, 2008.

[133] I. Corona, D. Maiorca, D. Ariu, and G. Giacinto, "Lux0r: Detection of malicious pdf-embedded javascript code through discriminant analysis of api references," in *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pp. 47–57, ACM, 2014.

[134] D. Stevens, "Malicious pdf documents explained," *Security & Privacy, IEEE*, vol. 9, no. 1, pp. 80–82, 2011.

[135] B. Hartstein, "Jsunpack: An automatic javascript unpacker," in *ShmooCon convention*, 2009.

[136] F. A. Manzano, "Adobe reader x bmp/rle heap corruption," report, Binamuse security research, 2012.

[137] J. M. Esparza, "Pdf attack: A journey from the exploit kit to the shellcode," in *Blackhat Asia Workshop proceedings*, 2014.

[138] M. Cova, A. Kapravelos, Y. Fratantonio, C. Kruegel, and G. Vigna, "Wepawet - detecting and analysis of web-based threats," 2010.

[139] M. Egele, E. Kirda, and C. Kruegel, "Mitigating drive-by download attacks: Challenges and open problems," in *In Proceedings of the Open Research Problems in Network Security*, p. 5262, INetSec, 2009.

[140] Y.-T. Hou, Y. Chang, T. Chen, C.-S. Laih, and C.-M. Chen, "Malicious web content detection by machine learning," *Expert Systems with Applications*, vol. 37, no. 1, pp. 55–60, 2010.

[141] V. Lam, I. Welch, X. Gao, and P. Komisarczuk, "Anatomy of drive-by download attack," in *Australasian Information Security Conference*, 2013.

[142] N. Provos and T. Holz, *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addisson-Wesley, 2008.

[143] G. Auberger, "Secure cross domain iframe communication," 2011.

[144] C. Kanich, N. Chachra, D. McCoy, C. Grier, D. Y. Wang, M. Motoyama, K. Levchenko, S. Savage, and G. M. Voelker, "No plan survives contact: Experience with cybercrime measurement," in *CSET*, 2011.

[145] J.-L. Lin, "Detection of cloaked web spam by using tag-based methods," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7493–7499, 2009.

[146] A. Kapravelos, M. Cova, C. Kruegel, and G. Vigna, *Escape from monkey island: Evading high-interaction honeyclients*, pp. 124–143. Springer, 2011.

[147] F. Howard and O. Komili, "Poisoned search results: How hackers have automated search engine poisoning attacks to distribute malware," *Sophos Technical Papers*, 2010.

[148] A. Agresti, *Categorical data analysis*, vol. 359. John Wiley & Sons, 3rd edition ed., 2013.

[149] A. K. Sood, R. J. Enbody, and R. Bansal, "Malware analysis - styx exploit pack: Insidious design," report, 2013.

[150] G. Wang, J. W. Stokes, C. Herley, and D. Felstead, "Detecting malicious landing pages in malware distribution networks," in *43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–11, IEEE, 2013.

[151] Z. Gyongyi and H. Garcia-Molina, "Web spam taxonomy," in *First international workshop on adversarial information retrieval on the web (AIRWeb 2005)*, 2005.

[152] B. Wu and B. D. Davison, "Cloaking and redirection: A preliminary study," in *First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pp. 7–16, 2005.

[153] J. Zhuge, T. Holz, C. Song, J. Guo, X. Han, and W. Zou, *Studying malicious websites and the underground economy on the Chinese web*. Springer, 2009.

[154] D. Mahjoub, "Details on exploit kits, as told by the umbrella security graph," 2013.

[155] SpiderlabsResearch, "Rig reloaded - examining the architecture of rig exploit kit 3.0," 2015.

[156] V. Lam, I. Welch, X. Gao, and P. Komisarczuk, "Identification of potential malicious web pages," in *Australasian Information Security Conference (AISC)*, 2011.

[157] Maxmind, "Industry leading ip intelligence," 2002.

[158] N. Champion, "At what cost? it pricing and the australia tax," report, The Parliament of the Commonwealth of Australia, July 2013.

[159] B. Eshete and V. Venkatakrishnan, "Webwinnow: leveraging exploit kit workflows to detect malicious urls," in *Proceedings of the 4th ACM conference on Data and application security and privacy*, pp. 305–312, ACM, 2014.

[160] M. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos, and L. Schmidt, "Trends in circumventing web-malware detection," report, 2011.

[161] "hphosts - freely downloadable community managed hosts file for ad and malware site blocking."

[162] J. Nazario and T. Holz, "As the net churns: Fast-flux botnet observations," in *3rd International Conference on Malicious and Unwanted Software*, pp. 24–31, IEEE, 2008.

[163] C. V. Zhou, C. Leckie, and S. Karunasekera, "Collaborative detection of fast flux phishing domains," *Journal of Networks*, vol. 4, no. 1, pp. 75–84, 2009.

[164] B. Eriksson, P. Barford, B. Maggs, and R. Nowak, "Posit: a lightweight approach for ip geolocation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 2, pp. 2–11, 2012.

[165] T. J. McLaughlin, "The benefits and drawbacks of http compression," *Department of Computer Science and Engineering, Lehigh University*, vol. 19, 2002.

[166] Z. Nagy, "Improved speed on intelligent web sites," *Recent Advances in Computer Science, Rhodes Island, Greece*, pp. 215–220, 2013.

[167] E. Law, "Compressing the web," 2014.

[168] B. Hoffman, "Lose the wait: Http compression," 2012.

[169] S. Kovalchuk, "Html compressor and minifier," 2012.

[170] S. K. Bajaj and J. Pieprzyk, "Can we can the email spam," in *Cybercrime and Trustworthy Computing Workshop (CTC), 2013 Fourth*, pp. 36–43, IEEE, 2013.

[171] R. B. Basnet and A. H. Sung, "Classifying phishing emails using confidence-weighted linear classifiers," in *International Conference on Information Security and Artificial Intelligence (ISAI)*, pp. 108–112, 2010.

[172] C. Baraniuk, "Spike in brexit email spam following referendum result," 2016.

[173] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," *arXiv preprint arXiv:1609.03020*, 2016.

[174] L. R. A. Alonso-Prrizas, "Neutrino exploit kit: An alysis and threat indicators," 2016.

Abbreviations

AES: Automated Experimentation System
AIRS: Adversarial Information Retrieval System
AJAX: Asynchronous JavaScript and XML
AP: Address Prefix
API: Application Program Interface
APNIC: Asia Pacific Network Information Centre
ARIN: American Registry for Internet Numbers
ASN: Autonomous System Number
ASCII: American Standard Code for Information Interchange
BEP: Browser Exploit Pack
BGP: Border Gateway Protocol
CAIDA: Centre for Applied Internet Data Analysis
CBG: Constraint Based Geolocation
ccTLD: Country Code Top Level Domain
CDMA: Code Division Multiple Access
CI: Confidence Interval
CPU: Central Processing Unit
CSS: Cascading Style Sheets
CVE: Common Vulnerabilities and Exposures
DDOS: Distributed Denial Of Service
DOM: Document Object Model
DOS: Denial Of Service
DNS: Domain Name System
FMEA: Failure Mode and Effects Analysis
GDP: Gross Domestic Product
GEE: Generalizes Estimation Equation
GPS: Global Positioning System
GSM: Global System for Mobile Communications

gTLD: Generic Top Level Domain
HAZOP: Hazard and Operability
HTML: Hyper-Text Markup Language
HTML5: Hyper-Text Markup Language version 5
HTTP: Hyper-Text Transfer Protocol
HTTPS: Hyper-Text Transfer Protocol Secure
IANA: Internet Assigned Numbers Authority
ICANN: Internet Corporation for Assigned Names and Numbers
ICMP: The Internet Control Message Protocol
IGP: Interior Gateway Protocol
IMAP: Internet Message Access Protocol
IP: Internet Protocol
IPv4: Internet Protocol Version 4
IPv6: Internet Protocol Version 6
ISP: Internet Service Provider
JPEG: Joint Photographic Experts Group
LACNIC: Latin America and Caribbean Network Info. Centre
LICH: Low Interaction Client Honeypot
LIR: Local Internet Registry
LOPA: Layer of Protection Analysis
MAC: Media Access Control
NIR: National Internet Registry
NRO: Number Resource Organisation
OS: Operating System
P2P: Peer-to-Peer
PCRE: Perl Compatible Regular Expression
PDF: Portable Document Format
PID: Process ID
POP3: Post Office Protocol 3
RIR: Regional Internet Registries
RFID: Radio Frequency Identification

RPC: Remote Procedure Call
ROI: Return on Investment
SEO: Search Engine Optimisation
SMTP: Simple Mail Transfer Protocol
SSL: Secure Sockets Layer
TBG: Topology-aware Geolocation
TCP: Transmission Control Protocol
TLD: Top Level Domain
TLS: Transport Layer Security
TTL: Time-To-Live
UAE: United Arab Emirates
UDP: User Datagram Protocol
UI: User Interface
UML: Unified Modeling Language
URL: Uniform Resource Locator
UK: United Kingdom
US: United States
UTC: Coordinated Universal Time
VB: Visual Basic
VPN: Virtual Private Network
VPS: Virtual Private Server
VUW: Victoria University of Wellington
XML: eXtensible Markup Language
YALIH: Yet Another Low Interaction Honeyclient

Glossary

Adware - The term adware is short for "advertising-supported software." It refers to any piece of software or application that displays advertisements, usually through pop-up or pop-under windows.

AJAX - Asynchronous JavaScript and XML is a technique that web pages use to have the server perform certain processing without reloading the web page.

API - Application Program Interface is a set of commands, functions, and protocols which programmers can use when building software for a specific operating system. The API allows programmers to use predefined functions to interact with the operating system, instead of writing them from scratch.

ASCII - American Standard Code for Information Interchange is a standard but limited character set containing only English letters, numbers, a few common symbols, and common English punctuation marks.

BGP - Border Gateway Protocol is a system routing protocol for routing information on the Internet (versus internal networks). It is a highly scalable and robust protocol that uses route parameters to define routing policies and maintain a stable routing environment.

Botnet - Bot network is a network of hijacked zombie computers controlled remotely by a hacker. The hacker uses the network to send spam and launch Denial of Service attacks, and may rent the network out to other cybercriminals.

Client - Any program that uses the service of another program. On the Web, a Web client is a program, such as a browser, editor, or search robot, that reads or writes information on the Web.

Cookies - Cookies are text files that are created when users visit web sites. Cookies are used to tell the server that users have returned to a particular web site and provides information used to show targeted content.

Crimeware - Crimeware is a general term for software used to perpetrate crime, such as stealing personal identities, money or proprietary informa-

tion. Crimeware can spread by way of viruses, Trojan horse programs, worms, spyware, or adware.

XSS - Cross-site Scripting is a security vulnerability usually found in websites and/or web applications that accept user input. Examples of these include search engines, login forms, message boards and comment boxes. Cybercriminals exploit this vulnerability by inputting strings of executable malicious code into these functions.

DDoS - A Distributed Denial of Service describes a scenario where servers get overwhelmed by multiple queries from distributed quartiers, generally from bots, with the intention of overwhelming the service and making it unavailable to answer legitimate queries.

DNS - The Domain Name System is the Internet standard for assigning IP addresses to domain names.

Encryption - Encryption is the process of converting data into a form that cannot easily be read without knowledge of the conversion mechanism (often called a key).

Exploit - An exploit is code that takes advantage of a software vulnerability or security flaw. Exploits are often incorporated into malware, which are consequently able to propagate into and run intricate routines on vulnerable computers.

Exploit kits - refer to a type of hacking toolkit that cybercriminals use to take advantage of vulnerabilities in systems/devices so they can distribute malware or do other malicious activities. They normally target popular software such as Adobe Flash ®, Java™, and Microsoft Silverlight.

Heuristics - Heuristics is a scanning method that looks for malware-like behavior patterns. It is commonly used to detect new or not-yet-known malware.

HTML - Hyper-Text Markup Language is the language that Web pages are written in. Also known as hypertext documents, Web pages must conform to the rules of HTML in order to be displayed correctly in a Web browser. The HTML syntax is based on a list of tags that describe the page's format and what is displayed on the Web page.

HTTP - Hyper-Text Transfer Protocol (HTTP) is used to transfer informa-

tion, such as HTML documents, on the Internet.

HTTPS - Hyper-Text Transfer Protocol Secure (HTTPS) is a variation of HTTP that uses the Secure Socket Layer to increase security.

IANA - Internet Assigned Numbers Authority, the authority originally responsible for the oversight of IP address allocation, the coordination of the assignment of protocol parameters provided for within Internet technical standards, and the management of the domain name system, including the delegation of top-level domains and oversight of the root name server system.

ICANN - The Internet Corporation for Assigned Names and Numbers (ICANN) is an internationally organised, non-profit corporation that has responsibility for Internet Protocol (IP) address space allocation, protocol identifier assignment, generic (gTLD) and country code (ccTLD) Top-Level Domain name system management, and root server system management functions.

IGP - Internet Gateway Protocol is distributes routing information to the routers within an autonomous system.

IP - Internet Protocol governs how computers send packets across the Internet.

IPv4 - Internet Protocol V4 is the system that routes most traffic on the Internet. It is a connectionless protocol for packet switched networks based on best-effort delivery.

IPv6 - Internet Protocol V6 is the latest version of the Internet Protocol that provides the location system for devices on the Internet. It uses a 128-bit address to allow a virtually limitless number of addresses (2^128)

Java - A programming language developed (originally as "Oak") by James Gosling of Sun Microsystems. Designed for portability and usability embedded in small devices, Java took off as a language for small applications ("applets") that ran within a Web browser.

JavaScript - JavaScript is a scripting language used for client-side web development. It is used in millions of webpages to add functionality, validate forms and detect browsers.

LIR - Local Internet Registry (IR) primarily assign address space to the users of the network services they provides. LIRs are generally ISPs, whose customers are primarily end users and possibly other ISPs.

Malware - Malware is a general category of malicious code that includes viruses, worms and Trojan horse programs.

Obfuscation - Obfuscation refers to the process of concealing something important, valuable, or critical. Cybercriminals use obfuscation to conceal information such as files to be downloaded, sites to be visited, etc.

Payload - A payload is the action that a threat performs, apart from its main behavior. Payloads can range from stealing personal information to deleting the contents of a hard drive.

PDF - Portable Document Format is a multi-platform file format developed by Adobe Systems. A PDF file captures document text, fonts, images, and even formatting of documents from a variety of applications.

Polymorphic virus - Polymorphic viruses are complex file infectors that change physical forms, yet retain the same basic routines, after every infection.

Proxy server - A proxy server is a server that sits between a client and a web server and represents itself to each end as being the other.

Ransomware - A type of malware that prevents or limits users from accessing their system, either by locking the system's screen or by locking the users' files unless a ransom is paid.

Registrar - Domain names can be registered through many different companies (known as "registrars") that compete with one another.

SandBox - A sandbox is a restricted environment that contains the actions and effects of the sandboxed software within it.

Script - Scripts are written code that are interpreted and implemented by another application. Malware authors have taken advantage of relative ease of producing scripts and have produced significant numbers of script-based malware.

SEO - Search Engine Optimisation is the process of improving the volume and quality of traffic to a web site from search engines via "natural" ("or-

ganic" or "algorithmic") search results.

Spam - Spam is unwanted and unsolicited bulk email. Over 95% of all email today is spam.

Spear phishing - Spear phishing is a targeted form of phishing in which fraudulent emails target specific organisations in an effort to gain access to confidential information.

Spyware - Spyware is a program that monitors and gathers personal information and sends to a third party without the user's knowledge or consent.

SQL Injection - SQL injections are one of the most common code injection techniques used by attackers to attack web sites. An attacker finds a vulnerability in the target web site SQL-based application software.

SSH - SSH stands for Secure Shell. It is a communication protocol for connecting to remote computers over TCP/IP. Various authentication methods can be used which make SSH more secure than Telnet.

Subnet Mask - A subnet mask is a number that defines a range of IP addresses that can be used in a network. Subnet masks are used to designate subnetworks, or subnets, which are typically local networks LANs that are connected to the Internet. Systems within the same subnet can communicate directly with each other, while systems on different subnets must communicate through a router.

TCP - Transmission Control Protocol is a computer protocol that allows one computer to send the other a continuous stream of information by breaking it into packets and reassembling it at the other end, resending any packets that get lost in the Internet. TCP uses IP to send the packets, and the two together are referred to as TCP/IP.

TLD - Top Level Domains are the names at the top of the DNS naming hierarchy. They appear in domain names as the string of letters following the last (rightmost) ".", such as "net" in "www.example.net".

Topology - The allowable connectivity between nodes, anchors and links: for example, 1-1 or many-1 mappings.

Trigger - A trigger is a system condition or date that sets off the payload of

a specific threat. A trigger condition can be anything from the presence of a certain file or a specific user action, such as the clicking of certain buttons.

Unicode - A widely supported and preferred character encoding system.

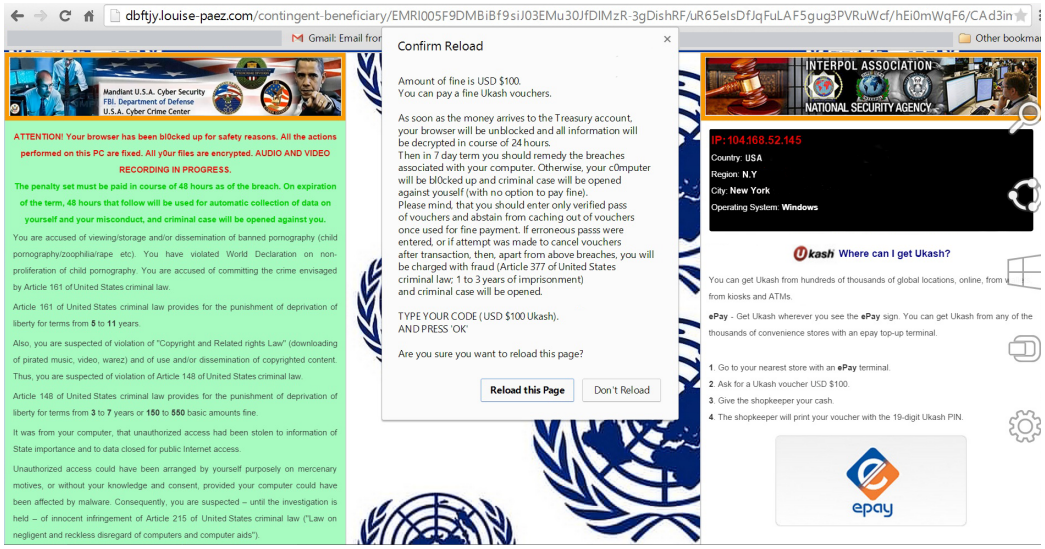
Vulnerability - A vulnerability is a security weakness typically found in programs and operating systems that leaves computing systems open to malware and hacker attacks.

WHOIS - WHOIS protocol (pronounced ”who is”; not an acronym) An Internet protocol that is used to query databases to obtain information about the registration of a domain name (or IP address).

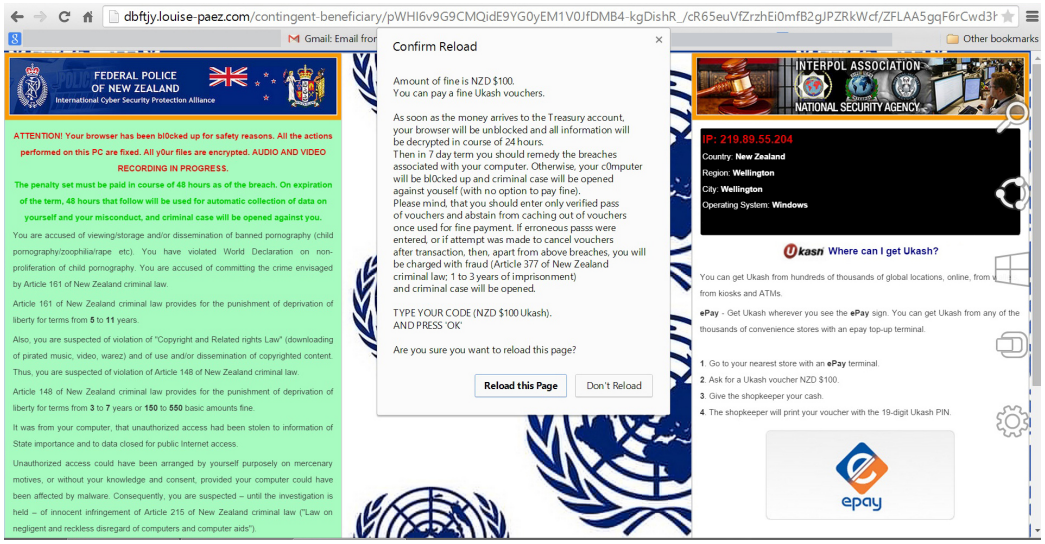
Worm - A computer worm is a self-contained program (or set of programs) that is able to spread functional copies of itself or its segments to other computer systems. The propagation usually takes place via network connections or email attachments.

Zero-day exploits - Zero-day exploits refer to software vulnerabilities that have been found in-the-wild before security researchers and software developers become aware of the threat. Because of this, they pose a higher risk to users than other vulnerabilities.

Appendices



Geotargeting web site retrieved from United States



Geotargeting web site retrieved from New Zealand

```
http://hosts-file.net
https://easylist-downloads.adblockplus.org
https://zeustracker.abuse.ch
http://malwaredomains.lehigh.edu/files/justdomains
http://osint.bambenekconsulting.com/feeds
http://mirror1.malwaredomains.com/updates
http://vxvault.siri-urz.net/ViriList.php
http://cybercrime-tracker.net
http://www.malwareblacklist.com/showMDL.php
```

List of publicly available malicious domains

```
#! /usr/bin/env python
>>> from tld import get_tld
>>> URL = get_tld("http://example.web site.co.nz", fail_silently=
    True as_object=True)

>>>print URL.tld
web site.co.nz
>>> Print URL.suffix
co.nz
```

Extraction of Top Level Domain (TLD) information from a URL through tld library in Python

```
>>> w=whois.whois("http://www.drugzdravia.ru/")
>>> w.country
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'WhoisRu' object has no attribute 'country'

>>> w
{'status': 'REGISTERED, DELEGATED, VERIFIED', 'expiration_date':
    datetime.datetime(2016, 3, 31, 0, 0), 'domain_name': '
    DRUGZDRAVIA.RU', 'creation_date': datetime.datetime(2012, 3,
    31, 0, 0), 'registrar': 'REGTIME-RU', 'name_servers': ['ns5.
    ukrdomain.net.', 'ns6.ukrdomain.net.'], 'org': None, 'emails
    ': None}

>>> w.registrar
REGTIME-RU
```

Extraction of Top Level Domain (TLD) information from a URL through Registrar attributes of WHOIS data

```
from langdetect import detect
import mechanize

>>> br = mechanize.Browser()
>>> r = br.open(http://www.hemayat.net/, timeout=12.0)
>>> response = br.response().read()
>>> text = unicode(response, errors="ignore")
>>> detect(text)

"fa"
```

Detection of a web site language through Google's language detection library

```
import mechanize

br = mechanize.Browser()
  cj = cookielib.LWPCookieJar()
  br.set_cookiejar(cj)
  br.set_handle_equiv(True)
  br.set_handle_gzip(True)
  br.set_handle_redirect(True)
  br.set_handle_referer(False)
  br.set_handle_robots(True)
  br.set_debug_responses(True)
  br.set_debug_redirects(True)
  br.set_handle_refresh(mechanize._http.HTTPRefreshProcessor(),
     max_time=0)
  br.set_proxies(config.proxy)
  br.encoding = "UTF-8"
```

List of several features and functionalities supported by Mechanize virtual browser

```
browser.open(link, timeout=12.0)
response = browser.response().read()

# .js extraction using Regular Expression library
response.compile("\.js(\?)?.*$")

# Hyperlink extraction using Beautifulsoup library
soup = BeautifulSoup(response)
for tag in soup.findAll('a', href=True):

# Script extraction using python-lxml library
document = lxml.html.document_fromstring(response)
script_list = document.xpath('//script/@src')
```

Example of code illustrating Component extraction using Regular Expression, BeautifulSoup and python-lxml Library

```
import BeautifulSoup

def strip_content(input_soup):
   if input_soup.name == 'script':
      return input_soup
//strip content from all children
children = [strip_content(child) for child in input_soup.children
    if not isinstance(child, NavigableString)]
//remove everything from the tag
   input_soup.clear()
   for child in children:
      input_soup.append(child)
   return input_soup

def feat_remove(filename):
   soup = BeautifulSoup(open(filename))
   output = strip_content(soup)
   print(output.encode('utf8'))
```

Removing all the text from an HTML file with BeautifulSoup, while keeping the structure of the web site intact

```
import lxml.html
import lxml.html.clean as clean

cleaned=clean.Cleaner(scripts=False,page_structure=True,
    javascript=False, embedded=False, remove_unknown_tags=False,
    frames=False)

print cleaned.clean_html(s)
```

Python HTML parser library, lxml allows for removal or extraction of individual attributes

```
//Minification
command = "java -jar htmlcompressor-1.5.3.jar input_file
    output_file"
os.system(command)

//Compression
Input_file = open("inputfile or stream", 'read binary')
Output_file = gzip.open("Outputfile.html.gz", 'write binary')
Output_file.writelines(Input_file)
Output_file.close()
Input_file.close()
```

Minification and compression of a retrieved web site content using "html-compressor" and gzip library in Python

```
#!/bin/bash
script_dir=$(dirname $0)
until $script_dir/honeypot.py --file $1; do
   output="$(/bin/date '+%x %X')  exit code $?, restarting
      honeypot.py."
   echo -e "$output" >&2 >> crash.log
   python $ script_dir/honeypot.py --file $2
   sleep 5
done
```

BASH script to check the status of our honeypot process and restart it if crashed

```
def checkMonitor():
#Start the timer
threading.Timer(config.checkfrequency,checkMonitor).start()
  #check if the process is running
if not p.is_running():
  #start the process
  process = subprocess.Popen(['python','honeypot.py']+sys.argv
      [1:])
processID = int(process.pid)
  p = psutil.Process(processID)

#creating and running the process
Global process, processID
process = subprocess.Popen(['python','honeypot.py']+sys.argv[1:])
#assigning the process ID
processID = int(process.pid)
p = psutil.Process(processID)
checkMonitor()
```

Checking the status of a process through psutil utility and its associated PID

```
#Check for running process and network activity by the process
p = psutil.Process(processID)
if p.is_running():
  print p.connections()

  #No network connections being made, restart honeypot.py
  if str(p.connections()) == "[]":
    process.kill()
process=subprocess.Popen(['python','honeypot.py']+sys.argv[1:])
    processID = int(process.pid)

    #notification to user admin
sendEmail("Yalih Status Update","Restarting the process.")
    checkMonitor()
```

Monitoring the state of a process through its network bandwidth utilisation

```
rule Possible_Shellcode_Pattern
{
strings:
$a1 = /=\s*?unescape\(\s*?\n?\s*["'](%u[a-fA-F0-9]{4}|%[a-fA-F0
    -9]{2}){2,}['"]\s*?[\+\)]/ nocase

$b1 = "unescape" fullword nocase
$b2 = "%u0A0A" nocase
$b3 = "%u9090"
$shellcode = /(%u[A-Fa-f0-9]{4}){8}/
$c1 = /document\.write\(unescape\(\s*?\n?\s*["'](%u[a-fA-F0
    -9]{4}|%[a-fA-F0-9]{2}){2,}['"]/ nocase

condition:
$a1 or ($b1 and ($b2 or $b3)) or ($b1 and $shellcode) or $c1
}
```

Yara custom rule to detect a possible Shellcode pattern in a retrieved web site content

```
rule PossibleIFrame
{
 meta:
 description = "Possible malicious link redirection"
 strings:
 $a1 = /ini\.php['"]\s*?width=['"]0['"]\s*?height=['"]0["']\s*?
     frameborder=['"]0['"]><\/iframe>/
 condition:
 $a1
}
```

Yara custom rule to detect a possible malicious link redirection in a retrieved web site content

```
rule SuspicousBodyOnload
{
 meta:
 impact = 6
 strings:
 $body = /<body [^>]*onload\s*=\s*['"]*[a-z0-9]+\(['"][a-f0
    -9]{300}/ nocase
 $a1 = /ini\.php['"]\s*?width=['"]0['"]\s*?height=['"]0["']\s*?
    frameborder=['"]0['"]><\/iframe>/
 $b1 = "unescape" fullword nocase
 condition:
 ($body or $a1) and ($a1 > 5 and $b1)
```

Yara custom rule to detect suspicious Body Onload JavaScript code in a retrieved web site content

```
rule generic_javascript_obfuscation
{
meta:
author = "Josh Berry"
description = "JavaScript Obfuscation Detection"
sample_filetype = "js-html"
strings:
$string0 = /eval\(([\s]+)?(unescape|atob)\(/ nocase
$string1 = /var([\s]+)?([a-zA-Z_$])+([a-zA-Z0-9_$]+)?([\s]+)?=([\
   s]+)?\[([\s]+)?\"\\x[0-9a-fA-F]+/ nocase
$string2 = /var([\s]+)?([a-zA-Z_$])+([a-zA-Z0-9_$]+)?([\s]+)?=([\
   s]+)?eval;/
condition:
any of them
}
```

Yara custom rule to detect a possible JavaScript Obfuscation in a retrieved web site content