Traffic Classification for the Detection of Anonymous Web Proxy Routing

Shane Miller, Kevin Curran, Tom Lunney
Ulster University, Northern Ireland

Abstract

There is an increasing need to be able to classify whether an incoming packet is from a legitimate originating IP address or has been modified through an intermediate proxy or node. Being able to verify the originating IP address allows a business (e.g. bank) to use geolocation services in order to then ascertain which geographical location that packet was sent from. This can then feed into the system intrusion system or backend fraud alert mechanisms. The web however is going 'dark'. There is a noticeable uptake in the amount of encrypted data and third party anonymous traffic proxies which aim to mask the try location and IP address of a web request. We present here a system which identifies the characteristics or signatures whenever a user is using a web proxy by developing a Detection System that records packets and analyses them looking for identifying patterns of web proxies.

1. Introduction

A proxy server, in terms of computer networks, is a server that acts as an intermediary for requests from clients for resources located on other servers on a network or the Internet. This is the most basic type of proxy which is known as a gateway. Another type of proxy is a reverse proxy. This consists of a server on an internal company network and acts as an intermediary for other servers based on that network. Reverse proxies are typically used as an Internet facing server that handles a number of different tasks. Some examples include: SSL acceleration using specially designed hardware for the encryption and decryption of SSL traffic, load balancing to distribute requests between several web servers and acting as a cache for static content such as pictures and other graphical content. The proxies that will be discussed in this research are anonymising proxies which are based on another type of proxy known as an open proxy. Open proxies are a proxy that is available to any user on the Internet. They are mostly used to set up anonymous proxy websites. Anonymising proxy sites act as an intermediary, forwarding requests and fetching the results, whilst also hiding a user's identity by concealing their IP address from web servers on the Internet. This type of server is regularly used as a means to hide a criminal's identity so they can commit various crimes on the internet without being caught. There are also a number of risks with using an anonymous proxy as a method to bypass network filters on a company network. The anonymous proxy server might not be a simple intermediary that only forwards requests and fetches the results. It could also be logging all the requests and information that pass through it. This information could include usernames and passwords and the operators of the proxy site may use these to steal the identity linked to the credentials and use it to commit fraud and other criminal actions. A user employing an anonymous proxy on an enterprise network to bypass a network filter might be, unwittingly, leaking confidential information about their company. To combat this issue we propose a system that will detect suspicious traffic on the network and attempt to determine whether the traffic indicates the usage of an anonymous proxy website. The system will specifically check for characteristics that appear in packets generated by anonymous proxies and then create rules to determine the usage of anonymous proxies.

2. Intrusion Detection and Traffic Classification

2.1. IP Blocking

IP blocking is the most basic technique used to combat malicious threats to networks and is one of the most commonly used techniques for protecting networks [1]. Using this method an administrator can block an IP address or a range of IP addresses from accessing a certain domain name IP address. A network administrator can also block access to an IP address that is being used by a disruptive user. IP blocking can however be overcome by using anonymising proxies. The user's IP address is usually sent out as a source IP address in the network packet containing the request to a web server. However, when using a proxy, this request is first sent to the proxy server which then forwards it on towards the web server. This forwarded request is encased in a new network packet which means that the source IP address is no longer that of the end user but instead is that of the proxy server. So, the blocked IP address of the user is not actually making any direct contact with the web server running the IP filter. The network administrator may also block the IP addresses of websites that they do not want users to access, but in this case a proxy will separate the business network and the website being accessed. The IP block filter will only detect the IP of the proxy site, which will likely not be in the filter's block list. Fig. 1 shows how the proxy is located between the user and the website they are trying to access.

2.2. Firewalls and Intrusion Detection

A significant security problem for business type networks is hostile or unwanted access by users or software [2]. Unwanted user access (an intrusion) can be in the form of unauthorised logon to a machine or gaining the ability to perform higher privilege actions than what is normally authorised. Unwanted software access can take the form of a virus, Trojan horse or other form of malware. To combat these intrusions there are a number of defences. There are host based security methods that are managed by the operating system of the machine, various types of firewall used to filter network packets, such as Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). A firewall is defined as a component or set of components that restrict access between a protected network and external networks [3]. Intrusion Detection Systems detect intrusions on a network. IDSs come in many different configurations, two of which are Host-based IDS (HIDS) and Network-based IDS (NIDS). The difference between these two is the location of the IDS on the network. A HIDS monitors and collects the characteristics for hosts containing sensitive information, servers running public services and suspicious activities [4]. To detect intrusions to the network HIDSs typically follow one of two general approaches. These are anomaly detection and signature detection. Anomaly detection involves the collection of data relating to behaviour of legitimate users over a period of time. Next, tests are applied to observed behaviour to determine if it involves an illegitimate user. Signature detection involves a set of rules or attack patterns that can be used to decide if an observed behaviour is that of an attacker [4]. A NIDS captures network traffic at specific points of a network through sensors and then analyses the activities of applications and protocols to recognise suspicious incidents [4]. A typical NIDS configuration includes a number of sensors to monitor network traffic, a NIDS management server and one or more user interface consoles for human interaction with the IDS. The analysis of network traffic may occur at either the sensor and/or the management server. As with HIDSs, NIDSs make use of both anomaly detection and signature detection [4].

2.3. Intrusion Detection and Prevention Systems

Intrusion detection is the process of monitoring connections coming to and leaving from a computer or network and then analysing those connections for signs of potential violations or incidents that go against security and acceptable use policies [5].Causes of these incidents can include attackers gaining unauthorised access to systems, malware such as spyware and Trojan viruses and misuse of system privileges by users or attempts to gain additional privileges. An intrusion detection system is the software that automates this process. An intrusion prevention system has all the same capabilities of an intrusion detection system and also has the capability of preventing possible violations [6]. When detecting possible incidents, an IDS can take a number of actions. One would be to report the incident to a system security administrator, who could then initiate a response to mitigate the effects of the incident. Alongside alerting an administrator, the IDS could also keep a record of incident that could be referenced at a later date and as a way to help prevent future cases of that particular incident. There are a number of different types of IDS. These are: Network based, Host based, Network Behaviour and Wireless [5]. Network based systems monitor the traffic of a network using sensors placed at certain parts of the network and IDS management servers. They analyse the activity recorded by the sensors in order to identify incidents of intrusion. Host based systems differ from network based systems by monitoring a single host. NBA systems monitor network traffic in order to identity threats that generate unusual traffic flows such as malware or port scanning attempts. Wireless IDSs apply similar techniques to network based systems specifically to wireless network traffic that makes use of wireless networking protocols. IDSs typically use 3 primary detection methodologies; signature based detection, anomaly based detection and stateful protocol analysis [7, 4]. IDSs can make use of only one of these methods or, more commonly, they can make use of multiple methods which provides a broader and more complete approach to intrusion detection. Signature based detection is the process of using signatures to define what is and is not a potential incident. Signatures are defined as a pattern or string that signifies a known attack or threat [4]. An example of a signature would be more than 3 consecutive failed logins within 2 minutes signifying an attack attempt. Signature based detection is the simplest methodology available to IDSs as it compares the current network packets or network logs against a list of signatures and patterns using string comparison techniques[8]. Scanning network packets would be useful for an online, real time detection system whereas scanning network logs would be more suitable in finding out if an attack had taken place in the past. A limitation of signature based detection is that they are limited to the information that they were trained on. This means that new types of attack or an attacks that use intrusion detection evasion techniques may not be identified [4, 9]. If a certain type of malware has a signature that is based on the name of a file included as an attachment in an email, then a simple way to evade signature detection is to change the name of the file.

Anomaly based detection is the process of comparing the known behaviours of the network against observed events in the same network to identify significant deviations. An anomaly is defined as a deviation to a known or normal behaviour. Profiles are used to represent the normal or expected behaviours derived [4]. These profiles are typically generated over a period of days or weeks and will contain statistical data of system activities such as CPU usage. The profile that results can then be either a static profile of the system or network or a dynamic profile. Static profiles remain unchanged until there is

a need to change them, in which case the system is instructed to generate a new profile of the system or network. A dynamic profile is updated regularly as new events are observed on the network [5]. Anomaly based detection tends to be effective in detecting new types of incidents or attacks but there are a number of drawbacks as well. For an accurate normal profile to be generated, a lot of time must be invested otherwise the profile will be inaccurate and will trigger false alerts [10]. These will mostly be in the form of false positives where the system reports an incident that is part of the legitimate network activity. This can lead on to the system mistaking actual incidents for normal activity, also known as false negatives. Stateful protocol analysis is similar to anomaly based detection in that it compares predetermined profiles against observed network activity to identify deviations. The profiles used are where there are differences. Unlike in anomaly based detection, stateful protocol analysis makes use of universal profiles that define how particular protocols are expected to behave in normal everyday operations [5]. These profiles are based primarily on protocol standard developed by software development companies and standards bodies. By comparing observed traffic against these profiles, stateful protocol analysis can identify unexpected utilisations of protocols that may indicate an attack. However, companies regularly modify protocols to suit their own needs and sometimes fail to document or detail the changes made. This makes it difficult for stateful protocol analysis IDSs to perform a complete analysis unless the protocol profile is updated. The main disadvantage to stateful protocol analysis though, is that it is a very resource intensive task because of the complexity of the analysis performed and the process of tracking the state of a protocol [5]. To combat the deficiencies of the different detection methods, most IDSs make use of multiple methods. Signature based and Anomaly based detection are complementary methods as they both address the short-comings of the other [4].

2.4. Intrusion Detection Systems with Machine Learning

Integrating machine learning techniques into intrusion detection systems is seen as a way to increase the ability and accuracy of the detection system. These techniques include various kinds of artificial neural networks and classification techniques such as genetic algorithms and fuzzy logic. There has been various research studies looking into integrating machine learning into IDSs with the recent trend being in improving the machine learning aspect by combining different techniques to increase detection accuracy and to decrease the computational effort required to train the systems. [8] proposed a feature representation technique using a combination of the cluster centre and nearest neighbour approaches. Experiments that were carried out made use of the KDD-Cup99[1] dataset and showed that the approach required less computational effort to provide similar levels of accuracy to k-NN. [11] proposed a multiple level hybrid classifier that combined supervised tree classifiers with unsupervised Bayesian clustering. Performance of this approach was also measured using the KDD-Cup99 dataset and experiments showed that it provided a low false negative rate of 3.23% and a false positive rate of 3.2% with a high detection rate for both known and unknown attacks. [12] made use of a Support Vector Machine (SVM) for classification and a clustering tree technique called Dynamically Growing Self-Organising Tree (DGSOT) to improve the training times of the SVM. Experiments were carried out using the DARPA98[2] dataset and showed that using a clustering tree helped to increase the accuracy rate of the SVM and lower the rates of false positives and false negatives. [13] provided a system that made use of both genetic algorithms and fuzzy logic to create a genetic fuzzy classifier to predict different behaviours in networked computers. Their results showed that there was a benefit to using fuzzy logic to pre-screen rules before classifying with the genetic algorithm as it decreased the time needed to train the system. However the systems accuracy in detection did not show much increase and actually showed a decrease in accuracy in some classes compared to other approaches. An earlier study used 3 different anomaly detection techniques for classifying program behaviour [14]. These techniques were an equality matching algorithm for determining what was and wasn't anomalous behaviour, a feed forward backpropagation neural network for learning the program behaviour and the third being a recurrent neural network called an Elman network for recognising recurrent features of program behaviour. Their study showed that the performance of intrusion detection benefited greatly from the use of the backpropagation network and the Elman network. The general consensus that can be gathered from these studies is that the use of machine learning techniques does improve the accuracy and performance of intrusion detection systems.

2.5 Proxies

Anonymous web proxies come in many different forms. Some proxy scripts are produced using PHP based or CGI (Common Gateway Interface) based scripts. The reasoning behind the use of these technologies is that they both provide the functionality that an anonymous proxy requires and they are compatible with both UNIX-like and Windows hosts. To access the anonymous proxy a user client needs to connect to the proxy server first. From there, they are then able to send a request to the website anonymously. The proxy script takes the clients request and issues its own request to the destination website, receives the data back and forwards it on to the client. This is shown in Fig. 1.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2] https://www.ll.mit.edu/ideval/data/

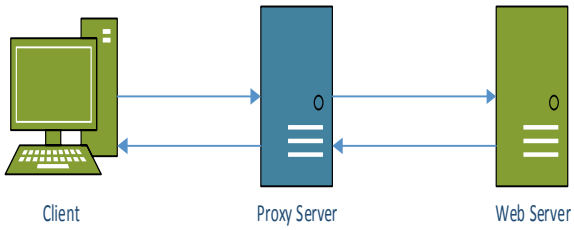

Figure 1. Proxy connection

Glype is a PHP based script and is one of the most common and popular web proxy scripts available on the internet. This is due to its support for content like JavaScript and to its ease of set up and use. To set up a Glype proxy server, a user must download the proxy files from the Glype website and then relocate the files to the correct directories on their webserver. Another option would be to use one of the many existing proxy sites already available. The Glype website provides a list of working proxy servers whose administrators have paid to have their site listed in the hope of increasing the popularity of their own server. At the time of writing this list contained 3,389 unique servers. This list, however only represents those that have paid to have better exposure; there are possibly many more Glype proxy servers. This presents a problem when trying to block access to these proxies because there are so many. This makes it difficult to compile a complete list to add to an IP block list or ACL. In addition, because it is so easy to set up the proxy, new servers are being added all the time. URL filtering will not work either as the majority of proxy servers based on the Glype script will have some form of URL obfuscation available. The most popular methods of obfuscation are encoding the URL using either base64 or ROT-13 encoding. Other methods of encoding exist, but these are the main ones used by the Glype script.

The CGIProxy is a Common Gateway Interface (CGI) script that acts as a HTTP, HTTPS or FTP proxy. CGI scripts can be programmed in a number of different languages. CGIProxy is programmed using the interpreted language Perl. While Glype proxies enable URL obfuscation by default, the CGIProxy script does not. ROT13 encoding can be enabled by removing the line comments for the methods *proxy_encode()* and *proxy_decode()* in the script. The script also provides support for custom encoding code to be added such as hexadecimal encoding.

2.6. SSL/TLS

Secure Sockets Layer (SSL) and its successor Transport Layer Security (TLS) are security protocols for establishing an encrypted link between a server and a client, for example, a website server and a user's browser (Digicert, 2014). SSL and TLS operate on top of TCP allowing protocols on higher layers of the network stack, such as HTTP, to be left unchanged while still providing a secure connection. Underneath the SSL layer, HTTP is identical to HTTPS. After building the TCP connection, the client starts the SSL handshake with the server. The handshake protocol is where the client and server agree on a protocol version, select cryptographic algorithms and authenticate each other [15]. After the handshake is established, the server will send its certificate to the client. This certificate is used to verify the server's identity by the client. The certificate must be trusted by the client or by a party that the client trusts in order for the identity of the server to be verified. Once the certificate has been verified, a key, most likely a public key, may be exchanged depending on the cryptographic algorithm that the client and server agreed upon. Both the client and server compute a key for the symmetric encryption session and the client tells the server that all communication will be encrypted going forward. The client sends an encrypted and authenticated message to the server which then verifies that the message can be correctly decrypted and sends an encrypted message in response. The connection is now secure and both parties can communicate securely. Fig. 2 shows the interaction between client and server as the handshake process progresses.

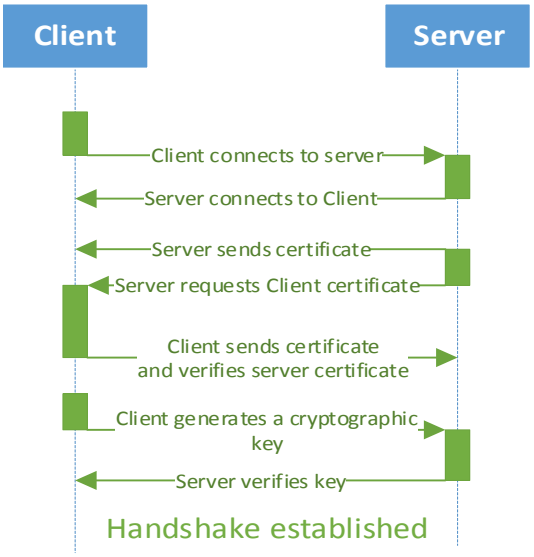

Figure 2. SSL/TLS Handshake protocol

Any attackers that may be eavesdropping on the connection at this point will not be able to see any of the encrypted message contents apart from perhaps the source and destination IP addresses, the ports being connected to and what encryption scheme is being used. SSL originated as a method for setting up and maintaining encrypted communications on the internet. It was designed to be platform independent and to be a generic transport layer mechanism, but the internet remains as its main user [16]. The most recent incarnation of SSL was 3.0 which was released in 1996 and was published by IETF in RFC [15]. TLS 1.0 was then defined in RFC 2246 in 1999 as an upgrade to SSL 3.0 [17] and included a mechanism which allows a TLS implementation to downgrade down to SSL 3.0 again for compatibility reasons. TLS is, at the time of writing, on version 1.2 [18] with version 1.3 being drafted and not fully defined yet.

2.7. The Onion router

Tor is a circuit-based low latency anonymous communication service that is based on the onion routing principles the Naval Research Laboratory [19][20]. It was initially released as a method for anonymous and secure communication with the goal of allowing military personnel to work online undercover, but was later released to the general public. It is now maintained by a group of volunteers called The Tor Project. In the first version of onion routing, instead of making a direct connection to a web server from a client machine, applications on the client side make connections through a sequence of machines called onion routers [19]. This network of onion routers allows the connection between the initiator and responder to remain anonymous. The network is accessed through a series of proxies starting with a socket connection from a client application to an application proxy. This proxy manipulates the connection format and changes it to a generic form that can be passed through the onion routing network. It then connects to an onion proxy, which is the part of the network that defines the route through the network by constructing a layered data structure called an onion. This structure is then passed to the first onion router. An onion router that receives an onion peels off a layer of the onion structure, identifies the next router in the sequence and sends the embedded onion to that router until the last layer is removed and the data is sent to the end point of the sequence. Before sending data over this connection, the onion proxy adds a layer of encryption for each onion router present in the route. As data moves through the connection each onion router removes one layer of encryption along with a layer of the onion structure so it arrives as plaintext. On the way back through the connection, the layers of the onion structure are added back on along with the corresponding layers of encryption [19]. The Tor project is known as the second generation Onion Routing system [20]. Tor was released to address limitations in the original onion routing design by adding a number of features that would improve the operation of the system. The biggest difference between the two is that Tor runs on the live Internet, whereas the original design was mostly operated on a single machine as a proof-of-concept.

2.8. MiTM attacks, ARP Spoofing

One method that can be used to intercept communications between two parties is ARP Spoofing (also known as ARP Poisoning). ARP spoofing occurs when an attacker, who is on the same Local Area Network (LAN) as an end user, sends fake Address Resolution Protocol messages to that user's computer [21]. These messages are sent to convince the user's computer that the attacker's medium access control (MAC) address is the MAC address of, for example, the gateway router of that network. This interception of communications makes use of a weakness in the ARP protocol which does not have any means to check and verify the identities of machines using it [22]. Through ARP spoofing, an attacker can then move on to perform what is known as a Man in the Middle (MitM) attack. Fig. 3 shows a typical layout for a MitM attack.

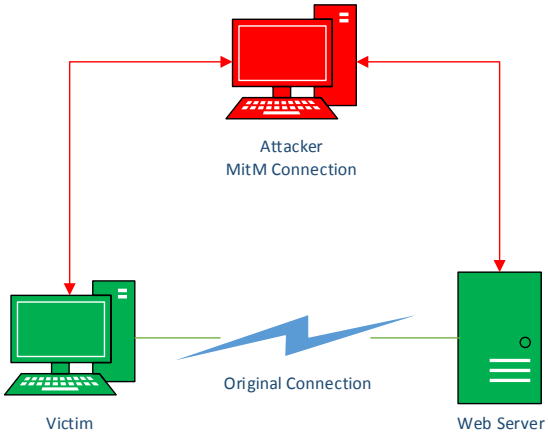

Figure 3. A typical Man in the Middle layout

This attack is a common way to interfere with communications between two parties. To execute a MitM attack, an attacker sets up a form of ARP spoof between two parties that are attempting to communicate with each other. The attacker will then create two simultaneous connections, one to each of the connected parties, and impersonate both parties at the ends of both connections. The two communicating parties view the connection as if they were actually connected directly, not noticing that the connection is being intercepted. Once connected, the attacker has access to network traffic flowing in both directions and can begin to sniff the connections for valuable information, such as bank details or website credentials, or modify the data being transmitted to include malicious code.

2.9. Commercial Detection Solutions

There are a number of commercial solutions available currently for the detection and blocking of anonymous proxy usage. A number of examples and a comparison of what methods they use for detecting proxies and other capabilities they have is presented in table 1. The methods include URL filters, IP Filters, Packet Analysis, SSL Detection, HTTP Header Filters, Adaptive Rule Definition, Pre-defined Rules and IP Geolocation. CIPAFilter is an enterprise level solution intended for use by schools. It is based on either a desktop server, a rack mounted server or as a virtual server. It captures network traffic by forcing students to connect to the internet through the server, with the server acting as a forwarding proxy. It then compares URLs of websites visited with a list of known anonymising proxy websites and then blocks the communication. The problem with this approach is that the list of anonymising proxies is changing all the time. As proxy sites are blocked, new sites are set up to replace them so a lot of time and effort needs to be expended in making sure that the URL lists are up to

date. Exinda is very similar to the CIPAFilter solution, also making use of a list of proxy websites and is also based on rack mounted servers of varying capabilities.

IP2Proxy uses a different technique for spotting anonymous proxy traffic. Instead of keeping a list of proxy websites, it analyses network packets and looks for the HTTP header HTTP_X_FORWARDED_FOR. Whenever a proxy is tasked with forwarding traffic without masking the identity of the original client, then they will include this header and the IP address of the original machine. However this is an optional header and many anonymising proxies opt to not include it in any forwarded requests in order to hide the identity of the client. The Glype proxy script even allows for fake information to be provided, including showing what operating system and browser the user is using, to further throw off trackers. Snort is a popular packet capture and intrusion detection application that is compatible with both Windows and Linux operating systems. It can run in one of three modes: Packet Sniffer, Packet logger or Network Intrusion Detection System. When running as an intrusion detection system, Snort detects and analyses suspicious traffic based on pre-defined rules. It comes with a default set of rules to allow users to get snort set up and working initially and it also allows users to define and add their own rules. Rules for detecting the usage of proxy websites based on PHP and CGI scripts were defined by John Brozycki that can be used to instantly send an alert whenever proxy traffic is detected or, in the hands of more advanced users of snort, even block proxy traffic [23].

Table 1. Comparison of related applications

		Method of detecting Proxies							
		URL Filter	IP Filter	Packet Analysis	SSL Detection	HTTP Header Filter	Adaptive Rule Definition	Pre-defined Rules	IP Geo-location
Application	CIPAFilter	✓							
	Exinda	✓	✓						
	IP2Proxy					✓			
	Snort							✓	
	ModSecurity					✓			✓
	MAXMIND		✓						
	Proxy Detection System	✓	✓	✓	✓	✓	✓	✓	✓

MAXMIND offers a proxy detection service on top of their geo location and fraud prevention services. It involves passing an IP address to one of their data centres where the address is compared against a list of IP addresses suspected of being an anonymous proxy. This however runs into the same problem as using a URL list. ModSecurity is an open source, cross platform compatible, application firewall that can offer proxy detection and blocking when configured to detect GeoIP country code mismatches between the IP address of the final host connecting to a web server and the first IP address listed in the X-Forwarded-For HTTP request header. This makes use of geolocation data through integration with geolocation databases such as MAXMIND. If an anonymous proxy is being used then the IP address of the host connecting to the server will be that of the proxies and this will clash with the IP address listed in the X-Forwarded-For header. However, as with IP2Proxy, this runs into the problem of anonymous proxy hosts choosing to remove the X-Forwarded-For header from the forwarded requests, leaving the firewall with just the proxy server's IP address and nothing to compare it to.

3. Proxy Detection System

Our system is a form of network-based intrusion detection system, developed specifically to detect the use of anonymous proxy scripts in a business or corporate network environment. It will include a number of different components, each with a specific task to perform. These are IP geolocation, a method for getting around SSL encryption used by a growing number of proxy sites and a proxy detection algorithm. Underlying these components will be the capability to capture network packets in real time in a similar way to the packet analysis software Wireshark. IP geolocation will provide information on where, geographically, network packets are coming from. This will be used to help detect the usage of an anonymous proxy by comparing the location data to an online database in a similar style to an IP block list. Depending on the location of the server that an anonymous proxy is running on, the network packets will be passed on for further analysis by the proxy detection system.

The method for getting around SSL encryption that may be used by proxy sites will be based on a penetration testing tool called sslstrip. It is a form of MITM attack that forces a user's browser into communicating with an adversary in plain-text over HTTP. This is possible because many HTTPS sites are normally accessed from a HTTP 302 redirect on a HTTP page. The connection is intercepted before the redirect can take place and modify it to redirect to the HTTP version of a site e.g. https://twitter.com would become http://twitter.com. The adversary then acts like a proxy and forwards the communication on to the internet as normal, using either HTTP or HTTPS depending on what is being requested whilst maintaining the HTTP connection between user and adversary. The "adversary" in the case of this project would be the proxy detection system in an attempt to gain access to encrypted network packets that are being sent to and from anonymous proxies for deeper analysis of their contents. The proxy detection will be the last part in this sequence. This will be the part of the system that will perform the analysis of suspicious network packets. The analysis will be based on the patterns of proxy traffic discussed above. The patterns will be included in a comparison as a rule base for the system. The system will compare network packets that are captured with the rules in real time. If an anonymising proxy is detected then the system will create a log of the detection that includes the packet that was analysed and the time and date it was

captured. An alert to the network administrator will also be sent to inform them of the detection.

3.1. System Design

This system will monitor a network by capturing packets as they go to and from the network and comparing the contents of the packets against a set of rules. Fig. 4 shows the architecture of the network for the proxy detection system. The system is located between 2 firewalls; one controlling access to and from the internet and another controlling access to the innermost network where the client machines reside. This creates an area known as a Demilitarised Zone (DMZ) which is a subnetwork that provides an additional layer of security to a network, separating a business' local intranet from the wider Internet. This is known as the perimeter of the network.

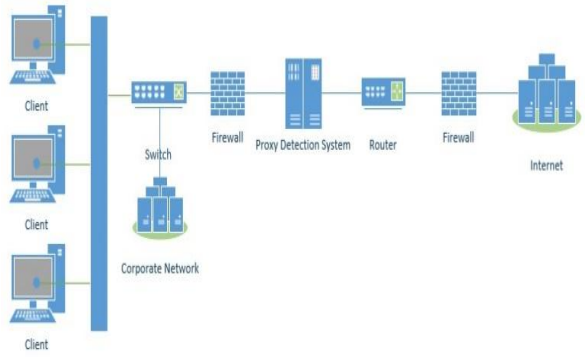

Figure 4. Network architecture

Fig. 5 shows the sequence of interactions between the individual components of the system including the interaction between the client and network admin. The case shown in fig. 5 is a user attempting to use an anonymising proxy to access a website. In this case the system will use all three components to analyse and identify behaviours belonging to the proxy traffic and notify the network administrator about the proxy usage.

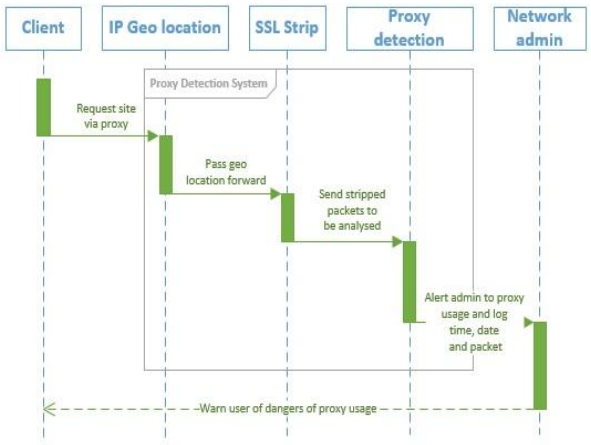

Figure 5. Sequence Interaction diagram

3.2. Software Analysis

Potential software tools are being investigated for the development of the proxy detection system. These tools include the general programming language Python as well as the network penetration testing tool sslstrip. Python is supported on both Windows and Unix-like based systems which means that the system will not be dependent on a single operating system. It also has generous support for packet snuffing and capture through the inclusion of the Scapy or Libpcap libraries.

SSL stripping is a concept that was developed by Moxie Marlinspike in 2009. It is a form of man in the middle attack that allows an attacker to prevent a web browser from upgrading an unencrypted HTTP connection to a HTTPS connection that is encrypted using SSL or TLS. He developed the tool sslstrip that was previously discussed above. The idea behind sslstrip is that users only encountered SSL in one of two ways, they either clicked on a hyperlink such as a login button or through a HTTP 302 redirect. What happens with the 302 redirect is a user will usually not type the "https://" prefix into the URL address bar. Instead they will type in "website.com" which the browser will automatically interpret as a request for "http://www.website.com". If the website being requested only normally runs on HTTPS then the web server of the site will reply to the HTTP request with the 302 redirect code, telling the users browser to request the HTTPS URL instead. Fig. 6 shows what this looks like in the network analysis tool Wireshark. The website requested normally runs on HTTPS as it contains a login form, however the URL request defaulted to HTTP. Therefore the web server sent a redirect telling the browser to instead request the HTTPS version of the website.

```
HTTP        337 GET / HTTP/1.1
HTTP        670 HTTP/1.1 302 Moved Temporarily
```

Figure 6. HTTP GET request for a website followed immediately by a 302 redirect

What sslstrip does is it watches HTTP traffic on a network and whenever it detects "https://" in a URL request, it intercepts the communication and changes it to "http://". Whenever such a connection is detected, sslstrip will then initiate a SSL connection to the desired server and then forwards on the request as normal as if nothing had changed. This way the server never knows that the connection is being forwarded by sslstrip. Everything that is passed along through this connection can be read and logged in an unencrypted format. Incorporating this into the proxy detection system should theoretically allow for network packets being captured by the system to be in an unencrypted format and for the packets to appear normal outside of the system. This would allow the system to apply its proxy detection techniques to proxy packets that would normally be encrypted and unreadable.

4. Conclusion

The majority of current methods for detecting and blocking proxies rely on variables that can be changed very easily such as URL addresses and IP addresses. The method proposed aims at using the contents of the network packets and the format of the URL generated by an anonymising proxy as the foundations for a rule base to be used to reliably and accurately detect whenever a proxy has been used. Then adaptive learning techniques will be applied to classify network traffic and identify its origin. Any proxy traffic identified using this approach will be added to the rule base by the system. This research will also attempt to address the problem of anonymous proxies using encryption through the incorporation of the existing tool sslstrip, which will provide access to unencrypted packets to the detection system.

5. References

[1] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song, "Design and evaluation of a real-time URL spam filtering service," in *Proceedings - IEEE Symposium on Security and Privacy*, 2011, pp. 447–462.

[2] W. Stallings and B. Lawrie, *Computer security*, 2nd ed. Pearson Education, 2008.

[3] S. T. V. Kumar, S. O. S. Khanna, S. M. P. Reddy, S. G. S. Reddy, and S. G. R. S. Reddy, "Overview of Emerging Trends in Network Security and Cryptography," *IJEIR*, vol. 3, no. 1, pp. 51–56, 2014.

[4] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013.

[5] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," *NIST Spec. Publ.*, vol. 800, no. 2007, p. 94, 2007.

[6] P. Stavroulakis, M. Stamp. Handbook of information and communication security.- New York: Springer-Verlag; 2010.

[7] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Ann. Des Télécommunications*, vol. 55, no. 7–8, pp. 361–378, 2000.

[8] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Syst.*, vol. 78, pp. 13–21, Apr. 2015.

[9] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 193–202, Jan. 2015

[10] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Networks*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007.

[11] C. Xiang, P. C. Yong, and L. S. Meng, "Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees," *Pattern Recognit. Lett.*, vol. 29, no. 7, pp. 918–924, May 2008

[12] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *VLDB J.*, vol. 16, no. 4, pp. 507–521, Aug. 2006.

[13] T. Özyer, R. Alhajj, and K. Barker, "Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening," J. Netw. Comput. Appl., vol. 30, no. 1, pp. 99–113, Jan. 2007

[14] K. Ghosh, A. Schwartzbard, and M. Schatz, "Learning Program Behavior Profiles for Intrusion Detection.," in *Workshop on Intrusion Detection and Network Monitoring*, 1999, vol. 51462.

[15] A. Freier, P. Karlton, and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0." 2011.

[16] D. Akhawe, B. Amann, M. Vallentin, and R. Sommer, "Here's my cert, so trust me, maybe?: understanding TLS errors on the web," pp. 59–70, 2013.

[17] C. Allen and T. Dierks, "The TLS Protocol Version 1.0," 1999.

[18] T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2." 2008.

[19] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, pp. 482–493, 1998.

[20] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," 2004.

[21] B. M. Luettmann and A. C. Bender, "Man-in-the-middle attacks on auto-updating software.," *Bell Labs Tech. J.*, vol. 12, no. 3, pp. 131–138, 2007.

[22] T. Chomsiri, "Sniffing Packets on LAN without ARP Spoofing," in *2008 Third International Conference on Convergence and Hybrid Information Technology*, 2008, vol. 2, pp. 472–477.

[23] J. Brozycki, "Detecting and preventing anonymous proxy usage," *SANS Inst.*, 2008.

