International Journal of Digital Crime and Forensics
Volume 10 • Issue 2 • April-June 2018

Detecting the Use of Anonymous Proxies

Jonathan McKeague, Ulster University, Londonderry, United Kingdom
Kevin Curran, Faculty of Computing and Engineering, Ulster University, Londonderry, United Kingdom

ABSTRACT

The Internet is built atop the Internet Protocol (IP) which has at its heart a unique identifier known as an IP address. Knowing the location of an IP address can be very useful in many situations such as for banks to know if a connection is in progress from online fraud hotspots. IP addresses can be spoofed allowing hackers to bypass geographical IP restrictions and thus render some category of fraud prevention useless. Anonymous proxies (AP) which act as intermediate relays which disguise the source IP addresses can play a large role in cybercrime. There is a need to ascertain whether an incoming IP connection is an original source matched IP address, or one being routed through an anonymising proxy. This article concentrates on various methods used by anonymising proxies, the characteristics of the anonymous proxies and the potential mechanisms available to detect if a proxy is in use.

KEYWORDS

Anonymous Proxies, Network Security, Security, Traffic Classification

1. INTRODUCTION

Almost 3 billion people access the Internet daily (ITU, 2013). Whether Internet users are checking and sending emails, reading an online newspaper, researching, doing online shopping or online banking, the need for a secure system is a major challenge for those who develop internet security systems (Mallia, 2013). This is especially true for users that use the internet to do business, or send private information, as more people are finding different ways to 'hack' into secure servers and exploit vulnerable data. In 2011 alone, the total amount that was stolen from businesses online amounted to $3.4 billion, which was up by $700 million from 2010 (Neustar, 2012). This figure is likely to increase, with businesses using the Internet more. It is therefore a priority for businesses to invest in methods to protect themselves against such attacks.

Internet misuse is also a major headache for employers due to the increase in popularity of websites such as Facebook, YouTube, Twitter and Google+. This has led to a decrease in the productivity

DOI: 10.4018/IJDCF.2018040105

of their employees, which in turn leads to less profit. Network administrators have therefore had to block many of these websites from being used in the workplace in an attempt to mitigate the problem. Initially they attempted to simply block the IP of the websites. IP addresses are registered to specific geographical locations, although they don't give the exact area of where the user is located. However, it does pinpoint the country that is accessing the network (Goralski, 2008). IP blocking worked quite well, as anytime a user tried to access a website that had its IP blocked they would be denied access. This prompted users to try to find a way around the blocked IP's.

One simple method was the use of a proxy. A proxy website masks the IP of the website that you are trying to view, which bypasses the IP blocking method used to detect the blocked website. Due to an increase in online banking, banks themselves have had to increase security in their systems and networks; examining IP's is one method they utilize. If a user is making a transfer online and the IP looks fraudulent, then the account holder will be contacted before the transfer is verified. There are thousands of free PHP/CGI proxies to use online, making it a simple way to bypass this basic security feature. Even if the proxy server that was used was blocked there are thousands more to choose from, making the task of blocking them difficult (Lyon, 2009). The code for all of these proxies is open source, it can be downloaded and setup with ease, which means that anyone with a computer could theoretically create a proxy server. Another method that can be used to bypass security measures is Onion Routing (e.g. Tor Browser) which is used to anonymize a user's traffic on the internet. This method uses a different port than what is typically used to access blocked websites. Onion Routing works by routing internet traffic through many different hosts, encrypting data at each different host (Dingledine et al, 2004).

This paper outlines a system called DetectProxy which can detect if any proxies are being used in the network by comparing the characteristics of the different proxies. This will be accomplished by analysing the packets entering the network using scripts to determine the type of proxy being used. Once the proxies have been identified, information will be sent to the network administrator. They will then be able to examine the time the proxy was in use and will give them the option to block the proxy if the proxy has been determined to be harmful or not needed on the network. Blocking the proxy will provide a more secure network for the business or institution.

2. ANONYMOUS PROXIES

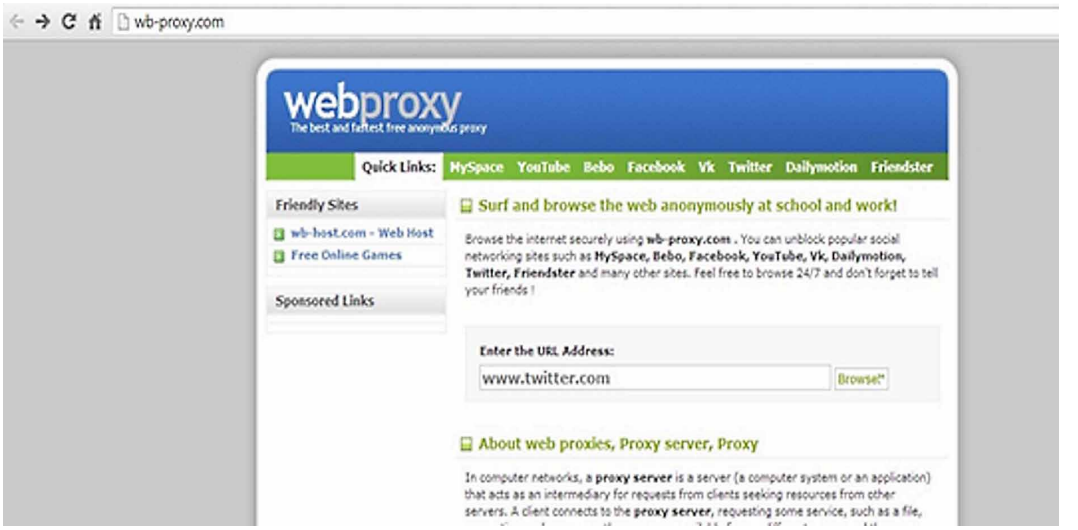
This literature review will be split up into two different sections. The first section will discuss the different ways people can access networks and systems using anonymous proxies. The second section will discuss the different ways of stopping or blocking the anonymous proxies and the different tools used to aid this. Some of the main proxies or ways to access the Internet anonymously are PHPProxy, CGIProxy, Glype, Onion Routing/Tor and SSL Proxy.

PHPProxy is one of the most commonly used Anonymous Proxy Servers. The code is written in PHP and can be obtained from SourceForge[1]. It can run on Windows, BSD (Berkeley Software Distribution), Solaris and Linux platforms, therefore making it possible to run on the majority of platforms. When taking a closer look at the statistics of the amount of times the code has been downloaded, we will see that over the past year there has been a gradual decrease, with the most downloads being 573 in one month and the lowest being 243, these statistics can be found on the SourceForge website[2]. A sample of a proxy website that uses PHPProxy can be found at http://wb-proxy.com/. This website simply allows the user to enter the URL destination that they would like, once entered it will re-direct the user to their website; this can be seen in Figure 1.

The resulting URL when the user clicks 'Browse' is as follows:

http://wb-proxy.com/index.php?q=aHR0cHM6Ly90d2l0dGVyLmNvbS8%3D

Figure 1. PHPProxy Website



The PHPProxy server obfuscates the URL to Base64 encoding; this means than any network administrators that use keyword analysis methods of blocking websites will not be able to block this method. Upon further inspection of the proxy URL, it can be split up into three parts. The first part is the hostname, which is http://wb-proxy.com, the second part is 'index.php?q=' and then the third part, the obfuscated URL, which in this case is 'aHR0cHM6Ly90d2l0dGVyLmNvbS8%3D'. When the obfuscated URL is put in a Base64 encoder/decoder[3], the outcome is 'https://twitter.com/'.

Base64 encoding is particularly important when the PHPProxy server is being used, if it was not used, the URL would be: 'http://wb-proxy.com/index.php?q=https://twitter.com/'. This would be easily detected by a keyword analysis program and blocked.

CGIProxy was created by James Marshall back in 1998 and can be downloaded from his website[4]. A notable difference between PHPProxy and CGIProxy is that the CGIProxy does not obfuscate the URL unless it is programmed to do so. This means that the programmer who is setting up the CGIProxy will have to customise the code, so that it obfuscates the URL. It can be used as a HTTPS, HTTP or FTP Proxy. There are three main ways to encode the URL, these are: Base64, ROT-13 and Hex. In PHPProxy, it solely uses Base64. A sample of a CGIProxy website that encodes the URL is https://scusiblog.org/proxy/nph-proxy.cgi. When www.twitter.com is entered into the website, the outcome is as follows:

https://scusiblog.org/proxy/nph-proxy.cgi/-0/68747470733a2f2f747769747465722e636f6d2f

From this we can see that the obfuscated URL is completely different from that of a PHPProxy altered URL. When the URL is split down the '-0/' can be removed from the hostname, leaving '68747470733a2f2f747769747465722e636f6d2f'. This particular CGIProxy uses hex encoding, therefore entering the string into a hex decoder[5] will leave you with 'https://twitter.com/'. The two URL's that are created by both PHPProxy and CGIProxy are completely different, however the result from both are exactly the same. The CGIProxy if it had Base64 encoding would be very similar to that of the PHPProxy.

Glype is a web proxy that has been coded in PHP. Glype was first released in 2007 and since then there has been over 721,000 downloads of the code[6]. When looking through a list of different proxies[7], Glype in particular stands out as being one of the most popular choices for hosting a proxy

server. Glype is very similar to PHPProxy, it uses PHP as its programming language and it uses Base64 to encode the obfuscated URL. The main difference between the two is the encoded URL; the encoded URL appears different from that of a PHPProxy encoded URL. An example of a Glype powered anonymous proxy can be found at https://branon.co.uk/glype/desktop-free/. As before in the other proxy websites, the user can enter in the website they want to view and just click 'Go', this will bring them straight to their destination webpage. When www.twitter.com is entered into the website, the resulting URL is as follows:https://branon.co.uk/glype/desktop-free/browse.php?u=czovL3R3aXR0ZXIuY29tLw%3D%3D&b=1

When you extract the encoded URL that contains the Base64 encoded string and compare it with the encoded URL from a PHPProxy, you can see the difference. However, upon decoding the URL the result is exactly the same. Decoding 'L3R3aXR0ZXIuY29tLw' with a Base64 decoder simply leaves '/twitter.com', the rest of the data in the encoded URL is just extraneous data.

2.1. Onion Routing and Tor

Onion Routing sends data through a network of nodes/servers, each node encrypts the data once it receives it the data goes through a series of different nodes, until it reaches the exit node (Lee, 2013). When the exit node is reached the data is then decrypted. The 'Onion' part refers to the various layers of encryption that takes place when moving through the different nodes. As each of the nodes encrypts your data, this makes the data virtually impossible to trace (Chaabane et al, 2010). Onion routing also uses several different ports on your computer to access the Internet, this makes it more difficult for network administrators to monitor traffic, as it will not only be going through the normal port for internet browsing, which is port 80 (Reed et al, 1998). The Tor Browser was originally called TOR, which stood for The Onion Router (Li et al, 2011). The Tor browser is exactly like any other web browser; however, the main difference between it and Chrome/Safari/Opera is that the user can surf anonymously. The Tor browser was first released in 2002. It was originally developed with the U.S, Navy in mind, for the purpose of protecting government communications. Originally this was its main use, however in more recent times; the popularity of the Tor Browser has steadily grown, with more people growing concerned about their online privacy with one of the main reasons behind this being the NSA surveillance revelations by Edward Swonden (Dredge, 2013).

The Tor Browser bundle is simple to setup and can be downloaded directly from the Tor website[8]. Once the bundle has been installed the user is presented with the Vidalia Control Panel from there they can connect to the Tor Network. While browsing the Tor Browser, users can access thousands of websites that they cannot view on a normal web browser. A typical URL on the Tor Browser looks like this: http://kpvz7ki2v5agwt35.onion/wiki/index.php/Main_Page. If the URL is entered into Chrome, it will bring up no results. Most of the websites on the Tor Browser use '.onion'. The high level of security provided by the Tor Browser may suit some organisations who want to send data through a secure network, however blocked websites can also be accessed through the browser, therefore a way of determining whether someone is using the browser is a must.

2.2. SSL Proxy

A Secure Sockets Layer (SSL) is the standard way to get an encrypted link between a web browser and a web server[9]. Whenever a user accesses a SSL Proxy, they will be using 'HTTPS'. Since the Proxy is using SSL it will encrypt the URL with 265-bit encryption, making it virtually impossible to detect in a network. One of the main problems associated with SSL Proxy's is the cost. SSL certificates are expensive and most anonymous web proxies will not pay for them, as they are trying to provide a free service. Some of the SSL Proxy sites that do charge (http://www.slickyproxy.com/ is an example), can be easily blocked by a network administrator, as it will be a static URL. Even if a free proxy is blocked by a network administrator, ten new proxies will replace it. The main income that the free SSL proxies such as thesslprxy.com https://www.thesslproxy.com/ will get is from advertising. When

entering www.twitter.com into the proxy website, the resulting URL is: https://www.thesslproxy.com/browse.php/CiNBfghu/8_2FLToI/7Me2cWjh/YpqKxM7W/8dVJGzXr/W1/b29/#.UoOc9vm-2m4

In comparison to the other proxies in this paper, we can see that this obfuscated URL is completely different. This is nearly impossible for a keyword analysis filter to pick up, however due to the lack of availability of SSL Proxies; many of them can be blocked, making a SSL proxy an unviable option.

2.3. IP Blocking

IP blocking is one of the most common and basic methods of blocking, filtering or censoring IP addresses that may potentially have a bad effect on the network/server (Thomas et al., 2011). When using this method of security, a network administrator can block a single IP or many different IP addresses from accessing the network, or certain parts of the network, depending on the level of security needed. Whenever the administrator has a list of blocked IPs in the network identified, anyone on the network who tries to access any of the IP addresses will be blocked from doing so (Murdoch & Anderson, 2008). Network administrators can also block IPs from accessing their network, this means that any IP not in the network that is blocked, will not be able to access their network. This is very useful if the network administrators have identified an IP that is trying to cause problems within the network. Companies such as Yahoo and Joomla have detailed measures in place for IP blocking, for instance Yahoo has a service for users who have a store set up with them in their Merchant solutions section[10]. Within this there is admin tools that are very useful, one of them is a section where you can enter IP addresses that you would like blocked. Firstly, you have to find the details of the IP you want to block using the DNS lookup, again provided by Yahoo. This will provide the IP address needed in order for you to block it. Yahoo allows up to 25 IP addresses to be added to the block list at once, however it does have its restrictions, one of them is the fact you can only block 150 IPs in total. Joomla is another company that provides solutions for IP blocking to its customers. Joomla is a content management system (CMS); they allow users of their product to build websites and other applications online[11]. They also have extensions that can be added onto the websites that are created, some of these include: content restriction, email authentication, content protection and IP blocking. In the IP blocking section, they have different types of extensions that can be added to the website, these are: Country/IP Block, Jban, GeoBlocker, CFBlockCountry, Um Ban, TorlpBlock and Ju BlockIP[12]. These extensions can be very useful when combined, for instance, if you did not want a certain country accessing your network, CDBlockCountry should be used, this extension will filter out any IPs from the country you want blocked and will not allow access to them. IP Blocking is a simple method of stopping a user from accessing a network, as it will make sure that the IP that is listed to be blocked is indeed blocked; however, this form of security is easily bypassed with the use of a proxy.

2.4. Access Control Lists

An Access Control List (ACL) is used by network administrators as a way of allowing different ports on user's local machines to be accessed or opened. The ports that are included in the ACL are called access control entries (ACE) (Microsoft, 2013). Whenever a user's port is included in the ACL, they are allowed to access the network, however any application used by the user will also have to be included in the ACL, this is due to the security in the ACL being very rigid. When a port that is not included in the ACL tries to access the network, it will be blocked straightaway. Although this shows that the ACL is actually working properly, a valid user who is using a port that is not on the list will find themselves being unable to access the network, they will have to contact the administrator to add them to the list. This may take some time, depending if the network administrator is onsite or if it is part of a major multi-national company. An example of a company that focuses on providing an ACL service to companies is Cisco. Within their ACL's, they have different criteria that has to be met when setting up the lists (Cisco, 2006). A network administrator can set up many different ACL's for different departments within the one company, for example, if a company has 2 departments (Research & Development, and Government), a network administrator can specify if a port can access both of

the departments or only just one of them. If the administrator does not include the port in the list, then the access to the two departments will be denied. In large companies that have many different networks and sub networks, setting up an ACL can take a lot of time (Lee et al, 2005).

2.5. Geolocation Security

Location Based Services (LBS) such as Parcel Tracking, Indoor Positioning, GPS Navigation and accessing networks have become a vital occurrence in some people's lives. Most if not all new smartphones come with GPS abilities inbuilt in them. People often track their parcels to have an idea of when they might arrive or to find out what is causing a delay in their delivery. Indoor Positioning systems such as SeniorLab[13], Pole Star[14] and IndoorAtlas[15] have all become very popular products over the past two years, as the indoor positioning market has seen a sharp rise, with more shopping centres, museums and airports using this new technology. Another useful service in the Location Based Services section is Geolocation Security. Within Geolocation Security companies can monitor who accesses their networks and sometimes block certain users from accessing the networks based solely on their location. If a company was being attacked by a hacker, the network administrator can look at the IP address of the hacker, find out what country the IP is located in and block the IP addresses associated with that country for a brief period of time until the attacks stop (Kibirkstis, 2009). One of the main companies that supplies software in the field of geolocation security for online applications is Neustar, formally known as Quova. One of their main products is IP Intelligence. This product provides the company using it with data on their customers, where they are and what they are using to connect to the web. Having access to this information makes it easier for companies to block transactions that they deem suspicious. Gmail also uses Geolocation Security, Gmail will monitor the user's main IP address logins and will then contact the user if a suspicious IP address has tried to access the account, and this gives the user a chance to change their password before the hacker can access their email[16].

2.6. Base64 Encoding

Base64 encoding takes a string of text data and changes it into ASCII format. One of the main reasons for changing the text data to ASCII is so that when messages are being sent through a network that generally deals with text, it can be sent through securely (Knickerbocker et al, 2009). Base64 encoding is very useful when it comes to bypassing IP Blocking or blacklist filtering, for example, when you enter www.twitter.com into a Base64 encoder, you get the following output: d3d3LnR3aXR0ZXIuY29t. Many proxy websites will use this form of encoding to bypass any filters on the network.

To convert text over to Base64 format, firstly you have to change each character to its equivalent ASCII value. Once the ASCII value is got, it will be changed into 8-bit binary format. Each 8-bit binary is split into 6-bit binary groups; each 6-bit binary number is converted into a decimal number. The decimal number is then compared with the Base64 index table, which is shown in Figure 2.

Table 1 shows steps involved in converting 'www' to Base64 encoding. The reason why the binary number is split into 6-bit is so that all the Base64 values can be represented. The maximum binary value in 6-bit format is 111111, which when converted to decimal format equals 63, the biggest value in the Base64 index.

A major security risk can be PHP obfuscation; Base64 encoding can be used to do this. The code in some of the web-based programs can be made extremely difficult for a human to read if it is converted to Base64, therefore rogue code, or code that can be harmful can make its way onto the machine without the user or some security software knowing (Raynal et al, 2012). However, changing the code to Base64 can sometimes be quite a tedious task, and mistakes can often occur. In HTML5, there has been two methods created that has allowed developers to change the pages content to and from Base64 encoding. These two methods are atob() and btoa()[18]. These two methods are very useful when looking to change binary to Base64 and vice-versa.

Figure 2. Base64 index[17]

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Table 1. Base64 encoding

Letter	W	W	W	
ASCII	119	119	119	
Binary	01110111	01110111	01110111	
Divided Binary	011101	110111	011101	110111
Decimal	29	55	29	55
Base64 encoded	D	3	d	3

2.7. Snort

Snort is an open source network intrusion detection and prevention system that was created by Martin Roesch and released in 1998; the Snort program is able to run quietly in the background, providing real time traffic analysis and packet logging within networks[19]. Snort has many useful capabilities in terms of detecting attacks and probes, some of these include: Stealth port scans, Operating System Fingerprinting attempts, Server Message Block (SMB) probes, Buffer Overflows and Common Gateway Interface (CGI) attacks (Stanger et al, 2007). Sourcefire, a company that was founded by Roesch, currently owns and continues to develop Snort. The program has had millions of downloads and currently has nearly 400,000 registered users19. Snort provides three different functions/modes, these are Sniffer Mode, Packet Logger Mode and Network Intrusion Detection System (NIDS) Mode.

Sniffer mode reads all the packets that are going through the network; it will then display all the packets that were read from the network on the console. This process runs continuously until the user turns it off. Packet logger mode like the sniffer mode will read all the packets going through the

network; however, it will save the packets to a disk instead of displaying them continuously on the console. The NIDS mode will monitor all traffic that moves through the network and will detect any intrusions that occur. This is the most complex mode of Snort (Sourcefire, 2013). The installation of the NIDS can be complicated; however, there is a step by step guide in order for the program to be installed correctly. SNORT can be used in conjunction with other programs in order to analyse the data that is going through the network, an example of one such program is BASE (Basic Analysis and Security Engine). BASE is a web interface that analysis the intrusions that are detected from the Snort IDS (intrusion detection system), within the program users can also use the simple web-based setup program for those that might not be comfortable in editing files[20].

3. CAPTURING NETWORK TRAFFIC

It is important for network administrators to monitor traffic that is entering and exiting the network. Security is very important therefore it is vital that any proxy or onion routing applications used can be identified. If a proxy is in use, more often than not it is not being used for legitimate reasons. There can however be a valid reason for someone in the company to use a proxy, for instance, if they wanted to block certain web scripts from being used or if they needed to test an application that is being mistakenly blocked online, the user should contact the network administrator to allow the blocked application to be made available, the latter however should not happen.

3.1. Monitoring Network Traffic

The first step in the design of the system is to monitor the network traffic. Programs such as "Httpfox", "Snort" or "Wireshark" can be used to monitor the traffic. This category of program can 'sniff' the traffic on a continuous basis, which is ideal for intrusion detection systems. Included in the data is the Destination IP, the Source IP, the Protocol and different information about each packet. All of these different programs have the ability to save the network packets into a text (.txt) file; this would make it easier for the IDS to read the packets.

The first step in the design of the IDS was to examine the network packets and have a good understanding of the data provided in each of the packets. Being able to determine the relevant information in the packets would greatly reduce the time spent when finding common sequences within the proxy packets. Therefore, Wireshark would initially be used to monitor the network traffic, with the results from the monitoring displayed in a text file, this text file will be key to determine how to detect an anonymous proxy. As the main program being used to 'sniff' the network packets was Wireshark, the initial idea was that it could be used in conjunction with the proxy detection script. This however meant that the program would not be standalone, which would not have been ideal. As the IDS's main purpose is to work on different platforms as a standalone program, a program wrote in Python was used. This program examined all the packets coming into and leaving the network. From this it could be altered to print all the packets to a log file, so they could be examined, or run in the background scanning each of the packets as they entered/exited.

3.2. Software Used

Python was used as IDS is run on multiple platforms as the network administrator may be using more than Windows. One of the main libraries used in the Python script is the Pcapy python library. Pcapy can be defined as follows "Pcapy is a Python extension module that interfaces with the libpcap packet capture library. Pcapy enables python scripts to capture packets on the network"[21]. The Pcapy library will be used alongside other libraries in the script to produce the IDS. One of the other main libraries is the 're' library[22]. The 're' library provides support for regular expressions which will be used frequently within the script; these regular expressions will be used to match different key words against words in the network packets.

The network administrator will start the program running; they can do so by running it on the command line or using a python program such as IDLE[23], IDLE is the python IDE, it's a multi-platform IDE with a multi window text editor, it also has a python shell window, where the network administrator can interact with the program. Once the program is started, a list of all the available network interfaces is presented. Each of the available network interfaces is listed with a number, so the administrator can select the interface they require. Once the interface is selected, the IDS starts scanning the network packets continuously. When the script has started a new directory is created. This directory stores all the log files that are created when the IDS is in use. Along with a directory being created, a log file is created to store any proxies found when it is running. A new log file is created each time the program is restarted, to differentiate between the logs a date and time is supplied within the log file name. A list of common characteristic strings that each of the different proxies has when the network packets are traversing the network is necessary. Each of the proxies or onion routing applications have their own unique characteristics which make them different from a normal web browsing network packet. As each proxy has its own unique characteristics, this however does not mean if one of the characteristics is found then it is definitely a proxy. Each of the proxies have to match two or more of the characteristics before they are flagged up to the administrator and printed to the log file. As the program is designed to work in large networks, there could be many different proxies or onion routing applications being used at the same time. Matching the different characteristics and printing the result to the log is vital. As the program is running continuously, each packet containing the matched characteristics will be printed to the log, providing the type of proxy, the date and time each packet went through the network and all the different information contained in the packet. This information that is contained in the packet will allow the administrator to track down the computer using the proxy and disable the access or query the usage of the proxy. The time it takes from a proxy entering the network and it being logged should ideally only be a few seconds; this enables the administrator to quickly find the proxy user.

3.3. Log Files

As the system runs on a local machine the use of a database is not necessary. The IDS stores each proxy found in a text file which is similar to the output received when Wireshark is used to examine the network packets. However, the size of the text file is greatly reduced depending on how many proxies are in use on the network. The file is designed with the user in mind and only the necessary details are printed to it. The details that are printed to the log can be viewed in Table 2.

A number of the packets may not contain all of the details that are listed in Table 5. However, they will contain the majority. The most important details contained in the network packets are the Proxy Name, the Date and Time of Proxy Usage, the Destination MAC, Source MAC, both the Source and Destination IP Addresses and the Source and Destination Port. All this information should give the network administrator enough details to track down the proxy usage. Due to the large amount of network packets, if there is prolonged proxy usage without the network administrator addressing the situation then the log file could become very large and may take a while to open, therefore it is a good idea to monitor the program, and restart it if the file is getting too big. Restarting the program will simply create a new log file with a new timestamp.

3.4. Wamp Server

As the proxies had to be hosted on a web server, a server had to be sourced. WAMP v2.4 was the version used. It contains Apache 2.4.4, MySQL 5.6.12 and PHP 5.4.12. One of the main sections of wamp is the www directory, and this directory is where each of the different proxies is stored. Different versions of Apache and PHP can be downloaded and installed by selecting the Apache folder/PHP folder then selecting the version. Selecting a different version can be necessary for older versions of the proxy that may not be able to use the newest version of PHP. To make sure the wamp server is working correctly, open a web browser and in the address bar type in http://localhost, if the

Table 2. Network Packet Details

-Proxy Name	**-Date and Time of Proxy Usage**
-Destination MAC	-Source MAC
-Protocol	-Version
-IP Header Length	-Time to Live (TTL)
-Source IP Address	-Destination IP Address
-Source Port	-Destination Port
-Sequence Number	-Acknowledgment
-TCP Header Length	-Data
-Host	-User Agent
-Accept (html, xml, etc.)	-Accept-Language
-Accept Encoding	-Referrer
-Cookie	-Connection Type

wamp server homepage appears, the wamp server is working correctly. When the WAMP server is functioning correctly, the proxies can be downloaded and placed in the server, the Tor Browser can also be downloaded; however, it does not need to be placed within the server. The Tor browser can be downloaded directly from the tor project website[24]. The download contains a Vidalia Control Panel and the Tor web browser itself. Each of the three web scripts were downloaded next, PHPProxy[25], Glype[26] and CGIProxy[27], all of the proxies were available to download as a ZIP file, which can be extracted into the www directory on the wamp server. The files however have to be edited before they can be used properly on the server. Perl has to be downloaded before the CGI Proxy can be used; CGI functionality also has to be enabled. PHP functionality has to be enabled before the PHPProxy and Glype web scripts can be used. Whenever these steps are performed each of the web scripts can be used to browse the internet anonymously. Free proxies can also be found online that enable you to test the system and also to compare the network packets, there are many lists that contain these proxies, a sample of the list can be found at http://list.glype.com/.

4. PROXY DETECTION

When the proxies are running the network, packets have to be captured, to do this Wireshark had to be downloaded and installed[28]. Python was used to sniff for network packets. The code initially printed all the packets out to the command line or to IDLE, as the code was open source it could be edited which made using the code very convenient. It was decided to continue to use the Python network analysis code as part of an integral part of the IDS. The first step was to get it to print the code to a log file. This was done by creating a directory to store the log files in. Once the directory was created it would be checked each time the program is run, just to make sure it exists, if it doesn't it will create it. The log file is the next item that is created each time the program is started, this however is different from the directory as it is not always static, the log file created will have the date and the time that it was created in its unique name. After the log file code was finished, selecting the network interface that needed to be scanned had to be coded. The original code had the function to search for the network devices, however selecting the devices when they were printed was time consuming.

4.1. Glype Proxy Detection

For each of the different proxies there were four different logs created to compare each of the packets to find similarities within them that could be used to prove that they are in fact a proxy. If any of the similarities are also contained in normal web browsing packets then it may throw off the results, therefore getting the similarities to be unique is a must. In comparison to a normal web browsing network packet, there are a few common occurrences, one being the destination port, which is port number 80. This is the port that is used for most of the network packets, if the packets that are going through the network are secure, then it would be going through port 443. Each of the packets viewed when the Glype proxy was being run contained the command "GET", and the protocol used was "HTTP" the command and the protocol were contained within the data in the packet. Another difference noticed in the packet was the use of "browse.php?u=", in particular '.php?u=' was identified, this is mainly because the 'browse' can be called anything as that is just the index page, therefore this may differ between the different proxy servers. Once the three characteristics had been identified they could be used to detect the Glype proxy. The first action that had to be taken was to add the three different characteristics to a list. To be able to search the proxies entering the system, after some research it was decided it would be best to use Regex. To get started with regex, 'import re', was added to the globals in the IDS code. Then the regex strings were created, these strings were specifically created so they would ignore case sensitivity and also whitespace.

After the regex list was completed it could be used to match against the network packets. The regex will go through each different characteristic and try to match it against the packet, this can be seen from the line 'glype[0] = re.match(glypeStrings[0], str(packet1))', the string .php\?u=' will be matched against packet1. The code will then go through an IF statement, if all three characteristics are found within the packet then it will make the result equal to 1, it will also print 'GLYPE' followed by the time the proxy was found and then the packet that the proxy was found in to the log. The result is passed through the loop, if it equals 1, then "Glype usage detected" would be printed to the console.

4.2. PHPProxy Detection

The detection method of PHPProxy is very similar to the detection of Glype. The protocol used in the packet is 'HTTP' and the command is 'GET', the only difference between the 3 characteristics of Glype is the third characteristic. In the packet 'index.php?q=aHR0c' is the common occurrence in the different log files. Again, the 'index' part of the string can be dropped as it can vary between the different proxy servers, this leaves the detection string as '.php?q=aHR0c'. The only difference from the PHPProxy code and the Glype code is the detection string in the regex. The result if a proxy is detected will be '2', which would result in "PHPProxy Usage Detected" being printed to the console.

4.3. CGI Proxy Detection

The CGI Proxy differs from the previous two proxy servers. The previous two proxies use port 80 when transferring packets, while the CGI script uses a secure server and goes through port 443. The data received in the network packet is encrypted as it goes through a secure server using the Secure Socket Layer (SSL) protocol; this makes it extremely difficult to find the characteristics needed to determine if it is in fact a CGI proxy which is being used. Decrypting the data without the use of an encryption key would take many years; this unfortunately means it is impossible to provide the criteria necessary to detect the CGI proxy. The only visible data that can be used from the network packets is the protocol and the port number, which is used by a number of different websites that use 'https', including Gmail, Facebook and all banking websites.

When testing different CGI proxies that are available online it was noted that they all use SSL. 100% of the CGI proxies viewed online charged a subscription fee, which could cost up to €120 a year, or if paid on a monthly basis, €20 per month; due to this fee, they are not as common as Glype or PHPProxy, with both offering their services for free. This however only applies to the proxies using SSL, the CGI proxy can also be used without SSL, though it is highly recommended on the

CGI webpage[29] that it should be used on a secure server. Since the CGI script can be implemented on an unsecure server, the packets would then be readable. The CGI proxies' main difference from the previous two proxies is the use of .cgi instead of .php. The proxy protocol is 'HTTP' and the command used is 'GET', it also uses port 80. Therefore the 4 characteristics used to determine the usage of an unsecure CGI proxy script are: HTTP, GET, .cgi and Dest Port: 80. The format of the code is similar to the other two proxies, the only difference being the extra matching string. If each of the characteristics are matched in the network proxy the result will be printed to the log firstly then the console.

4.4. Tor Browser Detection

The code for the Tor Browser was the last to be implemented. The detection characteristics compared to the other three proxies are completely different. This is mainly due to the randomness of the network packets when the Tor Browser is being used. The Tor Browser uses many different ports when sending and receiving packets, the different ports are: 9001, 9002, 9003, 9004, 9030, 9031, 9032, 9033, 9150, 9151, it also uses port 80 which is used for all normal web browsing that doesn't use SSL and also port 443, which is used for secure browsing. The main two ports the Tor Browser uses are port 80 and 443, these two ports however cannot be used to identify the onion routing application, as all normal web browsing would also be flagged up as using the browser, therefore the other ports listed have to be used to identify it. This unfortunately means the Tor Browser could be used for many minutes before it is flagged up on the screen. When examining the packet there were a few interesting bits of data that could be seen, firstly the port that was used was port 80, this generally means the data that is in the network packet can be viewed, however the data in the packet in this instance is encrypted and therefore no details can be taken from it. The second thing noticed in the packet was the Source Address, which was 131.188.40.188. When searching for the IP it was found within a list of known Tor nodes, this verified that it was indeed a packet from the Tor Browser. The source address is useful to verify that it is the Tor browser; however due to the large amount of IP addresses in the Tor network it is not possible to add them to the characteristics. This leaves the only way to identify them is through the ports listed above, due to this the accuracy of the data may not always be 100% correct. One of the main differences in the code is the use of the operator 'or' instead of 'and', this is because the system doesn't have to match 3 or 4 different characteristics, it only has to match one of them to flag it up on the console. Another port that the Tor Browser uses is port 9100, this port however is used often by wireless printers and using this port would create a lot of false positive results, it was decided that due to the large amount of false positive results leaving that port out of the detection string would be the best action to take.

5. TESTING

This section will document the thorough testing that was performed on the system to ensure the system is performing the tasks that was detailed in the previous sections and that it is performing them to a high standard. It is important that any errors or unexpected crashes are found and fixed before the end product is finalized.

Each of the different proxies and onion routing applications were tested thoroughly by performing a series of Internet activities that may be carried out on a daily basis by an average Internet user. The activities are listed in Table 3.

The IDS will be given 5 minutes per test to monitor the network and to verify that it is detecting each of the proxies. The websites were accessed by firstly entering the URL into the proxy start page or from the start page in the Tor Browser. The program was also tested when the user was not using any proxy or the Tor Browser, just to verify that it wasn't flagging up any proxies when they were not in use. Altogether there were 60 log files created in total to test the program.

Table 3. Regular Internet Browsing Tasks

Test	Activity
1	Browse the Guardian news website and view videos
2	Log into Gmail and send an email
3	Log into Twitter and view some tweets
4	Browse Amazon and make a purchase
5	Log into Facebook and browse multiple pages
6	Visit the BBC Sports section and post a comment in the comments section
7	Listen to iRadio on their live radio stream
8	Upload an image to Imgur or Photobucket
9	Select a Youtube video from your account
10	Download a ZIP file from a reliable source
11	Perform a search using a Search Engine such as Google or Bing
12	Go to Miniclip and play a game

5.1. Normal Browsing Test

Before any of the proxies and onion routing applications could be tested, the IDS was tested while the user was browsing the internet normally without the use of a proxy. The web browser used for all the tests apart from the Tor Browser was Google Chrome. Only one tab was open at any one time, with all other internet related activities such as Skype, Dropbox and Google drive closed so the tests would be precise.

Table 4 shows the results when there is no proxy usage in the network, there was nothing printed to the console, therefore no proxy was found in the 5 minutes the IDS was running for each of the twelve individual tests. These results are exactly what was expected from the program, if a proxy or onion routing application had been found, the system would be flagging up false positive results.

Table 4. Normal browsing test results

Test	Result
1	No Proxy usage detected
2	No Proxy usage detected
3	No Proxy usage detected
4	No Proxy usage detected
5	No Proxy usage detected
6	No Proxy usage detected
7	No Proxy usage detected
8	No Proxy usage detected
9	No Proxy usage detected
10	No Proxy usage detected
11	No Proxy usage detected
12	No Proxy usage detected

5.2. Glype Proxy Test

The first proxy to be tested was the Glype proxy. It was decided that since the availability of proxies online was very high, it would be best to test a proxy that was currently available online. The web based proxy used to test the Glype proxy was 'www.proxyserver.com'. This proxy was found within a list of available proxies on the Glype website[30], the list also contained many other different proxies which were used to test the other proxies. The first thing to do was to start to the program running. All other web pages were closed to make sure it was just the proxy being used in the network. The IDS was then started to sniff the network packets. The results that were printed onto the console when the program was running during each of the tests are shown in Table 5.

As can be seen in Table 5, the results show that the IDS is working as it should when a Glype proxy is being used in the network. Each test was detected, with the statement 'Glype Proxy usage detected' being printed multiple times. The network packets were also printed out to the log showing the 3 different characteristics used to detect the proxies contained within them. Another Glype proxy was also tested; this proxy can be found at ''. The results from the proxy were identical to those in Table 8, proving that the characteristics are correct and that the IDS has a 100% succession rate when a Glype proxy is in use.

5.3. PHPProxy Test

The second proxy that was tested was the PHPProxy. The same format as the previous test was used to test the proxy. The proxy that was used can be found at 'http://proxyanonymizer.net/'. The results from the different tests can be seen in Table 6. There were problems with using the proxy. While logging into a secure website such as Gmail, the Gmail services blocked the login as the location was completely different from where the email is usually accessed. The IDS however still picked up the use of the proxy, as there were 3 different pages accessed while performing the test.

Once again, the results from the IDS proved to be successful, with 100% of the tests being detected by the system. This proved that the 3 characteristics, 'GET', 'HTTP' and '.php?q=aHR0c' were being picked up in every network packet that was being created by the PHPProxy. A second PHPProxy was tested to verify the results, the proxy can be found at: 'http://proxy-up.net/'. Again, the results returned 100% accuracy.

Table 5. Glype Proxy Test

Test	Result
1	Glype Proxy usage detected
2	Glype Proxy usage detected
3	Glype Proxy usage detected
4	Glype Proxy usage detected
5	Glype Proxy usage detected
6	Glype Proxy usage detected
7	Glype Proxy usage detected
8	Glype Proxy usage detected
9	Glype Proxy usage detected
10	Glype Proxy usage detected
11	Glype Proxy usage detected
12	Glype Proxy usage detected

Table 6. PHPProxy usage test

Test	Result
1	PHPProxy usage detected
2	PHPProxy usage detected
3	PHPProxy usage detected
4	PHPProxy usage detected
5	PHPProxy usage detected
6	PHPProxy usage detected
7	PHPProxy usage detected
8	PHPProxy usage detected
9	PHPProxy usage detected
10	PHPProxy usage detected
11	PHPProxy usage detected
12	PHPProxy usage detected

So far testing both proxies returned a 100% success rate, with 48 tests performed, 24 for the Glype proxy and 24 for the PHPProxy (Figure 3).

5.4. CGI Proxy Test

The CGI proxy was the third proxy to be tested, as it was previously noted, due to the use of SSL in the proxy, the characteristics could not be found and therefore the proxy could not be detected. This meant that the results for the test of the CGI proxy would be a 100% fail rate, this only applied to the proxy when it was using SSL. The proxy however can also be used without SSL and due to this, the characteristics were found.

Table 7 verifies the results as expected when the CGI proxy is using SSL, each of the tests failed to show any proxy usage within the network. The proxy used was found at 'https://morphium.info/'.

After the SSL CGI proxy was tested, a CGI proxy that does not run on a secure server was tested. This proxy's URL is: 'http://anonymouse.org/'. One of the main differences that stand out between the two CGI Proxies URL's is the first one contains 'https' in the URL and in the second CGI proxy,

Figure 3. Pass rate for both the PHPProxy and Glype Proxy

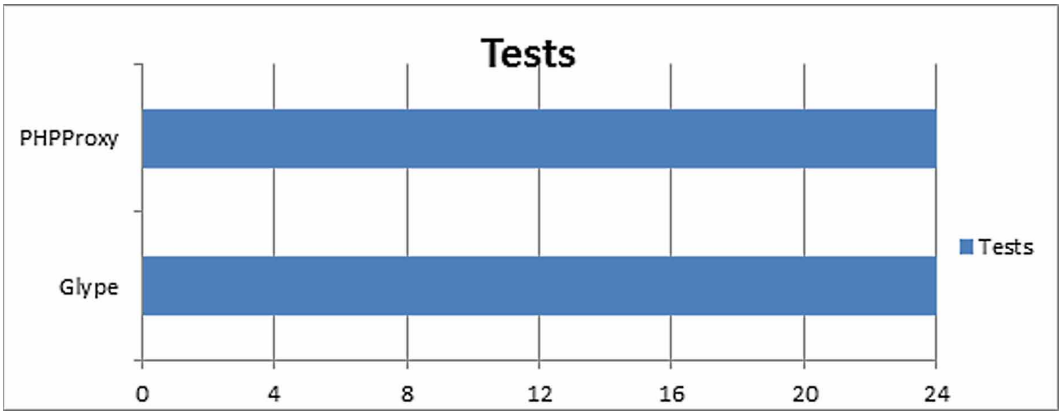


Table 7. CGI Proxy using SSL

Test	Result
1	No Proxy detected
2	No Proxy detected
3	No Proxy detected
4	No Proxy detected
5	No Proxy detected
6	No Proxy detected
7	No Proxy detected
8	No Proxy detected
9	No Proxy detected
10	No Proxy detected
11	No Proxy detected
12	No Proxy detected

it has 'http' in the URL, this shows that the second one doesn't use a secure server. The results from the testing of the unsecure CGI proxy can be seen in Table 8.

The results between the two are stark, with the IDS catching 100% of the unsecure CGI Proxies and the SSL CGI proxy evading detection completely (Figure 4).

5.5. Tor Browser Test

The final proxy/onion routing application to be tested was the Tor Browser. Up until now the results from each of the previous tests have been straightforward, with the results returned as expected. This however was not the case for the Tor Browser, as the characteristics for it did not include two ports, from which most of the traffic flowed through. The results from the Tor Browser testing can be seen in

Table 8. Unsecure CGI proxy test

Test	Result
1	CGI Proxy usage detected
2	CGI Proxy usage detected
3	CGI Proxy usage detected
4	CGI Proxy usage detected
5	CGI Proxy usage detected
6	CGI Proxy usage detected
7	CGI Proxy usage detected
8	CGI Proxy usage detected
9	CGI Proxy usage detected
10	CGI Proxy usage detected
11	CGI Proxy usage detected
12	CGI Proxy usage detected

Figure 4. Pass rate for the SSL CGI Proxy and the Unsecure CGI Proxy

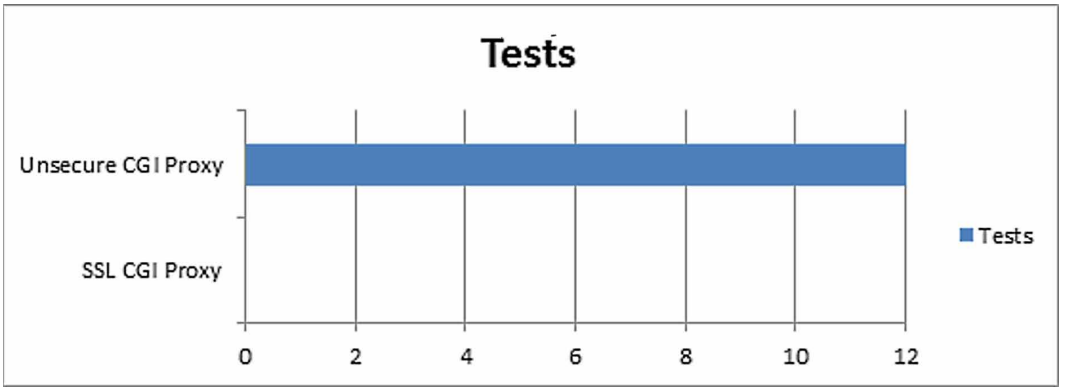


Table 9. With the twelve tests completed, eight of them passed, with 'Onion Routing usage detected' being printed to the console. Four of them resulted in nothing being printed to the console, therefore the IDS did not detect the use of the Tor Browser.

As these tests were carried out during a five-minute period, it is not always guaranteed that the IDS will miss the detection of the Tor Browser. If for instance it had ten minutes per test, the program may have picked it up. As the program is meant to pick up each of the proxies/onion routing applications almost instantaneously using ten minutes to test it would not be feasible. While having a closer look at the results gained from the Tor Browser tests, we can see it failed to detect the browser in tests 2, 3, 5 and 11. These tests involve using Gmail, Twitter, Facebook and Google respectively, one thing in common that each of them share is the use of 'https' for secure browsing. Taking a closer look at the network packets while browsing each of the websites shows that each of them use port 443 for all of the packets, due to this, the IDS will not detect them. Amazon also uses 'https' when the consumer is purchasing an item, this only applies when they are logging into their account to pay for the item. Before this point, amazon uses a regular 'http' connection, so the network packets can go through any of the ports in the characteristics and also port 80. In Figure 5, the results of all the tests can be

Table 9. Tor Browser test

Test	Result
1	Onion Routing usage detected
2	No Onion Routing usage detected
3	No Onion Routing usage detected
4	Onion Routing usage detected
5	No Onion Routing usage detected
6	Onion Routing usage detected
7	Onion Routing usage detected
8	Onion Routing usage detected
9	Onion Routing usage detected
10	Onion Routing usage detected
11	No Onion Routing usage detected
12	Onion Routing usage detected

seen. Three out of the five that were tested had a success rate of 100%, with the Tor Browser having a 66% success rate and the Secure CGI proxy having a 0% success rate.

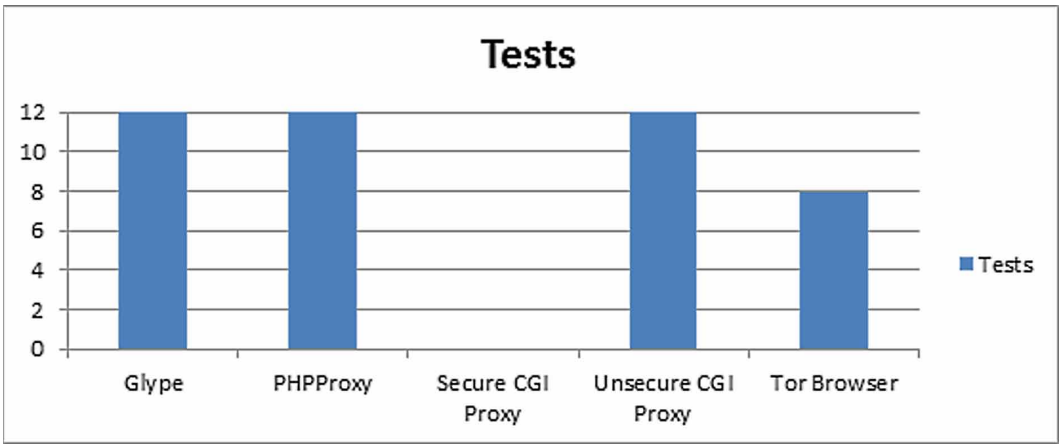
The results from each of the tests were as expected, when the proxy or onion routing application was using an unsecure server the IDS picked up its usage every time, when the proxy was using a secure server it evaded the IDS's detection. The result unfortunately came to the same outcome when other SSL proxies were tested, the IDS did not detect any of them. When using Wireshark to take a closer look at the packets each of them used port 443 and the TCP protocol, as the packets are very similar to those of a regular SSL connection that does not use a proxy there is little that can be done to fix the IDS without creating a lot of false positive results.

6. CONCLUSION

One of the main aims of the project was to firstly examine the packets in the network to see how each of the different packets looked and what was contained therein. Once a good grasp of the data in the packets was obtained, the proxies would then be used to compare the difference. Any noticeable difference contained therein could then be used to determine if a proxy was being used and what type of proxy it was. This was successfully done in four out of the five tests, with the SSL CGI proxy being the only downfall. Again, the program did not have a 100% success rate in determining the Tor Browser, however the differences in the network packets was noticeable in most of them. This criterion in the packets was put into regex strings to be compared with each inbound and outbound packet, in turn successfully determining the different proxies. When the project was first started there was a need for a system that would be able to detect the use of anonymous proxies, security is a major field in Information Technology and the sector is increasing at a fast rate. This system fills that need, it can successfully detect Glype, PHPProxy, Unsecure CGI proxy and the Tor Browser which in turn provides a more stable and secure environment for the company/organisation to perform its everyday tasks. Overall the project would be very useful to a network administrator in terms of monitoring the network packets to determine if there is a proxy in use or if the Tor browser is in use. This system, although it doesn't pick up every single proxy, would be an important system to any company that has high security measures.

The system was hoped to have 100% accuracy in detecting anonymous proxies. This however could not be achieved as the data travelling through a SSL proxy or pages that use 'https' on Tor generally use port 443, and thus the data in the network packets was encrypted. The system however

Figure 5. Full proxy/onion routing results



did achieve 100% accuracy when detecting the Glype, PHPProxy and the unsecure CGI proxy. Also in testing it had 66.67% accuracy in finding the Tor browser. The system's effectiveness can be debatable; it all depends on the type of proxy being used. The system does not detect 100% of proxies, it does however detect 100% of 3 different proxies and 66% of the Tor Browser, and therefore it can be quite effective when detecting those. However, it is not effective when it is detecting SSL proxies.

REFERENCES

Chaabane, A., Pere Manils, P., & Kaafar, M. (2010). Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network. In *Proceedings of the 4th International Conference on Network and System Security* (Vol. 1, p. 167). doi:10.1109/NSS.2010.47

Cisco. (2006). Cisco IOS Security Configuration Guide, Release 12.2, *Access Control Lists: Overview and Guidelines.* Retrieved from http://www.cisco.com/en/US/docs/ios/12_2/security/configuration/guide/scfacls.html

Dingledine, R., Mathewson, N., & Syverson, P. (2004). Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium*.

Dredge, S. (2013, November). *What is Tor? A beginner's guide to the privacy tool.* The Guardian. Retrieved from http://www.theguardian.com/technology/2013/nov/05/tor-beginners-guide-nsa-browser

Goralski, W. (2008). *The Illustrated Network: How TCP/IP Works in a Modern Network*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

International Telecommunication Union. (2013). *The World in 2013 ICT Facts and Figures.* Retrieved from http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2013.pdf

Kibirkstis, A. (2009). Intrusion Detection FAQ: What is Geolocation and How Does it Apply to Network Detection. Retrieved from http://www.sans.org/security-resources/idfaq/geolocation-network-detection.php

Knickerbocker, P., Yu, D., & Li, J. (2009). Humboldt: A distributed phishing disruption system. In *Proc. IEEE eCrime Researchers Summit,* Tacoma, WA.

Lee, J. (2013). What is Onion Routing, Exactly? *MakeUseOf*. Retrieved from http://www.makeuseof.com/tag/what-is-onion-routing-exactly-makeuseof-explains/

Lee, K., Jiang, Z., Kim, S., Kim, S., & Kim, S. (2005). Access Control List Mediation System for Large-Scale Network. In *Proceedings of the 6th Int Conf on Parallel and Distributed Computing* (pp. 483-487).

Li, B., Erdin, E., Gunes, M., Bebis, G., & Shipley, T. (2011). An Analysis of Anonymity Usage. In *Proceedings of the Traffic Monitoring and Analysis: Third International Workshop,* TMA 2011, Vienna, Austria (pp. 113-116). Springer.

Lyon, D. (2009). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure.

Mallia, D. (2013). When was the Internet Invented. *History News Network*. Retrieved from http://hnn.us/article/142824

Microsoft. (2013). Parts of the Access Control Model. *Access Control Lists*. Retrieved from http://msdn.microsoft.com/library/windows/desktop/aa374872(v=vs.85).aspx

Murdoch, S., & Anderson, R. (2008). Tools and Technology of Internet Filtering. *Access Denied: The Practice and Policy of Global Internet Filtering*, *1*(1), 58.

Neustar. (2012). Neustar® Insights: Online Fraud Prevention: Three Who Stood Their Ground, Available at: http://www.banktech.com/whitepaper/download/showPDF?articleID=191705583

Raynal, F., Ahmad, M., Shaikhli, I., & Ahmad, H. (2012). Protection of the Texts Using Base64 and MD5. *Journal of Advanced Computer Science and Technology Research*, *2*(1), 22–34.

Reed, M. G., Syverson, P. F., & Goldschlag, D. M. (1998). Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, *16*(4), 482–494. doi:10.1109/49.668972

SASI. (2006) *Internet Use 1990*, Poster of Internet usage, Available at: http://www.worldmapper.org/posters/worldmapper_map335_ver5.pdf

Sourcefire. (2013) Snort User's Manual 2.9.5, *The Snort Project*, May 2013. Available at: http://s3.amazonaws.com/snort-org/www/assets/166/snort_manual.pdf

Stanger, J., Krishnamurthy, M., Seagren, E., Alder, R., Bayles, A., Burke, J., & Faskha, E. et al. (2007). *How to Cheat at Securing Linux. Introducing Intrusion Detection and Snort*. USA: Syngress.

Thomas, K., Grier, C., Ma, J., Paxson, V., & Song, D. (2011) Monarch: Providing real-time URL spam filtering as a service. In *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, CA (pp. 447-462).

ENDNOTES

1 http://sourceforge.net/projects/phpproxy/
2 http://sourceforge.net/projects/phpproxy/files/stats/timeline?dates=2012-11-12+to+2013-11-12
3 http://www.motobit.com/util/base64-decoder-encoder.asp
4 http://www.jmarshall.com/tools/cgiproxy/
5 http://www.string-functions.com/hex-string.aspx
6 http://www.glype.com/
7 http://www.proxysiteslist.net/category.php?id=45
8 https://www.torproject.org/projects/torbrowser.html.en#downloads
9 http://www.digicert.com/ssl.htm
10 http://help.yahoo.com/l/us/yahoo/smallbusiness/store/risk/risk-18.html
11 http://www.joomla.org/about-joomla.html
12 http://extensions.joomla.org/extensions/access-a-security/site-access/ip-blocking
13 http://www.senionlab.com/
14 http://www.polestar.eu/en/
15 https://www.indooratlas.com/
16 http://arstechnica.com/security/2010/03/googles-new-gmail-geolocation-feature-aims-to-prevent-scams/
17 http://janav.files.wordpress.com/2013/05/base64chars.jpg
18 http://www.w3.org/html/wg/drafts/html/master/webappapis.html#atob
19 http://www.snort.org/
20 http://base.secureideas.net/about.php
21 http://corelabs.coresecurity.com/index.php?module=Wiki&action=view&type=tool&name=Pcapy
22 http://www.regular-expressions.info/python.html
23 http://docs.python.org/2/library/idle.html
24 https://www.torproject.org/projects/torbrowser.html.en
25 http://sourceforge.net/projects/poxy/?source=recommended
26 https://www.glype.com/download.php
27 http://www.jmarshall.com/tools/cgiproxy/
28 http://www.wireshark.org/download.html
29 http://www.jmarshall.com/tools/cgiproxy/
30 http://list.glype.com/

Jonathan McKeague is a graduate in Computer Science from Ulster University

Kevin Curran is a Reader in Computer Science and group leader for the Ambient Intelligence Research Group. Dr Curran has made significant contributions to advancing the knowledge of computer networking evidenced by over 800 published works. He is a regular contributor to BBC radio & TV news in the UK and quoted in trade and consumer IT magazines on a regular basis. He is an IEEE Technical Expert for Security and a member of the EPSRC Peer Review College.