Evaluating server-side internet proxy detection methods

An MSc thesis presented to the Cybersecurity Academy[*]
in partial fulfilment of the requirement for the degree of
Master of Cybersecurity

Author: J.M. (Hans) Hoogstraaten

Supervisors: Dr ir. P. Burghouwt (Pieter)
Prof. Dr ir. J. van den Berg (Jan)

Version: 1.1 final
Date: 10 November 2018

[*] The Cybersecurity Academy is a joint initiative of Delft University of Technology, Leiden University, and The Hague University of Applied Sciences.

CONTENT

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In today's use of the internet it is popular to obfuscate the originating Internet Protocol (IP) address to the receiving end of the internet connection. This can be achieved by using an intermediate service that forwards the user request (by proxy) to the destination on his behalf and returns the responses to him.[†] Typically this is done using a small overlay network like a VPN or a Socket Secure (SOCKS) proxy.

There are situations where people, observing connections on the internet, want to know if these incoming connections originate from a proxy or come directly from a client. An example of such a situation is when an online streaming video service places restrictions on specific geo-locations, or when illicit cyber activities are investigated. Identifying incoming connections as a proxy or a direct connection is a first step in enforcing policy or investigating cybercrime.

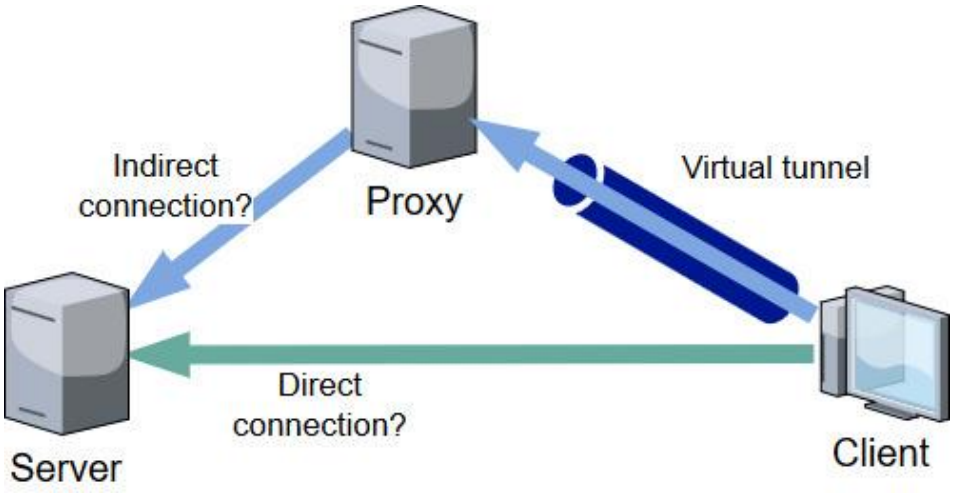

Figure 1 Is the connection originating from a client directly or was it relayed through a proxy?

1.1 PROBLEM DESCRIPTION

It is difficult to decisively determine if the originator of an internet connection is a direct client or an intermediate proxy device. There is no single method that is capable of identifying this for all possible proxy types and implementations. Although there are various detection techniques, it has been acknowledged that these methods all have advantages and disadvantages and can at best only detect specific proxy implementations (Pannu et al. 2016). If we assume that no single technique is capable of detecting all proxy implementations, the best detection method would be one where several techniques are combined. However, there is currently no structured overview of the available detection techniques and their capabilities and limitations. The work at hand closes that knowledge gap by providing this overview. This knowledge aids the design of effective detection systems that detect the use of a proxy from the perspective of a server in the internet.

Proxy techniques can be implemented in numerous ways, not least with the aim to avoid detection. Motivated by social and economic drives, people create proxy services that are hard to

[†] In some literature the term 'stepping stone' is used to indicate a proxy server.

distinguish from direct client connections. Likewise, there are similar drives that motivate endeavours to detect proxies. This 'cat-and-mouse' game makes the detection of proxies a dynamic process: detection methods that worked previously are not guaranteed to work in future situations, and new proxy implementations give cause to new detection methods.

1.2 RESEARCH OBJECTIVE

The objective of this work is to create a structured overview of the available proxy detection techniques as well as their capabilities and limitations. In order to achieve this, the following research questions are answered:

1. What are the current commonly used proxy techniques?
2. What are the *theoretical* capabilities and limitations of detection techniques?
    a. What techniques can be used to detect a proxy in general?
    b. How can a detection *technique* be implemented into a more practical detection *method*?
    c. What are the distinguishable properties of a detection method?
    d. What are the reasoned critical success factors of the detection methods?
3. What are the *practical* capabilities and limitations of detection methods?
    a. Can the theorised success factors be met in practice?

This study takes the perspective of the server receiving the connections. The point of departure is the (captured) network traffic on the receiving server.

1.3 RESEARCH DESIGN

The overview of proxy detection methods with relevant properties is developed by taking the following steps:

1. First, for a better understanding of the problem, the context is described in detail. The topic of discussion is conceptualised and the domain and the scope of this study are defined (Chapter 2).
2. Then, an overview of today's popular available proxy techniques is created (Chapter 3). This will answer research question 1.
3. Next, an inventory of various detection techniques is presented (Chapter 4). For each technique, the thesis:
    a. describes how it works;
    b. assigns selected properties;
    c. provides reasoned caveats and foreseen limitations; and
    d. present a list of critical success factors.

    This will answer research question 2.
4. The next step is to explore and experiment on the feasibility of selected success factors, and to test if and how the detection techniques can be used in practice (Chapter 5). This will answer research question 3.

5. Based on the test results and experiences gained during this research, the capabilities and limitations of each of the identified detection methods are assessed (detailed in Chapter 5, summarised in Chapter 6). This will answer the main research question.
6. Finally, conclusions are drawn and suggestions for improvements and further research are presented (also in Chapter 6).

This study clearly distinguishes between a theoretical and an empirical approach. The approach in step 3 (Chapter 4) is mainly by reasoning from theory, while the approach in step 4 (Chapter 5) is focused on experiments.

1.4 SCOPE

The objective of this study is to detect whether a proxy is used by a client by examining incoming connections. The scope of this study is not to identify the real originator of the connection, for example the IP address of the client. Moreover, this study adopts the server's perspective of the proxy connection, and does not focus on the communication between the client and the proxy. This distinguishes this study from studies that focus on detecting tunnels and (illicit) connections, for example malware communication with a Command and Control server, or illegal VPNs from within a company network.

2 DOMAIN DESCRIPTION

This chapter describes the domain of internet proxies and their detection. It outlines a conceptual framework and provides context to this study by establishing definitions.

2.1 PROXY CHARACTERISTICS

2.1.1 Overlay network/ Proxied connections

When people use the internet, they make their software client connect to a server on the internet. In this client-server model the server has a public IP address that a client can connect to. In order to create a connection, the IP packets are routed over the internet by the client's Internet Service Provider (ISP) and several network operators, or autonomous systems, between the client and the server. This routing of IP packets over the internet is beyond the control of the client because it is subject to agreements between the network operators (Staff 2008). However, when a proxy server is used, the client does, to some extent, have control over the routing.

The network packets between the client and the proxy are encapsulated and transported to the proxy server. However, the proxy is not the final destination. Effectively this creates a network tunnel or an overlay network (Tanenbaum and Wetherall 2011, para. 5.5.3). Because the user selects a proxy, this routing through a proxy is within the control of the user, therefore the tunnel is on top of the internet protocol.[‡] A consequence of the use of a proxy is that the source IP address of the client no longer needs to be exposed to the server. This provides the possibility to obfuscate the original IP of the client to a server on the internet.

2.1.2 Pseudo source address configurations

The source IP address, the proxy is using to connect to the destination server on behalf of a client, can be regarded as the pseudo-source address of the client: the source IP address a server observes is not the real address of the client, but a pseudonym.

A proxy server can be used by a single or by multiple clients at the same time. In addition, a proxy can have one or multiple IP addresses at its disposal. These options make various configurations possible. For example, a pseudo source IP address can be shared with other clients. A proxy then has to keep a status of the connected clients to map them to the proxied outgoing connections. It is also possible to use an IP address dedicated to a single client (for a period of time).

2.1.3 Multiple IP address setup

In a single IP address proxy setup, the IP address the client connects to, is also used as the pseudo source address. For example, in Figure 2 the client creates a tunnel between its IP address 1.1.1.1 and the IP of the proxy 2.2.2.2. The proxy then makes connections to the server using the pseudo IP address 2.2.2.2, which is the same address as that used for the tunnel endpoint.

[‡] It is assumed here that all clients connect through an IPv4 gateway to the internet, either using Ethernet, WiFi, mobile data, or some other link layer.

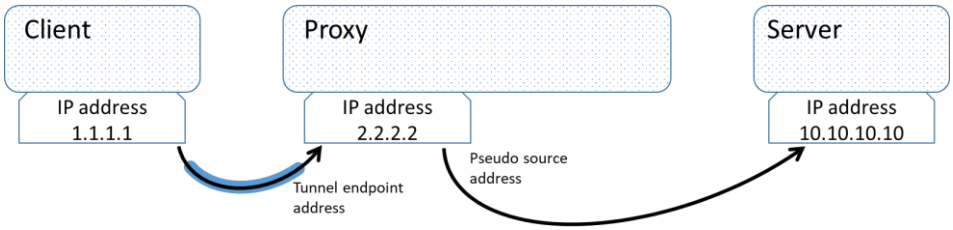

Figure 2 Single address proxy setup

When a proxy has multiple IP addresses at its disposal, the proxy server can also be configured to separate the addresses the client connects to from the outgoing pseudo source addresses. In that case the proxy software does not listen to incoming client connections on the IP that is also used as a pseudo source address (see Figure 3). This setup can be expanded where the proxy has multiple separated outgoing IP addresses. It should be noted that a proxy can also consist of several physical servers that have the same functionality and form a proxy network (see also Paragraph 2.1.7).

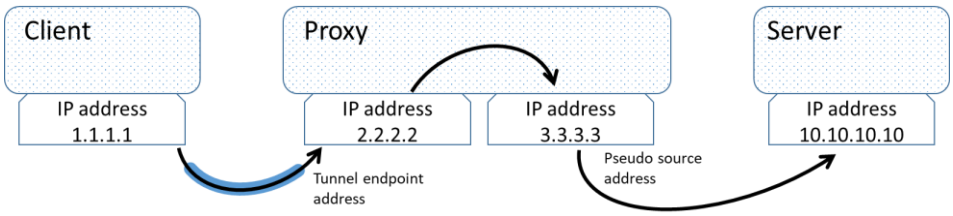

Figure 3 Separated address proxy setup

2.1.4 Proxying at different protocol layers

Proxies can be created on three different levels in the TCP/IP stack: on the application layer (OSI layer 7), on the internet or network layer (OSI layer 3), and on the link layer (OSI layer 2)("Proxy Server" 2018).[§**]

- **Application-layer proxies.** When a client uses an application-layer proxy, the *application* of the client sends its data to a proxy. The proxy configuration of the client is done in the user's application. For example, when a web browser is configured to use a proxy, all HTTP traffic is passed through that proxy. The tunnel is setup between the client application and the proxy. With application-layer proxies, from the perspective of the client, it is as if the application of the client is running on the proxy. Application-layer proxies make it possible for other applications on the user's computer to avoid using the proxy.
- **Internet-layer proxies.** When a client uses an internet-layer proxy, the client creates a virtual IP interface that directly connects to the proxy by a tunnel network. With internet-layer proxies, from the perspective of the client, it is as if the client is running on the same network as the proxy. The operating system of the client has a (virtual) IP interface, and all protocols on top of IP are routed through the proxy. All applications on the user's computer that communicate with the same destination make use of the proxy. It is

[§] The TCP/IP reference model of Tanenbaum is used (Tanenbaum and Wetherall 2011). The numbers refer to the ISO Open System Interconnection (OSI) standard.

[**] See also Paragraph 6.3: "Future work".

possible to make exceptions for specific IP addresses to not to use the proxy (also called split tunnel).

- **Link-layer proxies.** When a client uses a link-layer proxy, the client creates an Ethernet interface that directly connects to the proxy by a tunnel network. With link-layer proxies, from the perspective of the client, it is as if the client is running on the same network segment as the proxy. All low-level communication makes use of the proxy, including ICMP and ARP protocols.

2.1.5 Routing and other services at the proxy

When a client connects to a proxy, the tunnelled connections can be routed in different ways. For example, a dual-homed proxy server can give access to an internal network that is normally not accessible from the internet. A typical example of this is the remote VPN of a company, where an employee connects to the proxy to access internal servers and the internet as if he or she were physically present in the office. Another common configuration is where connections of a proxy client are directly routed to the outgoing gateway of the proxy. In this case, a client cannot connect to any device on the local network of the proxy. This is a typical configuration for a (commercial) proxy service.

In addition to the routing, the proxy has additional control over the tunnelled connections. For example, the connections can be filtered by firewalls or access control systems, analysed by anti-malware or intrusion detection systems, protocol can be scrubbed to remove identifiable fingerprints, and contents can be cashed for faster downloads.

2.1.6 Authentication and encryption

When a proxy provider wants to place limitations on the users of the proxy, some form of access control can be added. For example, a commercial proxy provider only allows paying customers access to the proxy service. When a client wants to use the service, he or she needs to authenticate him or herself, for example with a username/ password combination or a certificate.

Proxies can also be secured with encryption. In that case the communication between the client and the proxy is protected with encryption to ensure the confidentiality of the communication (secure tunnel). When a proxy uses encryption, the proxy is usually called a VPN. Note that for this study it makes no difference whether authentication or encryption is used or not. The terms VPN, proxy and overlay network are used interchangeably.

2.1.7 Internal structure overlay network

An overlay network can consist of a single node as in the previous examples, or can be made using multiple nodes. With multi-node proxies, the tunnel extends over more than one server. Commonly the first node of the tunnel, to which the client connects, is referred to as entry-node, whereas the last node, which determines the pseudo-source address, is called exit-node.

In addition, the tunnel itself can also consist of an overlay network (overlay-upon-overlay). This so-called chaining of proxy techniques, or tunnel-in-tunnel, creates extra obfuscation and security for the client. Various techniques can be chained together; the packets of the client can for example be split up into smaller packets, encoded to base64 text, transferred as DNS requests, reassembled

and sent over an OpenVPN connection, into a Tor network connection, printed out on a screen, read by a camera, and finally sent to the internet, en route to its final destination (although this setup would cause significant latency).

2.1.8 Reverse proxy/ port forwarding

Some proxy configurations allow the user to forward incoming connections back to the client through the tunnel. One or more TCP or UDP ports can be configured to be forwarded back to the client once the tunnel has been established. An internet user can then connect to that port on the pseudo source address and the connections will pass through the tunnel to the proxy client. This effectively creates a reverse proxy. Note that this can potentially be a security risk, especially if the proxy or VPN service provider does this by default without the user's knowledge.

2.1.9 Ownership

There is a difference between proxy services that are owned by proxy providers (either commercial or free) and proxies that are privately owned. This difference is the respective exposure. Commercial or free proxy providers expose their services and proxies to a wider public to attract more (paying) clients. It therefore possible to find these providers by searching internet pages. On the other hand, privately owned proxies are not advertised.

Many tutorials explain how to create a proxy using a cloud server. For example, a cheap Virtual Private Server (VPS) can be rented from Vultr.com, DigitalOcean or Amazone EC., and VPN packages can be installed on it such as Algo VPN ("Meet Algo, the VPN That Works" 2016). It is also possible to setup a proxy on a server located at home or on home routers.

2.2 QUANTIFYING ACCURACY

It is recognised that proxy detection methods do not provide absolute certainty (Pannu et al. 2016). Still, there is a need to evaluate individual detection methods and compare them with each other. Therefore some criteria must be chosen and preferably quantified. In an ideal situation, the accuracy of a test is represented in a single metric. This would require a dataset representing all proxy and all direct connections. However, gathering a presentative set of known direct connections has proven very difficult. Therefore, the accuracy of a detection method is represented with multiple metrics.

Furthermore, choosing the proper criteria for evaluating the detection methods depends a great deal on the environment in which they are used. An example analogy can be a method for weighing apples that has a measuring accuracy of 1 gram. Although this is a convenient accuracy in some situations, it is an irrelevant criterion for market vendors. By analogy, it is appropriate to look at some practical issues.

2.2.1 Estimating the population: a priori odds

The main question this study tries to answer is:

Given an incoming TCP/IP connection φ at the server: is it coming from a proxy used by a client ($\mathcal{H}_p$: $\varphi = \varphi_{proxy}$), or directly from a client ($\mathcal{H}_d$: $\varphi = \varphi_{direct}$)?

In order to identify what the important factors in determining the accuracy of detection it is expedient to have a sense of the order of magnitude a connection can be a proxy. Therefore, an educated guess is made on the a priori odds $\frac{P(\mathcal{H}_{proxy})}{P(\mathcal{H}_{direct})}$. The following numbers are estimated:

- The total number of IP addresses that can make a connection from the internet. This is estimated as the total number of IPv4 addresses possible ($2^{32} \approx 4.3 \cdot 10^9$), minus the number of reserved IP addresses ($\approx 0.6 \cdot 10^9$) ("Reserved IP Addresses" 2018). This results in an estimated $3.7 \cdot 10^9$ usable IP addresses.
- The total number of proxies on the internet. Based on the number of proxies in a reliable database that is limited to publicly known proxies (IP2Location n.d.) ($\approx 0.2 \cdot 10^6$ proxies), the estimate of existing proxy IP addresses is $20 \cdot 10^6$.

This results in a rough estimation of the odds of a random IP address connecting to an internet facing server that is a proxy:

$$\frac{P(\mathcal{H}_p)}{P(\mathcal{H}_d)} = \frac{20 \cdot 10^6}{3.7 \cdot 10^9 - 20 \cdot 10^6} = 1{:}\,184 \approx 0.0054$$

This shows that proxies are relatively rare. Therefore, for any technique to significantly contribute to detection, the false positive rate should be low (Buchanan 2007).

2.2.2 False positives

False positives, also called type I errors, are known to have an misleading effect on large data sets with few true positives (Buchanan 2007). Given the estimate on the a priori odds of the total population in the environment considered here, this misleading effect can easily occur. This is illustrated using the following example:

A proxy test has a false positive rate of $FPR = \frac{number\ of\ false\ positives}{total\ number\ of\ negatives} = 0.01$. This seem to be a fairly good test based on the FPR. If the population consists of all the usable IP addresses, the following can be calculated:

$$Number\ of\ falsly\ identified\ proxies:$$
$$(population - \#proxies) \times FPR =$$
$$(3.7 \cdot 10^9 - 20 \cdot 10^6) \times 0.01 = 36.8 \cdot 10^6$$

This means that the number of incorrectly identified proxies is almost twice the number of existing proxies!

If the test is able to detect all the proxies ($TPR = \frac{number\ of\ true\ positives}{total\ number\ of\ positives} = 1$), the following can be calculated using Bayes' theorem:

$$P(\varphi_p \mid TestResult_{pos}) =$$
$$\frac{P(TestResult_{pos} \mid \varphi_p)\, P(\varphi_p)}{P(TestResult_{pos} \mid \varphi_p)\, P(\varphi_p) + P(TestResult_{pos} \mid \varphi_d) P(\varphi_d)} =$$

$$\frac{1.0 \cdot 0.0054}{1.0 \cdot 0.0054 + 0.01 \cdot (1 - 0.0054)} \approx 0.35$$

Which is equivalent to:

$$\frac{P(\mathcal{H}_p)}{P(\mathcal{H}_d)} = \frac{0.35}{1 - 0.35} = 0{,}54 \Rightarrow IP\ is\ (always)\ a\ direct\ connection$$

The example shows that a test with a low false positive rate is still not able to distinguish a proxy from a direct connection. In the example all tested IPs are classified as direct connections. As the rate between proxies and non-proxies is very small, special attention should be placed on the false positive rate to assess a detection method. Even if the false positive rate of a detection method is $FPR = 0.00543$, a positive result still means that at best there is only a 50:50 chance that it is actually a proxy. Therefore, the ratio between true positive (TP) and false positive (FP) results is a good indication for the accuracy of the test: the positive predictive value (PPV):

$$PPV = \frac{number\ of\ true\ positives\ (TP)}{number\ of\ positive\ test\ results\ (P)}$$

2.2.3 Available datasets

Positive and negative predictive values can be calculated when a representative labelled dataset is available (Borovicka et al. 2012). Unfortunately, this has proven to be a very difficult task. This is mainly because there are many different implementations to a proxy technique that are unknown, and because the availability of representative direct connections was limited. Therefore, the predictive values could not be determined. Instead, the True Positive Rate (TPR) is determined using biased or unilaterally labelled datasets. In some cases, a qualitative indication can be given on the false positives rate (FPR) by reasoning based on the theory of the detection technique.

2.2.4 False negatives

False negatives, or type II errors, are test results that are negative while in fact they should be positive. In this case a connection is a proxy, while the result of the test is that it is not a proxy. In the context of detection proxies these false negatives are mainly due to a lack of information and knowledge. By analogy, a medical doctor diagnoses a patient by eliminating known diseases, taking the observed symptoms into account. However, if no illness is detected, he or she cannot know for certain that the patient has no illness. Similarly, most detection techniques in this thesis are based on knowledge of known proxies: a detection technique can determine if a connection is a proxy, but if the test is negative it is still unknown if the connection is a proxy or a direct connection.

If all tests are negative, you might conclude the connection is not a proxy by elimination. However, the certainty that the connection is not a proxy depends on the coverage of the combined tests. The more knowledge on proxies is incorporated in the tests, the more certain the combination of tests becomes and the more certain it is that a negative result is a true negative.

For each detection method these false negatives are described as *omissions* of the method. These omissions are reasoned from theory. Detection methods that are based on non-statistical techniques, such as lookup lists or similar 'signature based' techniques, are particular prone to this type of omission.

2.2.5 Other criteria

You could argue that a good detection technique is one that detects many proxies. This however depends on the popularity of specific techniques or implementations. For example, if 90% of all proxy connections are through a Tor proxy, a Tor proxy detector is a useful and overall more valuable detection method. The segment of the total proxy connections that are detectable by a detection method is a good metric for assessing the quality. However, this metric is only a hypothetical one, because if a proxy method cannot be detected, total proxy population cannot be determined. Practically however, the popularity of a technique or implementation can be determined by other means, for example by the amount of attention it receives on internet fora.

Another criterion is the trustworthiness of the knowledge or data used for a detection technique. This is particularly the case when relying on data from third parties such as databases. For example, a detection method based on a timing technique that was extensively tested is more reliable than a lookup method that uses a free database that is maintained by volunteers. Although this is not straightforward, a qualifying metric can indicate this quality aspect of the detection method.

2.3 SUMMARY

This chapter describes the domain of internet proxies and their detection. Proxies can be assigned various characteristics. Clients on the internet can use a proxy to make their connection take a more controlled route over the internet. The connection between the client and the proxy is a virtual network, also called **overlay network** or **proxy tunnel**. The IP address the client appears to use to connect to a server is the **pseudo-source address** of the client. A proxy can have one or more IP addresses at its disposal, creating various configurations. On a proxy with a single IP address the tunnel endpoint address is the same as the pseudo source address. On a proxy with multiple IP addresses the tunnel endpoint could be a different IP address than the outgoing pseudo source address.

Proxying the traffic can be done on the three different layers in the TCP/IP protocol stack: the application layer, the internet layer, and the link layer. In addition, the traffic of the client can be routed in various ways at the proxy before it is sent on its way. The traffic can also be filtered or cached at the proxy.

To access and use a proxy it is possible to have an access control mechanism in place on the proxy service. It is also common to encrypt the overlay network to prevent eavesdropping. A proxy can consist of a single machine or node, or can be a configuration of multiple nodes. In that case the overlay network expands over these nodes. The first node of the tunnel is referred to as **entry-node**, while the last node is the **exit-node**.

Once a client has setup a proxy tunnel the traffic can also be initiated through the tunnel in reverse, creating a **reverse-proxy**. The TCP and UDP ports on a proxy are forwarded back through the tunnel to the client (**port forwarding**). Another distinction between proxies relates to ownership.

Proxies can be owned by commercial or free **proxy providers**, making their service available for their clients. A proxy can also be setup and created by an individual. In that case the proxy is **privately owned** or controlled.

In order to quantify the accuracy of a proxy detection method, it is imperative to take the context into account. Because of the relatively few proxies on the internet compared to the total number of usable IP addresses, the **false positive rate** is an important metric to indicate the accuracy of a detection method. It turns out that it is very difficult to create a representative dataset to test the accuracy of the detection methods in the best way possible. Therefore, biased or unilaterally labelled datasets are used to determine the **true positive rates**, partly representing the accuracy of the methods.

Most detection methods are based on knowledge of known proxies. A negative test result of a detection method therefore does not mean the tested connection is not a proxy. These **omissions** or false negatives are identified for each detection technique.

3 INVENTORY OF PROXY TECHNIQUES

To be able to detect a proxy it is important to know how the proxy techniques work. Tunnelling the data of a client and forwarding them on behalf of a user can be done in many different ways. For example, calling a friend and asking him to buy some concert tickets on the internet for you can also be considered as using an internet proxy. The techniques that can be used are in that sense only limited to the imagination of the people creating a proxy tunnel. In the next paragraphs the most well-known and frequently used proxy techniques are described.

3.1 HTTP PROXY

A Hypertext Transfer Protocol (HTTP) proxy handles HTTP requests on behalf of a client application. A user application creates a request and sends it to the HTTP proxy. The HTTP proxy receives the request, unaware of the method used to transport the data; the tunnel between client and proxy is created between the two applications. Any lower level protocol features, such as fragmentation or DNS, are not visible to the applications and therefore not transported through the proxy. Only complete application data frames are sent through the proxy.

The HTTP protocol is described in several Request For Comments (RFCs) standards (RFC7230 through RFC7240, RFC7615, and RFC7616). Information from the originating client or the proxy is often added by the HTTP proxy for debugging and statistics. The proxy does this by adding extra lines to the HTTP header, sometimes with the original IP address of the client. For example, the following headers are added by HTTP proxies:

- FORWARDED;
- X-FORWARD-* as defined by RFC 7239 (Petersson and Nilsson 2014);
- X-FORWARDED-FOR Identifies the originating IP addresses of a client;
- X-FORWARDED-HOST Identifies the original host requested that a client used to connect;
- VIA provides information about the proxy itself as defined by RFC 7230 (R. Fielding and J. Reschke 2014).

In addition, some proxy software or appliances add specific and non-standardised lines to the HTTP header (see also Appendix I and par. 4.7).

An HTTP proxy often has a store-and-forward functionality, meaning that the content is cached at the proxy in order to increase the speed for other users. It is possible for a client to start a Secure HTTP (HTTPS) connection to a HTTP proxy. In that case the encrypted tunnel is terminated at the proxy and a new encrypted tunnel is initialised between the proxy and the server. This means that there is no end-to-end encryption between the client and the server. However, there is an HTTP CONNECT method described in the standards that can facilitate end-to-end encryption, although most servers have not implemented this feature (Fielding and Reschke n.d., para. 4.3.6).

The HTTP proxy can be configured in the client application of the user. The application then uses this proxy instead of connecting to the destination server directly. Commonly the TCP port 8080 is used for HTTP proxies. In some implementations, it is possible to browse to a proxy server where the proxied web page is displayed in a frame on that page. This user-friendly method is often used by free HTTP proxy services.

Popular HTTP proxy software are Squid, TinyProxy, and node-http-proxy. In addition, there are many free and commercial HTTP proxy services available. The free services often change their domain name and can be found using index pages, for example on http://www.publicproxyservers.com.

3.2 Socket Secure proxy (SOCKS)

The socket secure (SOCKS) protocol was originally developed by David Koblas and published in 1992 (Koblas 1992). There are currently three major SOCKS version in use: SOCKS4, SOCKS4a and the latest version SOCKS5 that is described in RFC1928 (Leech 1996). The version SOCKS4 is especially designed to allow proxying of TCP traffic to a specific IP address. The version SOCKS4a extends the protocol by allowing a client to specify a destination domain name rather than an IP address, even if the client cannot resolve the IP address. The latest version SOCK5 offers more choices for authentication and adds support for IPv6 and UDP ("SOCKS" 2018).

A SOCKS client informs the proxy server about the connection the client tries to make. The proxy then acts as transparently as possible. This in contrast to a HTTP proxy, which tries to interpret and rewrite headers. For example, a SOCKS proxy simply forward the HTTP headers without altering them. A SOCKS proxy can forward any TCP. UDP and IPv6 connection. The SOCKS service commonly uses TCP port 1080.

3.3 Point-to-Point Tunnelling Protocol (PPTP)

The point-to-point tunnelling protocol was developed by Microsoft and published in July 1999 as RFC 2637. It is the oldest and the most commonly used protocol on both LAN and WAN networks. Although the traffic can be encrypted it is regarded as insecure (Constantin 2012). PPTP is considered one of the fastest proxy protocols because of its simplicity and fast encryption. PPTP uses the TCP port 1723 for communication.

3.4 Secure Socket Tunnelling Protocol (SSTP)

The Secure Socket Tunnelling Protocol (SSTP) was developed by Microsoft and first published in 2007. The SSTP allows encapsulation of Point to Point Protocol (PPP) traffic over HTTPS. As a result, SSTP uses both PPP and HTTPS for operation ("[MS-SSTP]: Secure Socket Tunneling Protocol (SSTP)" n.d.). SSTP was intended only for remote client access; it generally does not support site-to-site VPN tunnels. SSTP suffers from the same performance limitations as any other IP-over-TCP tunnel. In general, performance will be acceptable only as long as there is sufficient excess bandwidth on the un-tunnelled network link to guarantee that the tunnelled TCP timers do not expire. If this becomes untrue, performance declines dramatically ("Secure Socket Tunneling Protocol" 2018).

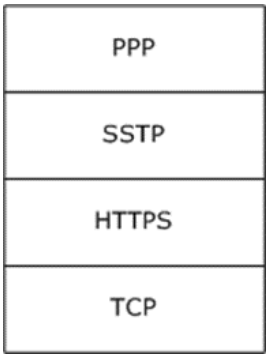

Figure 4 The SSTP protocol stack[††]

3.5 L2TP

The Layer-2 tunnelling protocol (L2TP) was published in 1999 as RFC2661. The L2TP protocol is a combination of the PPTP protocol and the Layer-2 Forwarding Protocol (L2F). A new version of this protocol, L2TPv3, appeared as the proposed standard RFC 3931 in 2005. L2TPv3 provides additional security features, improved encapsulation, and the ability to carry data links other than simply Point-to-Point Protocol (PPP) over an IP network, such as Frame Relay, Ethernet, and ATM ("Layer 2 Tunneling Protocol" 2018).

The two endpoints of an L2TP tunnel are called the LAC (L2TP Access Concentrator) and the LNS (L2TP Network Server). The LNS waits for new tunnels. Once a tunnel is established, the network traffic between the peers is bidirectional. Either the LAC or LNS may initiate sessions. The traffic for each session is isolated by L2TP, so it is possible to set up multiple virtual networks across a single tunnel. L2TP only provides encapsulation and does not provide encryption, integrity or authentication. The packets are transported over UDP, and typically port 1701 is used.

3.6 OpenVPN

OpenVPN is an open-source VPN project created by James Yonan and was initially released in May 2001. OpenVPN runs a custom security protocol based on SSL and TLS. Implementations use the OpenSSL library for authentication and encryption. OpenVPN has many configurable features, making it a versatile proxy application. The tunnel between client and the proxy can run over the UDP or TCP protocol. OpenVPN offers two types of interfaces for networking via the Universal TUN/TAP driver. It can either create an internet-layer proxy (TUN), or a link-layer proxy (TAP) that can carry any type of Ethernet traffic. Port 1194 is the official IANA assigned port number for OpenVPN ("OpenVPN" 2018).

[††] Figure taken from ("[MS-SSTP]: Relationship to Other Protocols" n.d.)

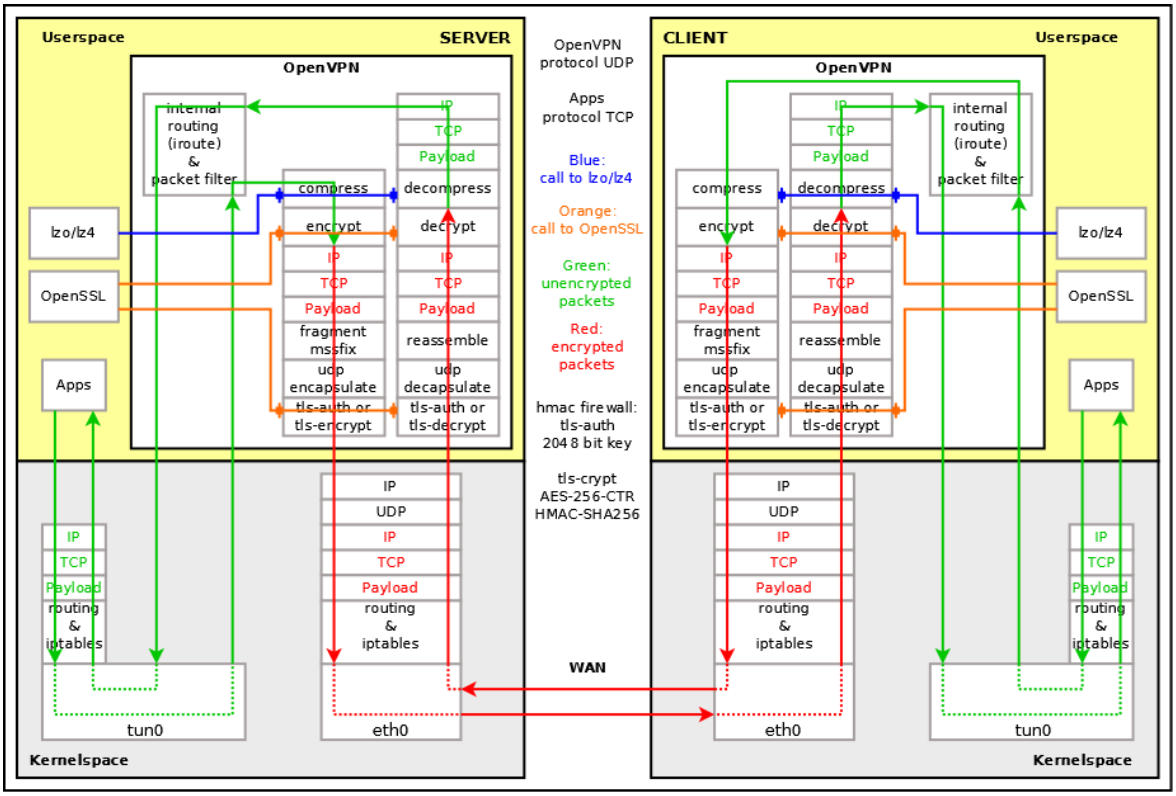

Figure 5 The traffic flow of OpenVPN[‡‡]

3.7 IPSEC

The internet Engineering Task Force (IETF) IP Security Working Group published a number of security protocols developed for the internet under the name IPsec in 1995. The IPsec protocol suite was published in RFC-1825 through RFC-1827 and later extended and updated in many other RFCs ("IPsec" 2018).

An IPsec tunnel is created by first creating Security Associations (SA) between the client and the proxy. Once the SA is established, data can be transported by using either the Authentication Header protocol (AH) or the Encapsulating Security Payload protocol (ESP). The authentication header protocol does not provide encryption but only encapsulate the packet. It also provides an integrity functionality of the new outer IP header. Because of this, the AH protocol does not work when the IP header is changed somewhere in the line, for example by the Network Address Translation protocol (NAT). The AH protocol is therefore not a popular protocol. However, the Encapsulating Security Payload protocol does provide encapsulation for encryption and authentication of user traffic.

The IPsec protocol suite includes a framework for authentication and key exchange: the Internet Security Association and Key Management Protocol (ISAKMP). The most popular key exchange protocols are the Internet Key Exchange protocols (IKE) used to establish the Security Association (SA) between the client and the proxy. The first version of this key exchange protocol (IKEv1) consists of two phases. In the first phase a management channel is created for the health status of the proxy and the like. In the second phase two uni-directional Security Associations (SA) are established.

[‡‡] Unknown author. Taken from http://nl.tinypic.com/view.php?pic=rlizcg&s=9.

The IKEv2 protocol is the next generation standard for secure key exchange between client and proxy and is defined in RFC5996. In this protocol the SA is established without phases by creating child SAs within the same negotiation.

Both AH and ESP protocols can be used as link-layer proxies (transport mode) and internet-layer proxies (tunnel mode). Commonly used ports are UDP port 500 for IKE and UDP port 4500 for NAT-traversal. When used as link-layer proxy (transport mode) the IP protocols 50 (AH) and 51 (ESP) are used.

In practice the IPsec protocol suite is used in two ways: IPsec over L2TP and IKEv2.

3.7.1 IPsec over L2TP

Before the IKE version 2 was developed it was common to use the L2TP protocol to setup the VPN tunnel, and to only use IPsec for the encryption. With the IKEv2 protocol the L2TP protocol is no longer needed, however most proxy providers still offer IPsec in combination with L2TP, commonly referred to as L2TP/IPsec ("Layer Two Tunneling Protocol" n.d.).

3.7.2 IKEv2

Most commercial proxy providers offer IKEv2 protocol. It is considered to be faster than PPTP and L2TP and offers the highest security. Most operating systems provide native support for this protocol.

3.8 The Onion Routing (Tor)

One of the most popular proxy services is The Onion Routing (Tor), which is based on the software maintained by The Tor Project Inc. Tor directs internet traffic through a free, worldwide, volunteer overlay network consisting of more than seven thousand relays ("Tor (Anonymity Network)" 2018).

Onion routing is implemented by encryption in the application layer of a communication protocol stack, nested like the layers of an onion. Tor encrypts the data, including the next node destination IP address multiple times, and sends them through a virtual circuit comprising successive, random-selection Tor relays. Each relay decrypts a layer of encryption to reveal the next relay in the circuit to pass the remaining encrypted data on to it. The final relay decrypts the innermost layer of encryption and sends the original data to their destination without revealing or knowing the source IP address.

Not only does the Tor network provide a proxy service to the internet, it also includes hidden services that can be hosted anonymously. These hidden services can only be reached using a Tor client. Typically, Tor uses TLS over TCP as its transport protocol. The well-known TLS port for Tor traffic is 443 TCP. Tor commonly uses TCP ports 9001 and 9030 for network traffic and directory information.

3.9 Other proxy techniques and implementations

Various other techniques and implementations exist to setup a proxy. While most use the commonly known techniques with different configurations, others use novel or proprietary techniques. Some examples:

- ICMP tunnelling. An ICMP tunnel establishes a covert connection between two remote computers (a client and a proxy), using ICMP echo requests and reply packets. An example of this technique is to tunnel complete TCP traffic over ping requests and replies ("ICMP Tunnel" 2018).
- DNS tunnelling. DNS tunnelling is the ability to encode the data of other programs or protocols in DNS queries and responses ("What Is DNS Tunneling?" 2016).
- Stunnel. Stunnel is an open-source multi-platform application used to provide a universal TLS/SSL tunnelling service. Stunnel can be used to provide secure encrypted connections for clients or servers that do not speak TLS or SSL natively. It runs on a variety of operating systems, including most Unix-like operating systems and Windows. Stunnel relies on the OpenSSL library to implement the underlying TLS or SSL protocol ("Stunnel" 2018).
- Shadowsocks. Shadowsocks is a free and open-source encrypted proxy project that is widely used in mainland China to circumvent internet censorship. Typically, the client software will open a SOCKS5 proxy on the machine where it is run, which internet traffic can then be directed towards. Additional features are implemented that make it hard to recognise (and block) the encrypted tunnel.

3.10 SUMMARY

To be able to detect a proxy it is important to know how the techniques work. Well-known and frequently used proxy techniques are:

- HTTP Proxy;
- Socket Secure proxy (SOCKS);
- Secure Socket Tunnelling Protocol (SSTP);
- Point-to-point tunnelling Protocol (PPTP);
- OpenVPN;
- IPsec over L2TP;
- IKEv2 IPsec;
- The Onion Routing (TOR).

4 INVENTORY OF DETECTION TECHNIQUES

Different techniques can be used to detect whether an incoming connection is a proxy or a direct connection. A technique can be used in different ways or on different data. Therefore, more concrete *methods* are derived from a general *technique*. This chapter includes an overview of detection techniques with corresponding detection methods. In addition, for each detection method, the properties are provided, as well as the critical success factors for each method to work in practice. The next chapter describes how some of these success factors are tested in experiments.

4.1 INTRODUCTION

The problem a detection technique has to solve is fairly simple: to distinguish between the situations $\varphi = \varphi_{direct}$ and $\varphi = \varphi_{proxy}$ by regarding incoming connections. It is however not an easy task, mainly because it is well conceivable that a proxy can be created where this is indistinguishable from the perspective of the server. Detection is possible when the proxy somehow reveals information that makes it distinguishable from a direct client connection.

The next paragraphs introduce several techniques or approaches that reveal information about the incoming connection's IP address. These techniques were found 'browsing the internet' looking for similar initiatives, proof-of-concept projects, and discussions on various fora. It was generally observed that no single technique stood out as being perfect. In fact, the general sentiment on various fora is that it is difficult, if not impossible, to detect all possible proxy types on the server side. It was also observed that there is very little scientific literature available on the subject.

4.1.1 Method properties

Utilising a detection method can be divided into a preparation and a detection activity or phase. In the preparation phase information and data are gathered and analysed to prepare for the detection. In the detection phase an incoming connection is exposed to testing. Based on the study of the methods, each detection method can be assigned general features that distinguish it from other methods. The following features are determined for each method:

- **Data source**. The type of data or information that is used as input to the detection method, for example the IP address or the packet latency.
- **Base assumption**. The assumption that forms the basis of the detection method.
- **Preparation**. A short description of the type of preparation that is needed for the method to be able to detect.
- **Operation**. A short description of how the method operates.
- **Statistical/ non-statistical**. The technique that is applied to the data in the preparation and the detection phase. A non-statistical technique is based on a blacklist or a signature; a lookup is performed, and a match generates a conclusion. In contrast, a statistical technique is based on empirical data that are statistically analysed. This is either by statistically evaluating multiple measurements of the tested connection, or by comparing measurements that were made beforehand in the preparation phase. In either case, statistically evaluated information forms the essence of the method.

- **Active/ Passive**. A detection method can be based on actively probing an incoming IP address to retrieve additional information. A method can also be purely passive or retrieve information from out-of-band sources like databases. This feature is determined separately for the perpetration phase and the detection phase.

In addition, each method can be characterised according to capabilities and limitations:

- **Hit**. This feature describes how a positive result should be interpreted; i.e. what does it mean when the method produces a 'hit' (i.e. the method determined it was a proxy)? This accuracy is based on the theoretical knowledge of the detection method.
- **Miss**. This feature describes the meaning of a negative test result. Due to the nature of the tests, in most cases a 'miss' (i.e. the method determined it was not a proxy) reveals very little information (see also Paragraph 2.2.4).
- **Omission**. By considering how a detection method works, its omission can also be recognised. While a 'hit' and a 'miss' describe what can be detected, the omission describes what the method cannot detect.

Finally, for each method to be able to function as intended in a practical situation, some conditions must be met. These are described as the critical success factors of the method. Some of these critical success factors have been tested; the results are presented in the next chapter.

- **Critical success factors**. The factors that are crucial for the detection method to work in practice.

4.2 LIST OF KNOWN PROXIES

For a user to be able to use a proxy, he must know where to connect to. Proxy providers must give the IP address where a client can connect to and what configuration to use. This information can be harvested and reverted into lists of known proxies. A list of the IP addresses that are known to be used as proxies makes detection very easy. When an IP address of an incoming connection is on such a list it is highly likely to be a proxy. This lookup technique is also known as a white/blacklist technique.

Different proxy providers and their (commercial) implementations view these lists with different sentiments. For example, for the popular proxy service Tor such a list is not a problem. In fact, the publication of this list is part of the design: any Tor client that initiates a proxy downloads the current available nodes, including if it functions as an exit-node. Other techniques and (commercial) implementations try to obfuscate their proxies in the best possible way and thus evade identification.

4.2.1 Temporality

The function of an IP address can be very fluid. When an IP address has been identified as a proxy, the next day this IP might have switched owner and can have a completely different function. This makes the blacklist detection performance time-dependant. To improve this method, additional features should be implemented such as a timestamp of the observation and an indication of its future validity, for example half-time value or some other time-dependant likelihood ratio.

4.2.2 Accuracy

For all detection methods involving lists of known proxy IP addresses it is assumed that a positive result of the test (the lookup) is unlikely to be false (the FPR is nearly zero). It is reasoned that a false positive could only occur if a typo was made, or if the IP address was removed from the influence of the proxy provider (e.g. reassigned to another user). The latter scenario means that the administration of the service is incorrect. It is assessed that these scenarios are unlikely to happen, and it is therefore deemed very unlikely that a false positive can occur.

When the lookup results in a negative (a 'miss' in the lookup), this by no means guarantees that the IP address was not a proxy. This is because the list of known proxies can in practice never be complete. In addition, privately owned proxies cannot be known with this method. Based on the nature of the test and the internet environment, it is assessed that a negative result provides no additional information to what is already known.

4.2.3 Properties and success factors

The following method is based on the above described technique:

- List of known proxies

Table 1 Properties of the method "List of known proxies"

Method name	List of known proxies
Data source	IP address
Base assumption	Information can be harvested at proxy providers
Preparation	Harvesting IP addresses and configurations from proxy providers' services
Operation	Incoming IP is looked up in a list
Non-statistical/ Statistical	Preparation: Non-statistical Detection: Non-statistical, blacklist/ lookup method
Active/ Passive	Preparation: Passive Detection: Passive
Hit	IP is proxy ($\varphi = \varphi_{proxy}$) false positive rate: nearly zero.
Miss	No information (not a known proxy)
Omission	Only known proxy providers No private/ proprietary service
Critical success factors	The information can be harvested from proxy providers The data harvested are complete and accurate

4.3 INFORMATION DATABASES

An IP address is not a self-contained entity: it exists in the context of the internet with all its protocols and rules. The contextual information about an IP address can reveal its use or its owner that in turn can give information on if it is a proxy or not. This information is stored in various databases. Some databases are publicly accessible while others require a subscription.

The following databases have been identified as useful sources:

- WHOIS;
- Reverse DNS;
- Specific databases;
    - Proxy database;
    - Usage type (ISP or Data Centre);
    - Port scan search platforms (see Paragraph 4.4.5).

4.3.1 Residential user vs. datacentres

A common assumption is that proxy servers are more commonly located in datacentres than hosted on home users' systems. This is based on the assumption that (commercial) proxy service providers use (virtual private) servers that need to be located in a reliable environment. Hosting providers provide these environments, and it is therefore more likely that proxies can be found in datacentres. Information databases can assist in determining if an IP address is located in a datacentre or if it is designated for residential users.

A detection method based on this assumption produces results that are not conclusive: an IP address of a residential user can still be used as a proxy, and a hosted server can still connect directly. In addition, there are proxy services (mostly application-layer proxies) that advertise they use millions of residential IPs or IP addresses that belong to mobile providers, for example: geosurf.com, luminati.io, and proxyrotator.com. Therefore, tests based on the above assumption provide a shift in odds: when an IP is located in a datacentre the odds increase in favour of it being a proxy. Likewise, when the IP is designated to residential users, the odds increase in favour of it being a direct connection.

4.3.2 WHOIS and reverse DNS

WHOIS and DNS are protocols for retrieving information from databases that are part of the internet. The WHOIS is a method for checking information about ownership of a domain name, an IP address or an autonomous system ("WHOIS" 2018). The WHOIS method queries regional internet registers like ARIN (American Registry for internet Numbers), RIPE (Réseaux IP Européens) and APNIC (Asia-Pacific Network Information Centre). Checking an IP address with WHOIS gives information on who owns or controls the range of IP addresses the queried IP address is part of. In some cases, the record shows a VPN provider owns the IP range, which is a strong indication that the IP address is used as a proxy. In other cases, the IP address is owned by an ISP providing services for home users. This is an indication the IP address is likely not a proxy, under the assumption proxies are commonly not hosted by home users.

In computer networks, a normal (forward) DNS lookup is to retrieve the IP address that is associated with a domain name. The reverse DNS lookup (rDNS) is the retrieval of a domain name of an IP address. While a forward lookup is stored in a DNS A-record, a reverse lookup is stored in the DNS PTR-record ("Reverse DNS Lookup" 2018). The purpose of a PTR record is mostly administrative: it shows an IP is used with a particular domain, or the function of the server. It is also used as an anti-spam technique. The owner of the IP range has the authority to set and change the PTR records. Some hosting providers allow their users to change the PTR-record of their IP address. In some cases, the

reverse DNS record of an incoming IP connection includes proxy-related information. This can for example be the name of the proxy service or indication of the type of proxy (e.g. a tor-node). On the other hand, it is also possible that an rDNS record holds information on whether or not the IP is owned or used by domestic or users. For example, keywords such as "adsl" or "ppp" indicate the IP addresses are used by home user.

Unfortunately, the information from the WHOIS and rDNS queries does not unambiguously identify the usage of an IP address. This is because the owner of the IP address can fill these records as he or she sees fit. To use the information, keywords need to be determined to classify a proxy by its WHOIS and rDNS records. This makes the detection using WHOIS and rDNS queries a two-step method. Some keywords are strong indicators the IP is a proxy, like the name of a proxy provider, while others are weaker indicators.

4.3.3 Proxy databases

Identifying proxies has been of interest to some people for some time, and there are several commercial lookup services for proxies. These services can be used to check if an incoming connection is a proxy. The trust in the accuracy of the data lies with the third party providing the service. It is not known how the data are gathered, handled, maintained and assessed. Nevertheless, a hit in a proxy database significantly increases the odds in favour that the IP is a proxy.

4.3.4 Properties and success factors of methods

The following methods are based on the above described technique:

- Proxy databases
- Provider- and proxy-related keywords in WHOIS and rDNS records
- Datacentre- vs. residential-related keywords in WHOIS and rDNS records

Table 2 Properties of the method "Proxy databases"

Method name	Proxy databases
Data source	IP address
Base assumption	Third-party databases hold proxy IPs
Preparation	None
Operation	Incoming IP is queried in database
Non-statistical/ Statistical	Preparation: Non-statistical Detection: Non-statistical, blacklist/ lookup method
Active/ Passive	Preparation: Passive Detection: Passive
Hit	Shift in odds in favour $\varphi = \varphi_{proxy}$ False positive rate: unknown (see omissions)
Miss	No information (not a known proxy)
Omission	3rd party database: - Unknown accuracy - Unknown source of data - Unknown method

Critical success factors	The proxy database is: - Accurate - Complete - Up-to-date

Table 3 Properties of the method "Provider- and proxy-related keywords in WHOIS and rDNS records"

Method name	Provider- and proxy-related keywords in WHOIS and rDNS records
Data source	IP address
Base assumption	WHOIS and rDNS contain info on IP usage as proxy
Preparation	Determine keywords
Operation	1. Incoming IP is queried in database 2. Record is searched for keywords
Non-statistical/ Statistical	Preparation: Statistical; keywords need to be tested on their predictive value Detection: Non-statistical, blacklist/ lookup method
Active/ Passive	Preparation: Passive Detection: Passive
Hit	Shift in odds in favour $\varphi = \varphi_{proxy}$ over $\varphi = \varphi_{direct}$ False positive rate: not likely to be high
Miss	No information
Omission	IPs without record Unidentified keywords
Critical success factors	The records for a searched IP are available Keywords can easily be determined The keywords are a good indication for the use of an IP (low false positives)

Table 4 Properties of the method "Datacentre- vs. residential-related keywords in WHOIS and rDNS records"

Method name	Datacentre- vs. residential-related keywords in WHOIS and rDNS records
Data source	IP address
Base assumption	Proxies are more often located in datacentres Proxies are less often hosted on residential IPs WHOIS and rDNS contain info on usage
Preparation	Determine keywords
Operation	1. Incoming IP is queried in database 2. Record is searched for keywords
Non-statistical/ Statistical	Preparation: Statistical; keywords need to be tested on their predictive value Detection: Non-statistical, blacklist/ lookup method
Active/ Passive	Preparation: Passive Detection: Passive
Hit	Shift in odds in favour $\varphi = \varphi_{proxy}$ over $\varphi = \varphi_{direct}$
Miss	Shift in odds against $\varphi = \varphi_{proxy}$ over $\varphi = \varphi_{direct}$
Omission	IPs without record Unidentified keywords

Critical success factors	The base assumption is correct (proxies are more likely hosted in datacentres than at designated residential IP addresses) The records for a searched IP are available Keywords can easily be determined The keywords are a good indication for the use of an IP (low false positives)

4.4 PORT AND SERVICE SCANNING

When a client uses a proxy, it does so by connecting to the IP address of a proxy server. When the proxy service consists of a single node and the IP address is reused as a pseudo source address, the server receiving a connection is able to scan the originating IP address for indications of the proxy service. Scanning for open ports and services running on the source of the incoming connection can therefore indicate whether it is a proxy.

The TCP and UDP port numbers used by the proxy service can be configured as the provider sees fit. This means that ports other than the default can be chosen. It is also observed that port numbers are designated on demand: when a client wants to connect to the proxy, a (random) port number is allocated which the client can use. In the observed case, once a client has connected, no new proxy connections are permitted at the open port and the port no longer responds to probes.

Subject to the used proxy server software and its implementation, scanning the service returns useful information for detecting the presence of the proxy. In some cases this information identifies the IP as a proxy, while in other cases the open port only indicates its use. The probing technique is specific to the type of proxy used. Three proxy methods are examined more closely.

4.4.1 OpenVPN

A prepared packet can be sent to an OpenVPN server to trigger a response. Because of the protocol, the response provides a high level of certainty that the IP is a proxy. However, if the server is configured to accept only TLS authenticated connections on an UDP port, the server will not respond unless the probe has a correct HMAC. A response can be elicited if the correct authentication key and hash algorithm is known. Some (commercial) OpenVPN providers use a shared authentication key. Once this key is known, all proxies of that provider can be scanned and identified.

4.4.2 IPsec

Like with OpenVPN, a packet sent to an IPsec proxy service that is listening on a port can trigger a response. With both L2TP over IPsec and IKEv2, the port 500/UDP is used for key exchange and always returns a response when a proper packet is sent. The response can either be a returned handshake or a notification. In some cases, the vendor and device type can also be revealed.

4.4.3 PPTP

The default port for PPTP is 1723/TCP. A targeted probe reveals the firmware version, vendor, and the hostname when available. This can for example be done using for nmap: `nmap -Pn -sV -p 1723 TARGET(S)`.

4.4.4 Other scan techniques

Some proxy services have other identifiable responses to probes. These can be unrelated to specific proxy techniques, but nevertheless reveal the server is used as a proxy. This can be a wide range of scan types, for example:

- Home routers running the MircoTik RouterOS can be configured as a proxy. In default mode the proxy shows an error page for unauthorised access. In that webpage the text "Mikrotik HttpProxy" is present, revealing it is used as a proxy.
- Kerio Control Proxies (http://www.kerio.com/support/kerio-control) also serve a webpage with the text "Forbidden" when probed on port 80. It also includes the text "This message was created by Kerio Control Proxy", revealing it is used as a proxy.
- Proxy providers sometimes also host a webpage on their proxy server advertising their proxy service. This also reveals it is a proxy when probed.
- Some proxy servers also host an HTTPS webpage using a socket-layer encryption, where the name of the company is in the certificate.

4.4.5 Port and service scan databases

Instead of actively scanning the IP addresses, it is also possible to use databases with scan results. Services like Shodan (shodan.io), Censys (censys.io) and URLScan (URLScan.io) periodically scan the internet for open ports and services and publish their results. A query of an IP address provides information that can be used for classifying the IP.

A disadvantage is that not every IP address is scanned, and the frequency of scans is not always clear. Therefore, the scan result might be old or outdated. In addition, not all ports and services are scanned, especially the non-standard ports.

4.4.6 Properties and success factors of methods

The following methods are based on the above described technique:

- Proxy port and service scanning
- General port and service scanning
- Port and service scan database

Table 5 Properties of the method "Proxy port and service scanning"

Method name	Proxy port and service scanning
Data source	Responses to TCP/UDP probes
Base assumption	Proxy software responds to probes identifying themselves
Preparation	Select tools for each proxy technique Specific for OpenVPN: Harvesting of shared authentication key of proxy providers
Operation	1. Incoming IP is probed 2. Result is assessed
Non-statistical/ Statistical	Preparation: Non-statistical Detection: Non-Statistical

Active/ Passive	Preparation: - Passive: harvesting configurations - Active testing in testlab Detection: Active
Hit	IP is a proxy ($\varphi = \varphi_{proxy}$)
Miss	Shift in odds against $\varphi = \varphi_{proxy}$ over $\varphi = \varphi_{direct}$
Omission	Pseudo client IP is not proxy tunnel endpoint: - Proxy server with multiple IPs - Proxy with multiple nodes Specific to proxy technique: techniques not included in the probe are not detected OpenVPN specific: service with enabled authentication and unknown key
Critical success factors	The technique to scan a proxy is available The proxy is susceptible to probing Probing proxies is efficient: - The port number of the listening proxy software is known - The (OpenVPN) authentication is turned off or the key is known

Table 6 Properties of the method "General port and service scanning"

Method name	General port and service scanning
Data source	Responses to TCP/UDP probes
Base assumption	Scanned open ports and services give info on the use of the IP addresses
Preparation	Exploration of identifiable characteristics that can be scanned or probed. Strongly depends on provider/ software/ hardware.
Operation	1. Incoming IP is probed 2. Result is assessed
Non-statistical/ Statistical	Preparation: Non-statistical Detection: Non-statistical, blacklist/ signature based
Active/ Passive	Preparation: Active Detection: Active
Hit	IP is a proxy ($\varphi = \varphi_{proxy}$)
Miss	No information (no known port or service)
Omission	Pseudo client IP is not proxy tunnel endpoint: - Proxy server with multiple IPs - Proxy with multiple nodes Hardened servers are not identified Unknown identifiers are not detected Probing services that are forwarded through the tunnel are mistakenly seen as a direct connection
Critical success factors	

Table 7 Properties of the method "Port and service scan database"

Method name	Port and service scan database
Data source	IP address
Base assumption	Scanned open ports and services give info on the use of the IP addresses Third-party database hold appropriate port and service info
Preparation	None
Operation	1. Incoming IP is looked up in database 2. Result is assessed (manually)
Non-statistical/ Statistical	Preparation: - Detection: Non-statistical
Active/ Passive	Preparation: Passive Detection: Passive
Hit	Shift in odds either in favour or against $\varphi = \varphi_{proxy}$ over $\varphi = \varphi_{direct}$
Miss	No information
Omission	Dependant on completeness of 3th party database
Critical success factors	The IP address is in the database The appropriate ports have been probed The database entry is accurate and recent

4.5 LATENCY AND TIMING

If it is assumed that the routing of data on the internet always takes the fastest path, the use of a proxy causes the data to take longer than in a direct communication. This additional latency is caused by the extra transportation of data over the internet, as well as the extra time the proxy takes to process the data, for example for encryption and decryption.[§§]

In order to detect if a proxy has been used from the perspective of the server, time measurements have to be made between the server and the proxy, and between the server and the client. If these times differs significantly you can conclude a proxy has been used.

Four different methods are proposed to determine the additional latency caused by a proxy:

- measuring the round-trip time of the internet and application data stream;
- retrieving the timestamp fields in the TCP data streams, if present;
- probing the IP address by conducting out-of-band time measurements;
- triangulating measurements from other locations.

4.5.1 Round-trip time measurements

In telecommunications, the round-trip time (RTT) is the time it takes for a signal to be sent plus the time it takes for an acknowledgement of that signal to be received. This time delay includes the propagation times for the paths between the two communication endpoints (“Round-Trip Delay Time” 2018) (see Figure 6).

[§§] The setup of a tunnel also takes additional time. However, this only takes place once and is not part of the communication between client and server; therefore it is ignored here.

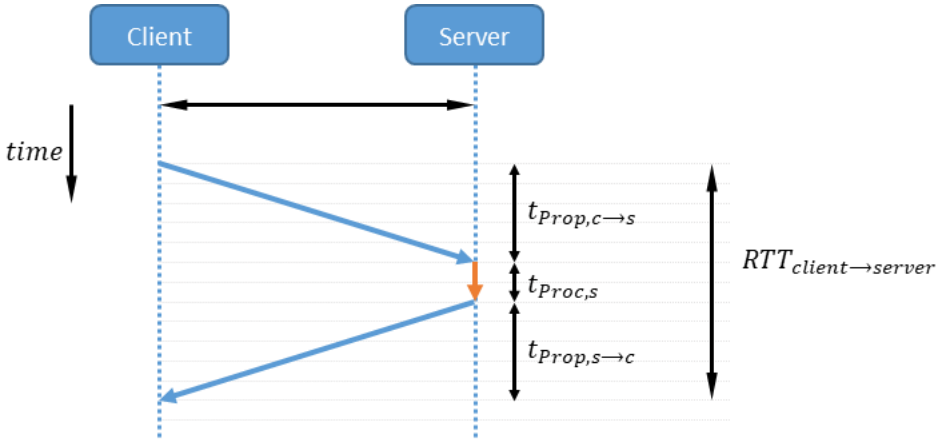

Figure 6 Standard round-trip time (RTT)

The standard simplified round-trip time consists of the propagation delay caused by the transportation of data from the client to the server ($t_{Prop,\,c\rightarrow s}$), the processing delay caused by the server to handle the incoming data and to create a response ($t_{Proc,\,s}$), and the propagation delay caused by the transportation back to the client ($t_{Prop,\,c\rightarrow s}$). This is a simplified model, as it does not specify smaller components of each delay, for example transmission delay, queuing delay, or processing delay on intermediate nodes like routers and switches.

When a proxy is used there is an extra time delay caused by the time it takes to send the data from the client to the proxy and vice versa. The difference between the RTT of the proxy and the server and the RTT of the client and the server could, theoretically, indicate the use of a proxy.

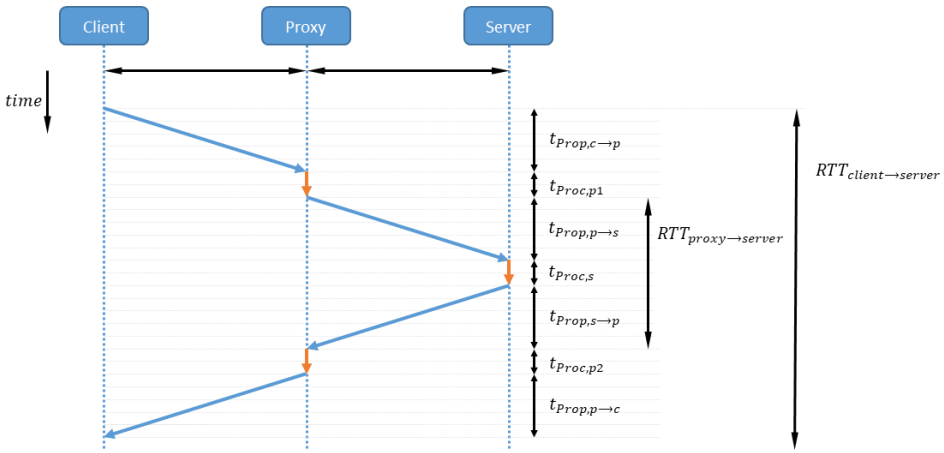

Figure 7 The Round-trip time of a proxy setup, measured by the client.

In Figure 7 data are sent by the client, relayed by the proxy and received by destination server. The server processes the packets and sends a response back to the client via the proxy. In the figure a distinction is made between the processing time by the proxy processing client packets ($t_{Proc,\,p1}$) and processing server packets ($t_{Proc,\,p2}$). This model is valid for all types of proxies, regardless of the layer the connection is proxied on.

When comparing a direct connection with a proxy connection with identical conditions (*ceteris paribus*), the proxy introduces the extra time: $t_{Prop,\,c\rightarrow p} + t_{Proc,\,p1} + t_{Proc,\,p2} + t_{Prop,\,p\rightarrow c}$.

Server perspective

The data flow in Figure 7 shows how the *client* can measure the total round-trip time $RTT_{client-server}$. However, this study focuses on traffic seen between the proxy and the server. It is therefore necessary to consider the flow of packets as presented in Figure 8.

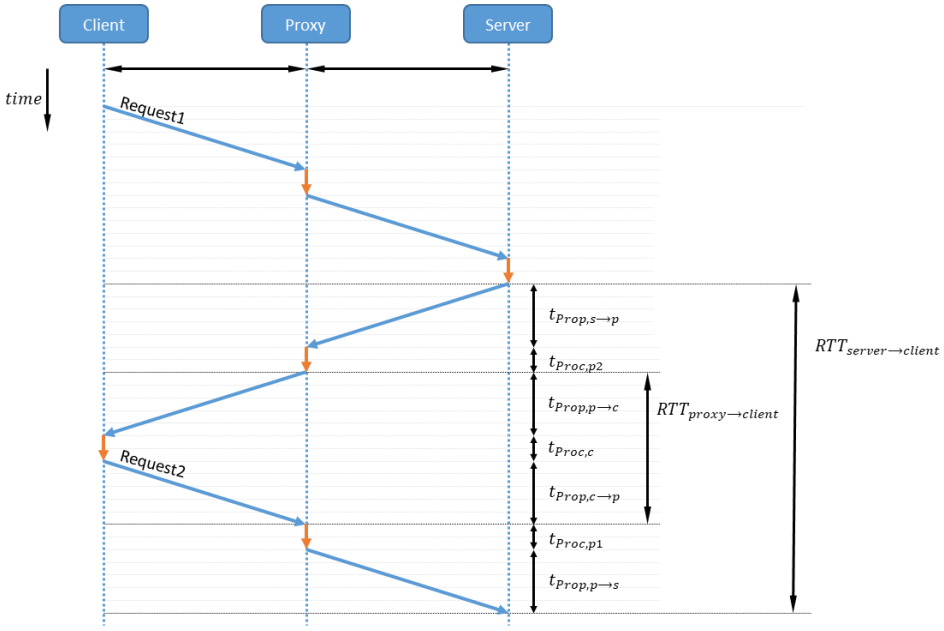

Figure 8 The Round-trip times of a proxy setup, measured by the server

At the server, the round-trip time can be measured by comparing the timestamp of the data sent and the received response. The measured RTT then includes the processing time of the client and not of the server.

There is however a small caveat here. In a client-server setup, the client initiates the communication. This means that at the server the RTT can only be measured when it is known that the *client* responds automatically to data sent by the *server* without user intervention.

Observer perspective

Another thing to consider is that the packet capture might not take place at the server, but somewhere between the proxy and the server. In that case, the time measurement by an observer is not the round-trip time as measured by the client or as measured by the server, as explained in Figure 9, but the 'full' round-trip time.

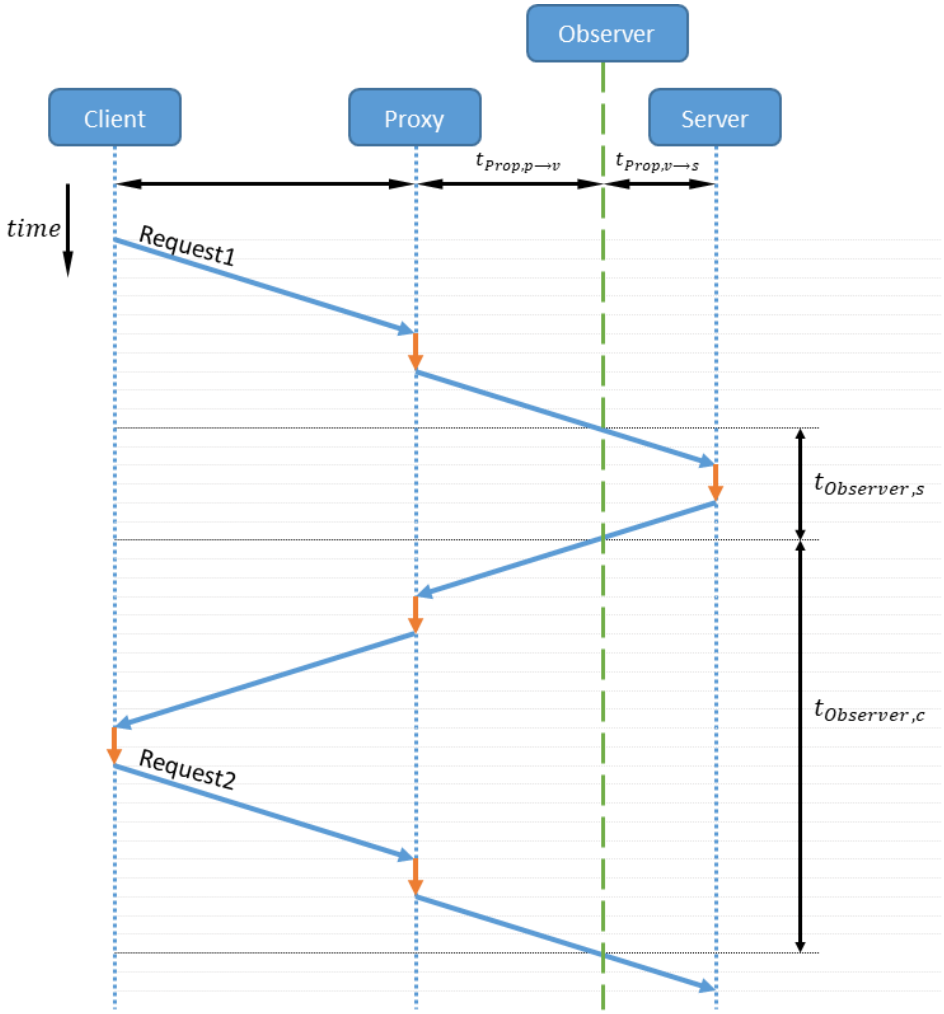

Figure 9 The 'full' round-trip time of a proxy setup, measured by an observer.

An observer somewhere between the proxy and the server measures a part of the propagation delay between the proxy and the server. The propagation delay between an observer and the server is expressed as $t_{Prop,\,v\rightarrow s}$ (with v for *viewer*).

In contrast to the previous round-trip times, a measurement by an observer includes the processing delay of *both the client and the server*. Together with the propagation delay between the client, the proxy, and the server, the observer measures the *full* round-trip time of the communication in contrast to the (normal) round-trip time of the client and the server.

The difference between the full-round-trip time of an proxy and a direct connection remains the time added by the proxy ($t_{Prop,\,c\rightarrow p} + t_{Proc,\,p1} + t_{Proc,\,p2} + t_{Prop,\,p\rightarrow c}$). It is safe to assume that the processing time of the client and server is the same for both situations.

4.5.2 Transport-layer round-trip time

There are three methods to determine the full round-trip time looking at the transport layer of the internet (i.e. the Transmission Control Protocol, TCP):

1. measuring the round-trip time of the three-way handshake;
2. timestamps included in the TCP header;

3. measuring the SYN ACK sequences.[***]

Measuring the three-way handshake

All TCP connections are established using a three-way handshake as described in RFC793. In this handshake the client and server establish a mutual agreement on the new connection. Because the handshake is a sequence of three packets between client and server, it is ideal for measuring the RTT at any location in the communication path. In addition the packets are small, giving them a good chance to be sent at maximum speed (larger packets might be buffered in transit somewhere). As a bonus, each TCP session starts with these packets, so they are easy to find and always present (packet-foo.com 2014).

Figure 10 shows an example of a three-way handshake. An observer sees three packets at times $t_{R.03}$, $t_{R.09}$, and $t_{R.17}$. The full-RTT is the difference of the local observer timestamp between the first and the third packet (in this case 14 time units).

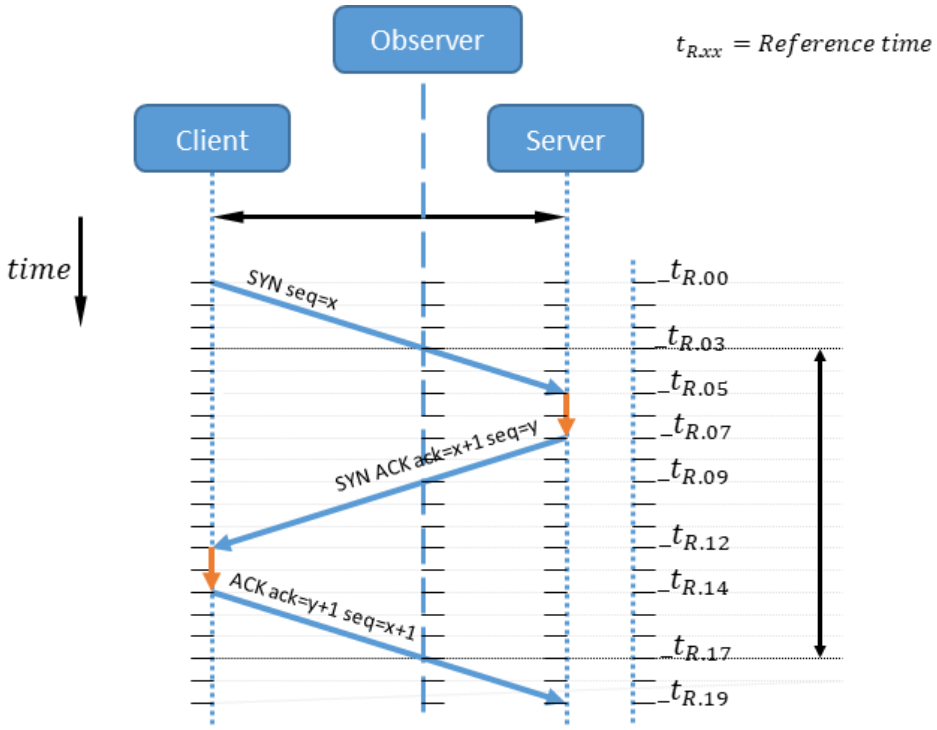

Figure 10 Full round-trip time of the TCP three-way handshake

Timestamps included in the TCP header

In 1992 new extensions to the TCP protocol were proposed for high performance, which included the TCP timestamp option (RFC1323, later obsoleted by RFC7323). When the timestamp option is negotiated in the three-way handshake, virtually every packet that is acknowledged can be measured using the RTTM mechanism as described in RFC7323. With this option set, the sender of a packet adds a timestamp (TSval) to the TCP header. On receiving the packet, a response packet (the ACK) is created where this timestamp is copied into the echo reply field (TSecr) and a new local timestamp is placed in the TSval field.

[***] Not explained further in this document.

This timestamp feature is very useful for determining the RTT of packets other than the handshake packets. Not only can this be done by the client and the server, as is its purpose, but also by an observer. Normally any sequence of packets seen by an observer, other than the three-way handshake, is not necessarily in a specific order. This is because a client can send multiple TCP data segments without having to wait for an acknowledgement from the server. Effectively, this means the client does not react to an acknowledgement packet sent by the server, therefore an observer cannot determine the RTT of these packets. However, when timestamps are added to the TCP header an observer is able to determine this.

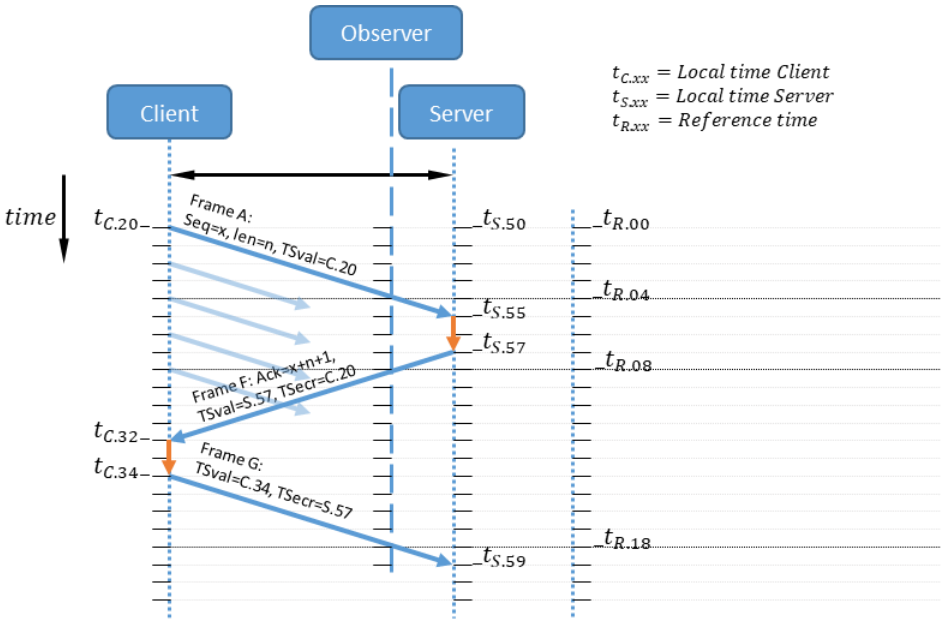

Figure 11 Measuring full-RTT using timestamps included in the TCP header

When timestamping is negotiated, both the client and the server echo back the last received timestamp of its counterpart. Figure 11 shows an example where a client sends a burst of data frames with a local timestamp added.[†††] At time $t_{S.55}$ the server returns an ACK packet (frame F), acknowledging frame A where the timestamp $t_{C.20}$ is echoed back, and a local server timestamp $t_{S.57}$ is added. After receiving this ACK-frame F by the client at $t_{C.32}$, any new frame sent by the client will use this last known timestamp of the server ($t_{S.57}$) until a new timestamp packet from the server is received.

An observer situated between the client and the server can measure the full round-trip time by comparing the timestamp values in the TCP header. In the previous example the observer sees seven packets passing through. With the following procedure the full round-trip time can be determined:

[†††] In the figure a distinction is made between the local time at the client ($t_{C.xx}$), the local time at the server ($t_{S.xx}$), and a reference time of the observer ($t_{R.xx}$). This distinction is made to make explicit that these times can be different and out of sync.

1. Examine consecutive packets coming from client.
2. When the TSecr value is different from the previous client packet, mark the timestamp and note the TSecr in this packet. In this example this is frame G that is observed at $t_{R.18}$ containing TSecr=$t_{S.57}$.
3. Look in the recent history for the first occurrence of a packet *from the server* to the client where the TSval equals the TSecr of the previous step. Note the TSecr of this found packet, as well as the ack sequence number. In this example look for the first packet from the server with TSval = $t_{S.57}$. This is frame F observed at $t_{R.08}$ containing TSecr = $t_{C.20}$ and ack = x+n+1.
4. Look further back in the recent history for the packet *from the client* that has this timestamp value from step 3 for its TSval. An extra check can be made by checking the sequence number. This should match the ACK of step 3 plus the length of this packet plus one. Mark the timestamp of this captured packet. In this example this is frame A observed at $t_{R.04}$.
5. Calculate the full round-trip time by comparing the captured timestamps of step 2 and 4. In this example the full round-trip time is: $t_{R.18} - t_{R.04} = 14$.

A similar procedure can be used by regarding a change in the TSecr from consecutive packets from the *server* instead of the client.

4.5.3 Application-layer round-trip time

When an application-layer proxy is used, the client sets up an internet connection with the proxy; in turn, the proxy sets up an internet connection with the server. There is no internet- or transport-layer connection between client and server. As a consequence, the round-trip time between server and client cannot be measured by looking at the transport layer. Another consequence is that application layer proxies are falsely classified as direct connections when the transport-layer round-trip time is measured. To distinguish application-layer proxies with time measurements, the focus must be on the communication of the application.

Similar to the transport layer, the client-server application communication usually has some automatic responses. A client sends a request and the server responds to it. For example, a web browser application sends a HTTP GET request that the server responds to with web content. For the observer to determine the full-round-trip time, the *client* needs to automatically respond to a communication from the *server*, for example, with nested with HTML objects. The client application retrieves an HTML page and then sends additional requests for pictures, iframes, and other objects. The client automatically reacts to a message from the server, as required to measure the full-round-trip time. By measuring the time of these client responses, the full-round-trip times can be calculated.

When making these types of measurements, the application protocol should be referenced carefully and various practical caveats should be considered. For example, local caching of HTML objects and parallel requests for nested HTML objects make time measurements difficult. In addition, different versions and implementations of applications can behave differently. It may be necessary to create a profile of the client and server applications first in order to be able to take the right time measurements. Encrypted traffic makes profiling more difficult, although this might be of no consequence for time measurements.

4.5.4 Probing IP addresses

It is common practice to use the internet Control Message Protocol (ICMP) to troubleshoot network problems. Most commonly the echo request and reply mechanism is used, also known as ping. Most computers respond to these low-level probes as described in RFC 1222, although ping responses are sometimes turned off for security reasons. Although ping is believed to be inaccurate in determining network latency, it still provides some indications. Most importantly, ping packets are answered by the proxy and not passed on through the tunnel to the client when internet and application-layer proxies are used.

Another way to measure the round-trip time between the server and the proxy is by probing open services on the proxy. If the proxy server is running other TCP services (e.g. a web server or SSH) a connection can be made invoking a three-way handshake. By measuring this, the RTT between server and proxy can be determined.

4.5.5 Triangulation of third-party measurements

If the investigator is not in a position to actively perform time measurements from the server, it is also suggested to determine the latency from servers located close to the client and server. Although this seems like an impossible task to perform in practice, there are services that provide latency testing from various locations on the internet. For example, with the service offered by MapLatency various latency tests can be performed from over 5,000 locations worldwide ("Global Internet Testing | Speedchecker Ltd." n.d.). By triangulating time measurements from various locations, the network and application latency between the client and server can be estimated. This method seems potentially interesting. Further studies are needed on this method.

4.5.6 Comparing times

This detection method is based on the comparison of two different time measurements. Depending on the type of proxy, different times should be compared. For internet-layer proxies the full-round-trip time can be measured with the TCP three-way handshake, because the TCP connections between the client and the server pass transparently through the proxy. To measure the round-trip time between the server and the proxy, the ICMP-ping times can be used. If the proxy server hosts other services like a webserver or SSH, these can also be used to measure the reference time; the server connects to the services on the proxy and measures the (full) round-trip time of the TCP three-way handshake.

To detect application-layer proxies the round-trip time of the application layer and the transport layer can be measured and compared. The first extends between the client and the server, the second between the server and the proxy. The difference in time could potentially indicate the use of a proxy.

Two-threshold method

When comparing two time measurements, a decision criterion must determine whether the IP is a proxy or not. When the time difference is relatively high it is concluded it is a proxy, and vise versa. It is proposed here to use a method with two thresholds:

$$\Delta_{time} > threshold_{proxy} \quad \rightarrow IP\ is\ likely\ a\ proxy$$

$$\Delta_{time} < threshold_{direct} \quad \rightarrow IP\ is\ a\ direct\ connection$$

$$threshold_{direct} < \Delta_{time} < threshold_{proxy} \quad \rightarrow Undetermined$$

where: $threshold_{proxy} > threshold_{direct}$. This creates a third category where it is uncertain whether IP is a proxy or not, as the time difference lies between the two thresholds. The exact thresholds depend on the used method of measurement and must be determined empirically. By making time measurement with known proxies and direct connections, the threshold can be determined. It is expected that some measurements cannot be clearly categorised and fall in the 'undetermined' category. When more experiments have been conducted and analysed, alternative models can be created.

It is assumed that time measurements of internet packets can be modelled as a normal or Gaussian probability distribution, characterised by a mean (μ) and a standard deviation (σ). Taking into account the spread of the measurements, a threshold should not be based on the mean alone. Take for example the two measurements in Figure 12. The difference between the mean of both measurements ($\Delta_{time} = \mu_{measurement\ 1} - \mu_{measurement\ 2}$) could be above the $threshold_{proxy}$ making it fall into the category 'is a proxy'. However, the individual times of measurement 2 all fall within the range of measurement 1 and therefore it could easily be concluded it is a direct (non-proxy) connection.

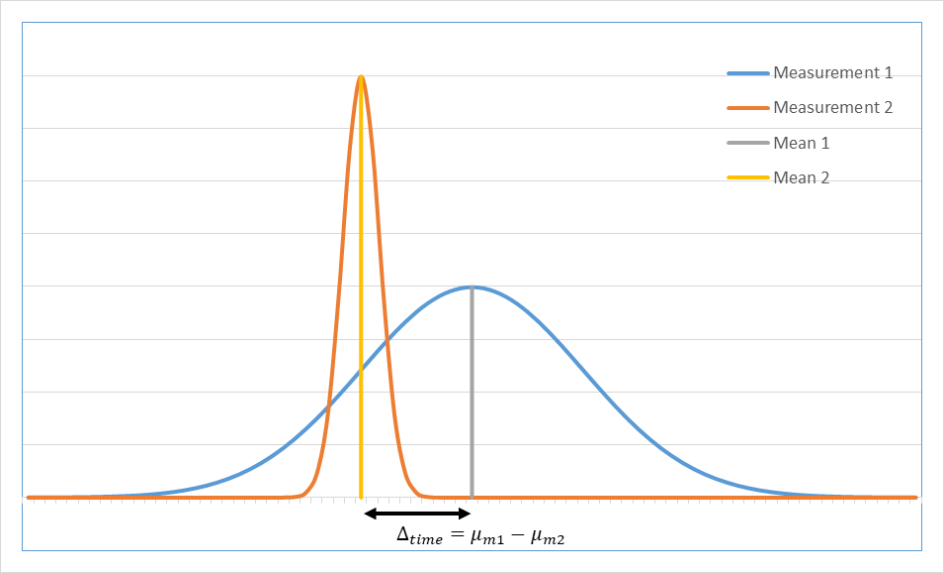

Figure 12 Two time measurements with different spreads compared by mean

A better method would be one where there is less overlap in the measurements and that considers the spread of the individual measurements. This can be done by regarding the percentage under the curve and taking an appropriate standard z-score (Z); for example, taking the value where 90% of the individual measurements would mean a z-value of 1.34 (according to the z-score table

("Standard Normal Table" 2018)). The values to be compared would then be: $\Delta_{time} = (\mu_{m1} - 1.34 \times \sigma_{m1}) - (\mu_{m2} + 1.34 \times \sigma_{m2})$ as shown in Figure 13.

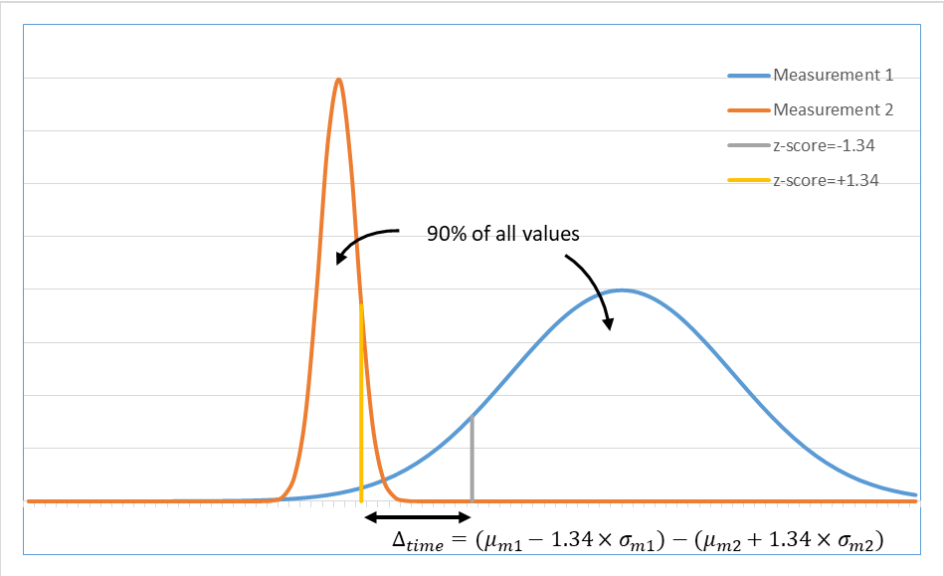

Figure 13 Two time measurements with different spreads compared using a z-score

This method would ensure that no more than 10% of all measurements would overlap, or whatever percentage is chosen. Similarly, this method is used for the detection of a non-proxy. In that case this method would ensure that at least 90% of all measurements would overlap when the negative z-score is subtracted in both measurements.

Give the previous, the classification is done by:

$$(\mu_{m1} - Z \times \sigma_{m1}) - (\mu_{m2} + Z \times \sigma_{m2}) > threshold_{proxy} \rightarrow IP\ is\ likely\ a\ proxy$$
$$(\mu_{m1} - Z \times \sigma_{m1}) - (\mu_{m2} - Z \times \sigma_{m2}) < threshold_{direct} \rightarrow IP\ is\ a\ direct\ connection$$

where $m1$ is the measurement between the server and the client, which is expected to take the longest, and $m2$ the measurement between the server and the proxy. The values Z, $threshold_{proxy}$, and $threshold_{direct}$ need to be determined experimentally.

4.5.7 Related research

A search for related (scientific) research of determining proxies by looking at timing and latency produced few results. Most of the studies found concerned the detection of proxy tunnels from the client side, for example to detect a client in a company network establishing a proxy tunnel outside of the network. However, research by (Webb and Reddy 2016) was found relevant to this study. In their research the RTT for each connecting IP address is measured, and anomalies in the distribution of these RTTs are flagged as proxies. There is no clear explanation of the difference in RTT distribution, however Nagle's algorithm and the TCP large segment offload mechanism are suggested to be the cause. The RTTs are measured at the transport layer (the TCP Delay Distribution) and at the application layer (the HTTP Response Time). The research shows there is a measurable distinction between normal and proxy traffic based on the distribution of RTTs, particularly for application-layer proxies.

Unfortunately, the study does not provide answers as to the cause of this distinction. In addition, the method is not deemed to be practical as it must be trained for each IP address. However, the closing remarks suggest that it might be worthwhile to investigate if IP addresses with similar paths can be trained as a group.

4.5.8 Properties and success factors

The following methods are based on the above described technique:

- Timing internet-layer proxies
- Timing application-layer proxies

Table 8 Properties of the method "Timing internet-layer proxies"

Method name	Timing internet-layer proxies
Data source	1. Internet-layer packet stream (TCP three-way handshake, timestamps in TCP header) 2. ICMP (ping) probes or probing of open ports
Base assumption	Time difference between TCP handshake and ICMP probes indicates proxy
Preparation	Determine the threshold or the likelihood
Operation	1. Capture packets and mark arrival times 2. Ping the originator 3. Compare times 4. Determine likelihood
Non-statistical/ Statistical	Preparation: Statistical. Binary decision based on trained thresholds, or likelihood based on trained dataset Detection: Statistical. Multiple measurements are classified based on trained criteria
Active/ Passive	Preparation: Active Detection: Passive and Active. Passive packet capture and active probing
Hit	Shift in odds either in favour or against $\varphi = \varphi_{proxy}$
Miss	Outside threshold bandwidth: No information (indecisive)
Omission	Thresholds must be accurately set Application proxies are falsely identified as direct connections
Critical success factors	Measuring the TCP-handshake time and ICMP-ping time distinguishes a proxy from a direct connection The used method for measuring times is a representative way to detect these times

Table 9 Properties of the method "Timing application-layer proxies"

Method name	Timing application-layer proxies
Data Source	1. Application stream packets 2. Internet-layer packet stream (TCP three-way handshake, timestamps in TCP header) or ICMP times (probes or triangulation)
Base assumption	Time difference between application-layer packets and internet-layer packets indicates a proxy
Preparation	Determine the threshold or the likelihood

Operation	1. Capture packets and mark arrival times 2. Probe the originator 3. Compare times 4. Determine likelihood
Non-statistical/ Statistical	Preparation: Statistical. Binary decision based on trained thresholds, or likelihood based on trained dataset Detection: Statistical. Multiple measurements are classified based on trained criteria
Active/ Passive	Preparation: Active Detection: Passive and Active. Passive packet capture and active probing
Hit	Shift in odds either in favour or against $\varphi = \varphi_{proxy}$
Miss	Outside threshold bandwidth: No information (indecisive)
Omission	Thresholds must be accurately set There are automatic responses of the client to a server response Depends on the client and server application
Critical success factors	A correlation exists between the application round-trip time and internet-layer round-trip time The used method for measuring times is a representative way to detect these times

4.6 MTU SIZE

When a client uses an internet proxy, packets are sent from the client to the proxy to be forwarded to the server. Depending on the protocol used to send these packets to the proxy through the tunnel, the data are encapsulated in other packets; extra headers are added, and in some cases the data are compressed and encrypted and MACs are added.

4.6.1 Maximum Transmission Unit

To make optimal use of the bandwidth of a network, the transported packets need to be of a certain size. If packets are too small, the number of overhead bytes used for IP and TCP headers make transportation inefficient (Murray et al. 2012). In data communication the Maximum Transmission Unit (MTU) is defined as the maximum data that can be transmitted in one packet. Every link-layer network protocol has an MTU value. For Ethernet the MTU is set to 1500 bytes.

When a packet from a client of 1500 bytes is encapsulated in a tunnel protocol and extra headers are added, the total packet size is larger than the maximum. When this packet is sent it will be split up into two packets that are smaller than 1500 bytes. This fragmentation will cause the speed to dramatically drop because of the overhead. To prevent this, tunnel protocols can advertise a smaller MTU to the application layer (Savola 2006). This causes the packets to be encapsulated to fit perfectly, and the total package never to exceed the MTU of 1500 bytes.

4.6.2 TCP Maximum Segment Size

When a TCP connection is established, both endpoints state their Maximum Segment Size (MSS) to inform the other of the maximum TCP data size it can receive without packet fragmentation. Because an internet proxy forward all IP and TCP packets transparently, the client sends its MSS value

as it is set by the proxy interface. The client sets the MSS to the MTU minus the size of the IP header (20 bytes) and TCP header (20 bytes). So, normally the MSS is set to 1500 – 40=1460 for TCP/IP over Ethernet.

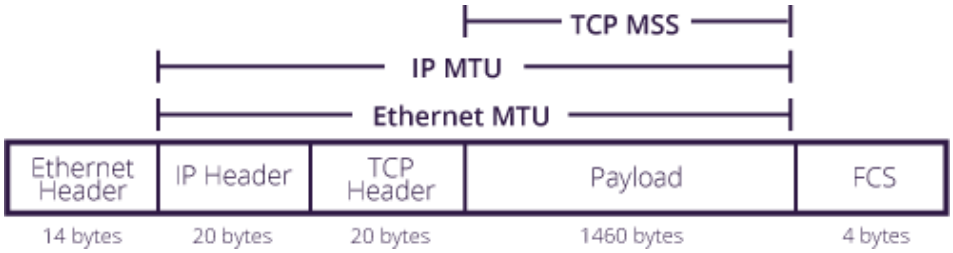

Figure 14 The MTU and MSS of an Ethernet packet

When an incoming connection advertises an MSS other than 1460 this does not necessarily mean the connection is a proxy. The MSS value can differ for many reasons. However, there is a relation between the proxy settings and the advertised MSS value. The internet user ValdikSS (real name unknown) observed this relation in OpenVPN (ValdikSS 2015).

4.6.3 Properties and success factors

The following method is based on the above described technique:

- MSS fingerprinting

Table 10 Properties of the method "List of known proxies"

Method name	MSS fingerprinting
Data source	MSS value in TCP three-way handshake
Base assumption	A correlation between the advertised MSS value in the TCP and the use of an internet-layer proxy
Preparation	
Operation	1. MSS setting of incoming IP is read 2. MSS value is looked up in a list of known MSS values
Non-statistical/ Statistical	Preparation: Non-statistical Detection: Non-statistical, blacklist/ lookup method
Active/ Passive	Preparation: Passive Detection: Passive
Hit	IP is a proxy
Miss	IP is not a proxy
Omission	Proxy configurations that prevent MSS fingerprinting
Critical success factors	There is a relation between MSS value and the use of a proxy The identified MSS values are unique for proxies

4.7 HTTP HEADER FIELDS

Some application-layer proxies used for web browsing add extra information to the HTTP header, as described in Paragraph 3.1. These extra lines can be detected once it is known what lines or keywords are added by proxies. A list of these keywords is included in Appendix I. This technique can be used by webservers to detect if the client is using a proxy.

4.7.1 Properties and success factors

The following method is based on the above described technique:

- HTTP header fields

Table 11 Properties of the method "HTTP header fields"

Method name	Known proxy HTTP header fields
Data source	HTTP header
Base assumption	A proxy adds information to the HTTP header
Preparation	Inventory of keywords included by proxies
Operation	1. Capture the incoming http header 2. Check for known keywords
Non-statistical/ Statistical	Preparation: Non-statistical Detection: Non-statistical, blacklist/ lookup method
Active/ Passive	Preparation: Passive Detection: Passive
Hit	Incoming connection is a proxy
Miss	No information
Omission	Encrypted traffic is not detected Proxies that do not add information to the header No internet-layer proxies are detected
Critical success factors	The proxy adds recognisable information to the http header

4.8 SUMMARY

An inventory of different techniques is made based on studying various literature, proof-of-concept initiatives, and blog posts on the internet. While creating this inventory it was found that no single technique stood out as the generally accepted one. The following techniques and derived methods were identified:

- Harvesting information at the source, used in the method:
  - Lists of known proxies: IP addresses of known proxy providers are harvested
- Information databases containing IP-related information, used in methods:
  - Provider- and proxy-related keywords in WHOIS and rDNS records
  - Datacentre- vs. residential-related keywords in WHOIS and rDNS records
  - Proxy databases
- Port and service scanning, used in the methods:
  - Proxy port and service scanning: probing the proxy service
  - General port and service scanning: probing the server running the proxy service
  - Port and service scan database: searching though databases containing IP addresses of proxies
- Latency and timing, used in the methods:

    - Timing internet-layer proxies: measuring TCP level latencies and comparing it with ICMP probes
    - Timing application-layer proxies: measuring application level latencies and comparing it with TCP level latencies or probing
- MTU size, used in the method:
    - MSS detection: reading the MSS value in the TCP three-way handshake
- HTTP header fields, used in the method:
    - Known proxy HTTP header fields
- Machine learning: described in general. No specific method is mentioned.

For each method the following general properties are included:

- The type of information on which the detection is performed;
- Base assumption;
- Necessary preparation;
- General operation;
- Whether it is based on statistical or non-statistical methods;
- If it uses active probing or not;
- Its reasoned accuracy; and
- The reasoned omissions.

Finally, for each method the identified critical success factors to implement it in practice are included. The results of tests of selected success factors are presented in the next chapter.

5 DETECTING PROXIES

The previous chapter described a number of detection methods that can potentially be used to detect proxies. For each method the limitations and capabilities from a theoretical perspective were provided, as well as the factors that are crucial for the method to work in practice. A next step in creating a proxy detection method is to gain practical knowledge about these essential conditions: *can proxies be detected in practice at all by using the method?* This chapter present the results of the experiments performed on the critical success factors. These results can be used for the further development of the detection methods.

The next paragraphs include a selection of the detection methods described in Chapter 4. For each of these detection methods, a research question is formulated that gives insight into a selected critical success factor. This is followed by a brief description of how the experiment was conducted, its results, and the conclusions. Note that the focus of the experiments was on the feasibility, general capabilities, and limitations of the detection method, and not on its validation. This is why only the true positive rate (TPR) was determined in most experiments using a set of known proxies.

5.1 LIST OF KNOWN PROXIES

With the method "List of known proxies", as described in Paragraph 4.2, it is assumed it can be known beforehand that an IP address is used as a proxy because the provider indicates it as such. The method involves previously compiled lists of IP addresses. An important success factor of this method that the information can be harvested from proxy providers:

Can a list of IP addresses be compiled for a specific proxy provider?

To answer this question, three providers are selected. For each the internet is searched for lists of IP addresses. This paragraph contains the results of the experiments creating these lists.

5.1.1 Experiment and results

Three random popular VPN proxy providers are selected:

- An open-source proxy provider: Tor;
- A commercial provider with a subscription: ExpressVPN;
- A commercial provider without a subscription: NordVPN.

Tor exit nodes

The exit nodes of the Tor network are publicly available. An internet search revealed various places where the list containing the current available exit nodes can be downloaded. On some websites it is also possible to check if a given IP has been used as a Tor exit node at some point in the past. [‡‡‡] For this study a lookup list is created by downloading the list from https://www.dan.me.uk/torlist/?exit. The list has approximately 935 IPv4 addresses.

[‡‡‡] For example the ExonerTor service of the Tor Project on: metrics.torproject.org/exonerator.html

ExpressVPN

On the website of ExpressVPN (www.expressvpn.com) the OpenVPN configuration files can be downloaded, as well as a list of servers for PPTP and L2TP/IPsec connections. These files and addresses can only be downloaded with a subscription. Parsing the configuration files and scraping the server list from the website results in a list of 689 unique IP addresses. However, this list seems incomplete; when the ExpressVPN client was used with the subscription credentials, other IP addresses were used as proxy servers that were not on the retrieved lists.

NordVPN

An internet search quickly revealed that NordVPN (www.nordvpn.com) conveniently provides an undocumented interface where information on all current available proxies with their status can be found (nordvpn.com/api/server). This API can easily be automatically queried with a script to regularly create an up-to-date lookup list. During the experiments a list of 4846 IP addresses was retrieved.

5.1.2 Other observations

All observed providers (including the 35 used in Paragraph 5.5) use domain names in the configuration of the client. This creates additional flexibility for the proxy providers since they can easily change an IP address without changing the configuration by simply changing the DNS A-record.

During the study it was noticed that other have similar interests. Searching specifically for provider names quickly points to posts on the internet where IP addresses can be found. Based on these posts and open-source projects involving proxy configuration, it is highly likely that a list of IP addresses of at least 36 commercial VPN providers could be created.

5.1.3 Conclusion

It is possible to create a list of IP addresses for a specific provider. In most cases the list is retrievable without a subscription and can be retrieved automatically using scripts. There are exceptions where a subscription is needed to gain access to this information.

This method is somewhat labour-intensive because it requires some scripting and research for each new provider. In addition, interfaces and websites of providers change and the (automatic) retrieval needs to be adjusted. For proxy providers like ExpressVPN the complete list is not so easy to obtain. In this case, the VPN client needs to be reverse engineered to know where and how the actual list of IP addresses can be found.

5.2 PROXY DATABASES

This detection method uses an external database with IP addresses labelled as proxy. This method assumes that an external (commercial) database contains an accurate list of known proxies that can be queried. An important success factor of this method is the accuracy of database. To partly determine this accuracy the following is investigated:

What is the true positive rate (TPR) of a proxy database?

To answer this question, a selected proxy database is queried with a selected list of known proxy IP addresses.

5.2.1 Experiment and results

Proxy database

A brief internet search revealed only two databases containing proxy IP addresses. For this experiment the proxy database IP2Location (www.ip2location.com) is used because (limited) queries can be done free of charge (IP2Location n.d.).

Population of known proxy IPs

To test the accuracy of the selected proxy database, a list of known proxy IP addresses is used. For this the IP addresses of Tor, ExpressVPN and NordVPN are used that were retrieved using the method described in Paragraph 5.1.

Results

Table 12 shows test results of the IP addresses found in the proxy database for each provider and the total for all providers.

Table 12 Test results of proxy IPs found in a proxy database

Proxy database lookup	Population of known proxies	# of IPs (n)	TPR
Tor-specific	Tor IPv4 nodes	916	0.96
ExpressVPN-specific	ExpressVPN	689	0.54
NordVPN-specific	NordVPN	5157	0.46
Total	**Tor & ExpressVPN & NordVPN**	**6762**	**0.65**

5.2.2 Conclusion

The result using a proxy database is diverse. The detection of Tor proxies is relatively accurate. This is logical because the IP addresses are publicly known, and that is what is expected of a commercial proxy database. On the other hand, the correct detection of commercial proxies is only 47%. Because there are many more commercial providers than public providers, the overall average is expected to be lower when more proxy providers are added to the test.

5.3 Provider- and proxy-related Keywords in WHOIS and rDNS

In this method the WHOIS databases and the reverse DNS with IP-related information are queried for an IP address. It is assumed that these databases hold information that can reveal if the IP address is used as a proxy or not. An important success factor is the accuracy of the method. This is partly measured by answering the following question:

What is the true positive rate (TPR) of lookups with proxy-related keywords in records from IP-related databases?

To answer this question, a selection of IP addresses known to be proxies are queried in the WHOIS and rDNS databases. For this the known IP addresses of Tor, ExpressVPN and NordVPN are used (see par. 5.1.1). The resulting records are then examined for specific keywords related to the provider and to proxy techniques. This list of keywords is then compared with all records of the known proxy IP addresses to determine the true positive rate. The WHOIS and rDNS records are examined separately. This means the following four experiments are performed:

- Implementation-specific keywords on WHOIS records;
- General proxy-related keywords on WHOIS records;
- Implementation-specific keywords on rDNS records; and
- General proxy-related keywords on rDNS records.

5.3.1 WHOIS

The WHOIS records of the known IP addresses of the selected proxy implementations were used to create a detection method based on the text in the WHOIS records. After careful examination some keywords were found that had a strong relationship with the proxy provider in question. Other keywords were found that have a weaker relationship and could be prone to false positives.

Implementation-specific keywords

Tor project. From the 935 known Tor exit nodes, the WHOIS records were retrieved. Of these records 83 have the text '`tor exit`' or '`tor-exit`' in them (e.g. netname: Tor-EXIT-HVIV), 39 contain the text '`torproject`', and 21 have other tor-related texts (e.g. "`tor server`"). Based on the extracted rules, 93 IP addresses were identified as proxies. This leads to the following keywords (here shown as regular expression rules as commandline arguments for `grep`):

```
grep -aiE "tor[ -]?exit"
grep -aiE "torproject"
grep -aiE "tor node"
grep -aiE "tor server"
grep -aiE "tornet"
```

ExpressVPN. From the 689 known ExpressVPN proxies, the WHOIS records were retrieved. Of these records 104 have the text '`express vpn`' or '`expressvpn`' in them (e.g. abuse-mailbox: abuse@expressvpn.com), 149 have the text '`vpn consumer`' with or without a space (e.g. abuse-reports@vpnconsumer.com), and one the text "vpn-services" (e.g. netname: VPN-Services). Based on these rules 254 IP addresses were identified as proxies. This leads to the following keywords (here shown as regular expression rules as commandline arguments for `grep`):

```
grep -aiE "express\s?vpn"
grep -aiE "vpn-consumer-|vpn consumer"
grep -aiE "vpn-services"
```

NordVPN. From the 4848 known proxies, the WHOIS records were retrieved. Of these records 507 contains the text '`nordvpn`' (e.g. abuse-mailbox: abuse@nordvpn.com), and 297 the text '`hypervpn`' (e.g. NetName: HYPERVPN). Based on these rules, 804 IP addresses were correctly identified as proxies. This leads to the following keywords (here shown as regular expression rules as commandline arguments for `grep`):

```
grep -aiE "nordvpn|hypervpn"
```

When testing this method with the lists of known proxy addresses, the following results were observed:

Table 13 Test results of implementation-specific keywords in the WHOIS database

WHOIS lookup type	Keywords	Population of known proxies	# of IPs (n)	TPR
Tor-specific	"tor[-]?exit", "torproject", "tor node", "tor server", "tornet"	Tor IPv4 nodes	935	0.10
ExpressVPN-specific	"express\s?vpn" "vpn-consumer-|vpn consumer", "vpn-services"	ExpressVPN	689	0.38
NordVPN-specific	"nordvpn", "hypervpn"	NordVPN	4848	0.10
Total	**All above**	**Tor & ExpressVPN & NordVPN**	**6472**	**0.13**

General proxy-related keywords

For this method general keywords related to proxy services are created. The keywords are created based on terms presumed typical for proxies and by examining random selected WHOIS records of known proxies:

```
grep -aiE "vpn|proxy|openvpn|ipsec|l2pt|pptp|sstp|socks"
```

These keywords are then searched in the WHOIS records of the known proxy's IP addresses with the following results:

Table 14 Test results of proxy-related keywords in the WHOIS database

WHOIS lookup type	Keywords	Population of known proxies	# of IPs (n)	TPR
Proxy-related keywords	(see above)	Tor IPv4 nodes	935	0.012
Proxy-related keywords	(see above)	ExpressVPN	689	0.29
Proxy-related keywords	(see above)	NordVPN	4848	0.17
Total	**(see above)**	**Tor & ExpressVPN & NordVPN**	**6472**	**0.16**

5.3.2 Reverse DNS

The reverse DNS (rDNS) records of known IP addresses of the selected proxy implementations were used to create a detection method based on rDNS records.

Implementation-specific keywords

After careful examination only one keyword was found that had a strong relationship with the proxy service in question, namely the name of the provider. The results include the number of IP addresses that have a reverse DNS record (column '# resolve'). The TPR are over the total number of IPs.

Table 15 Test results of implementation-specific keywords in the rDNS database

rDNS lookup type	Keywords	Population of known proxies	# of IPs (n)	% resolve	TPR
Tor-specific	"tor"	Tor IPv4 nodes	935	83%	0.37
ExpressVPN-specific	"expressvpn"	ExpressVPN	688	33%	0.0045
NordVPN-specific	"nordvpn"	NordVPN	4848	29%	0.0043
Total	**All of above**	**Tor & ExpressVPN & NordVPN**	**6471**	**37%**	**0.12**

General proxy-related keywords

For this method general keywords related to proxy services are created. The keywords are created based on terms presumed typical for proxies and by examining random selected rDNS records of known proxies:

```
grep -aiE "vpn|proxy|openvpn|ipsec|l2pt|pptp|sstp|socks"
```

Table 16 Test results of general proxy-related keywords in the rDNS database

rDNS lookup type	Keywords	Population of known proxies	# of IPs (n)	% resolve	TPR
Proxy-related keywords	(see above)	Tor IPv4 nodes	935	83%	0.019
Proxy-related keywords	(see above)	ExpressVPN	688	33%	0.0089
Proxy-related keywords	(see above)	NordVPN	4848	29%	0.0072
Total	**(see above)**	**Tor & ExpressVPN & NordVPN**	**6471**	**37%**	**0.011**

5.3.3 Conclusion

The implementation-specific keywords in WHOIS records gives some results (TPR=0.13), as do the general proxy-related keywords in WHOIS records (TPR=0.16). When considering the reverse DNS lookup, there are rather few results from using this method. Only a handful of IP addresses have an rDNS record that show them to be proxies (TPR=0.0074). The Tor IP addresses are a relative exception with 15 hits (TPR=0.018). In addition, the rDNS records were not always available. Only 37% of the tested IP addresses had an rDNS record.

It was also observed that although each IP address has an WHOIS record, the details of these records are not always the same. Some records encompass a wide range of IP addresses, while others

have more specific records of small subranges. These records show more detailed information on the usage of the IP. This difference was not measured in the experiments.

5.4 RESIDENTIAL VS. DATACENTRE IPS USING WHOIS AND RDNS

This method assumes that it is more likely that proxies are located in datacentres and at hosting providers than at IP addresses designated for residential users. The method uses lookups in the IP-related databases WHOIS and rDNS to determine the usage of the IP address.

An important success factor of this method is the accuracy. This is partly tested by answering the question:

What is the predictive values of a detection method based on the usage of known proxy IP addresses from the IP-related databases WHOIS and rDNS?

Some elaboration on the details is needed arrive at an acceptable method to answer this question. In anticipation, the following three intermittent questions need to be answered:

I. How to retrieve a list of keywords related to datacentres and hosting providers?

II. How to retrieve a list of keywords related to residential users?

III. How to create a dataset representing residential users to test the keywords?

5.4.1 Research design

Determine keywords

As previously described, the information on the usage of an IP address can be found in the WHOIS and in the reverse DNS records (see 4.3.1). There is however a small caveat here. The information in these records does not unambiguously identify the usage of an IP address. This is because the owner of the IP address can fill these records as he or she sees fit. A solution to this is to define keywords that are typical indicators for the usage of the IP address. Two lists of keywords are created: (1) datacentre- and hosting-related keywords, and (2) residential user-related keywords.

In turn, to find the datacentre-related keywords, two methods are used: (1a) using IP addresses of known proxies, and (1b) using the names of datacentre and hosting providers. The WHOIS and rDNS records of the known IP addresses of Tor, ExpressVPN and NordVPN are used to find datacentre- and hosting-related keywords in a similar way as described in Paragraph 5.3 (method 1a). In addition to this method, the commercial database Udger provides a list of names of datacentres and hosting providers (“Udger” n.d.) (method 1b). These names are added to the list of datacentre-related keywords.

Finding residential-related keywords is a more difficult task, because it is difficult to find a list of verified known residential IP addresses. Therefore, an explorative method is used where brainstorming activities lead to potentially related keywords that, in turn, are verified using a large repository of WHOIS records. The WHOIS records with these keywords are then explored for more related keywords.

To find residential-related keywords to be used to search reverse DNS records, the research by the internet user ValdikSS is used. ValdikSS (real name unknown) has done research in detecting

proxy connections. In his code repository regular expressions were found that are related to residential users (ValdikSS 2015).

Dataset representing residential users

The next difficulty is to find a dataset that is representative for residential users. A by-product of finding residential-related keywords, as described in the previous paragraph, are the WHOIS records that are likely from residential users. These WHOIS records are used to test the datacentre-related keywords for false positives and true negatives.

To test the reverse DNS records, the IP ranges in these (likely) residential WHOIS records are used. The rDNS records of a number of random selected of IP addresses are retrieved and used as a dataset to test both the datacentre keywords and the residential keywords.

It should be noted that from a research perspective it can be questioned whether this dataset is fully representative; when a dataset is obtained using a different method than what is being tested, the test results are considered more objective. That is not the case here.

Research method

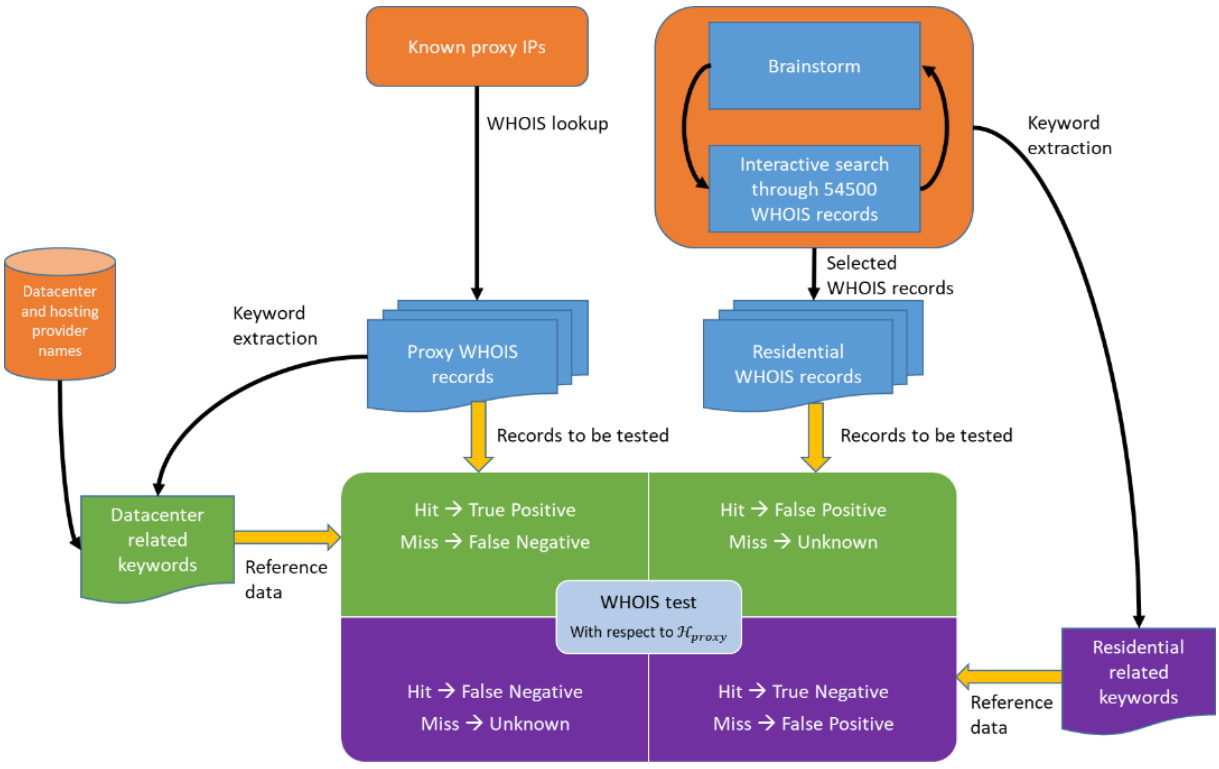

Figure 15 Method for testing residential vs. datacentre WHOIS records

The previous leads to the following method (also depicted in Figure 15 for WHOIS, and Figure 16 for rDNS records):

1. A list of datacentre-related keywords is created using
    a. known proxy IP addresses of Tor, ExpressVPN and NordVPN:
        i. The WHOIS and rDNS records are retrieved; and
        ii. Keywords in these records are extracted that are related to datacentres
    b. a list of datacentre names from the Udger database
2. A list of residential-related keywords is created using:

a. For the WHOIS based detection:
    i. Brainstorm for potential keywords related to residential users; and
    ii. Interactively browse and search through a large number of WHOIS records to identify new keywords
b. For the rDNS based detection: the keywords are used from (ValdikSS 2015)

3. The true positive and false negative rates are determined using:
    a. Tests with IP addresses of known proxies:
        i. The WHOIS and rDNS records of these IP addresses are searched for the occurrence of the keywords in the two lists
        ii. This results in the TPR and FNR for both lists
    b. Test with IP addresses that are strongly suspected to be residential users
        i. The WHOIS records that are suspected to be residential users are used (a by-product of the previous step 2)
        ii. These WHOIS records are tested against the set of datacentre-related keywords to find the false positives rate. Note that these WHOIS records cannot be tested against the list of residential-related keywords, since they are derived from these WHOIS records.

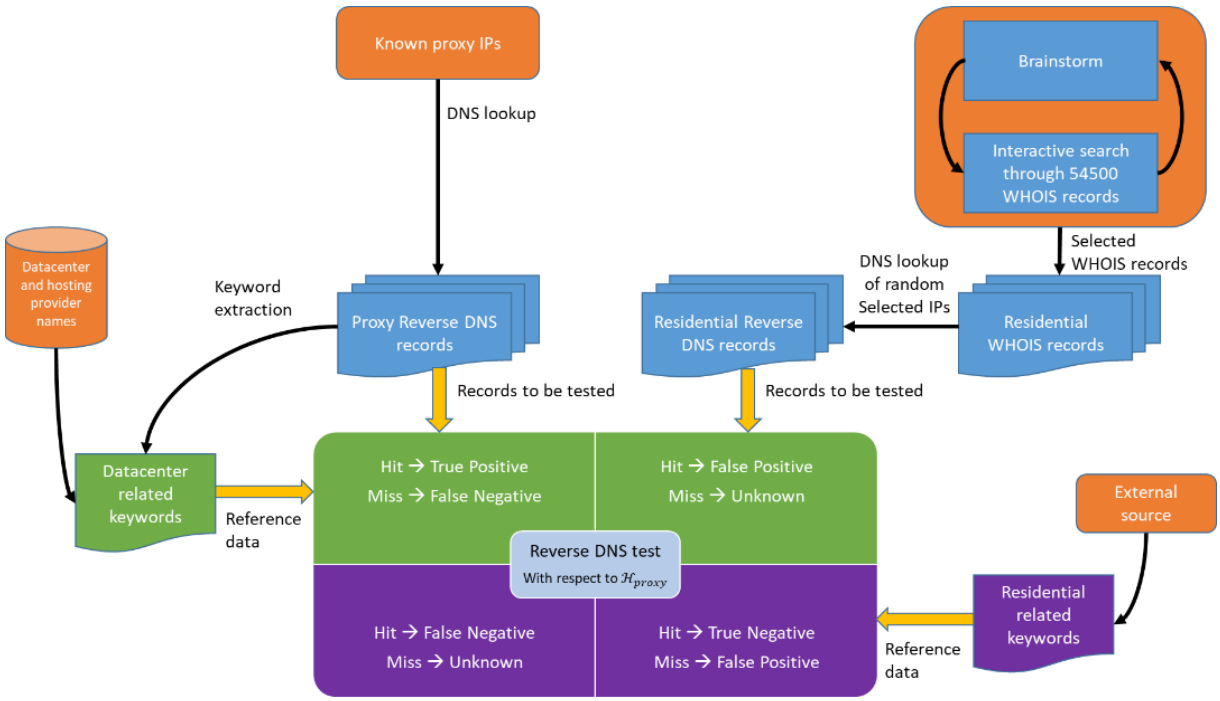

Figure 16 Method for testing residential vs. datacentre rDNS records

Note that a 'hit' in the test means a keyword was found in the tested record, and a 'miss' that the keyword was not found. In two cases a 'miss' reveals no information on the tested record. This is because the two tests are assumed to be unrelated and the results independent to the opposite hypothesis. Also note that the "Residential WHOIS records" are considered a weak reference dataset.

5.4.2 Determining WHOIS keywords

Datacentre-related keywords

The WHOIS records of three popular VPN proxy providers, Tor, ExpressVPN, and NordVPN, are selected. If the assumption is correct that proxies are more likely to originate from IP addresses of hosting providers, then the information of the owner of the netblock that has specific hosting-related keywords in the WHOIS record is more likely a proxy. The method here is to find general hosting provider-related keywords in the retrieved WHOIS records. After careful examination of these records the following keywords were found:

```
dedicated
dedicateserver
colocation
hosting
```

Based on the same assumption, when the company name of a known hosting provider is in the WHOIS record it is more likely the IP is located in a datacentre. With this lookup method the list of datacentres from the Udger database is used to search the WHOIS records. Udger (www.udger.com) is a commercial service that focuses on user agent strings. In addition, they also provide information on IP addresses that they have categorised in crawlers, public proxies, public cgi proxies, VPN services, Tor exit nodes, Fake crawlers, and Web scrapers ("Udger" n.d.). The datacentre database contains 1054 names of companies that are associated with datacentres. According to the Udger website (sic): "*If the IP address is in a range of data centres, it is not the end user, but it's a program for 99.9%.*"

Residential-related keywords

To obtain residential-related keywords, a list of potential keywords is created. These keywords are then searched in a collection of 54,500 WHOIS records. The records that contain the keywords are examined to determine whether they indeed concern residential users. New keywords are then explored in these records, repeating the process. This time-framed processes resulted in the following keywords (represented as regular expressions for the commandline tool `grep`):[§§§]

```
grep -aiE "residential (broadband|ethernet|adsl|television
     and|sip)?( )?(internet|customer|service|users|network|adsl|pool|
     conectivity|pppoe|subscribers|connectivity|isp) " | grep -vaiE
     "presidential"
grep -aiE "(internet |client )?pppoe (customers|pool)"
grep -aiE "home internet"
```

5.4.3 Determining rDNS keywords

To find datacentre and residential use of IP addresses in reverse DNS, different keywords need to be identified to these in the WHOIS records.

Datacentre-related keywords

General keywords. The rDNS records of the known IP addresses of the selected proxy implementations are used to create a detection method based on rDNS records. After careful

[§§§] The keyword '`residential`' gives a false hit on records containing '`presidential`'. This is excluded by the '`grep -v`' command.

examination the following keywords are found that have a strong relationship with datacentres or hosting providers:

```
hosted-by
baremetal
dedicated
.colo.
```

Keywords like 'static' and names with autonomous system numbers (ASN) in them are ignored because these are considered not specific for datacentres and hosting providers. Patterns with the IP address (e.g. `ip-static-188-42-255-162.server.lu`) are ignored for the same reason.

Datacentre names. The list is appended with the domain names found in the Udger database of datacentres names. To reduce the possibility of false positives the top-level domains are included, for example the keyword `ai.net` is used and not the keyword `ai`.

Residential-related keywords

Based on the work by (ValdikSS 2015) the following keywords are used to identify residential users in the rDNS records (expressed as regular expressions for the commandline tool `grep`):

```
grep -aiE "ppp-"
grep -aiE "dsl-pool"
grep -aiE "[\.]?(pool|pppoe|adsl|dsl|xdsl|dialup|broad|cust-
           adsl|dynamicip|dynamicIP|dyn)[\.-]?+"
```

5.4.4 Dataset representing residential users

The residential-related keywords for WHOIS records are searched in the WHOIS records of 54,500 autonomous systems (AS). This resulted in 364 ranges of IP addresses encompassing 632,068 IP addresses. A random selection of 1,000 IP address is made, evenly distributed from the IP ranges. These 1,000 IP addresses are used as a dataset to test the datacentre-related keywords for false positives.

5.4.5 Test results

WHOIS records

The previously described method leads to the following test results:

Table 17 Datacentre-related keywords in known proxy WHOIS records

WHOIS lookup type	Keywords	Population known proxies	# of IPs (n)	TPR
Datacentre keywords only	"dedicated", "dedicateserver", "colocation", "hosting"	Tor & ExpressVPN & NordVPN	6390	0.094
Datacentre names only	1054 datacentre names	Tor & ExpressVPN & NordVPN	6390	0.61
All datacentre-related keywords	**Datacentre-related**	**Tor & ExpressVPN & NordVPN**	**6390**	**0.66**

Table 18 Datacentre-related keywords in selected residential WHOIS records

WHOIS lookup type	Keywords	Population	# of IPs (n)	FPR
Datacentre keywords only	"dedicated", "dedicateserver", "colocation", "hosting"	Selected residential IPs	1000	0.00
Datacentre names only	1054 datacentre names	Selected residential IPs	1000	0.096
All datacentre-related keywords	**Datacentre-related**	**Selected residential IPs**	**1000**	**0.096**

Table 19 Residential-related keywords in known proxy WHOIS records

WHOIS lookup type	Keywords	Population known proxies	# of IPs (n)	FNR
Residential	Residential-related	Tor exit nodes	934	0.0
Residential	Residential-related	ExpressVPN	610	0.0
Residential	Residential-related	NordVPN	4846	0.0
Total	**Residential-related**	**Tor & ExpressVPN & NordVPN**	**6390**	**0.0**

Reverse DNS

The previously described test method leads to the following test results:

Table 20 Datacentre-related keywords in known proxy rDNS records

rDNS lookup type	Keywords	Population known proxies	# of IPs (n)	% resolve	TPR
Datacentre keywords only	"hosted-by", "baremetal", "dedicated", ".colo. "	Tor & ExpressVPN & NordVPN	6681		0.019
Datacentre names only	1054 datacentre names	Tor & ExpressVPN & NordVPN	6681		0.14
All datacentre-related keywords	**Datacentre-related**	**Tor & ExpressVPN & NordVPN**	**6681**	**36%**	**0.15**

Table 21Datacentre-related keywords in selected residential rDNS records

rDNS lookup type	Keywords	Population	# of IPs (n)	% resolve	FPR
Datacentre keywords only	"hosted-by", "baremetal", "dedicated", ".colo. "	Selected residential IPs	1000		**0.00**
Datacentre names only	1054 datacentre names	Selected residential IPs	1000		**0.002**
Datacentre	**Datacentre-related**	**Selected residential IPs**	**1000**	**72%**	**0.002**

Table 22 Residential-related keywords in known proxy rDNS records

rDNS lookup type	Keywords	Population known proxies	# of IPs (n)	FNR
Residential	Residential-related	Tor exit nodes	934	0.00
Residential	Residential-related	ExpressVPN	610	0.00
Residential	Residential-related	NordVPN	4846	0.00
Total	**Residential-related**	**Tor & ExpressVPN & NordVPN**	**6390**	**0.00**

Table 23 Residential-related keywords in selected residential rDNS records

rDNS lookup type	Keywords	Population known proxies	# of IPs (n)	% resolve	TNR
Total residential	Residential-related	Selected residential IPs	1000	72%	0.46

5.4.6 Conclusion

When looking at WHOIS records, 66% of the known proxies have datacentre-related keywords in their WHOIS record. In addition, these keywords are relatively rare in WHOIS records of residential IPs (9.6%). Residential-related keywords were not seen in WHOIS records of known proxies. The prediction values of the test are PPV=0.98 and NPV=0.23. This leads to the conclusion that, under the assumption that proxies are more commonly hosted in datacentres then that they are hosted by residential users, a detection method using keywords in WHOIS records is a good test for detecting proxies. However, due to the low negative predictive value, a negative test result provides little information. This means that when an IP address has a datacentre-related keyword in its WHOIS record, it is probably a proxy, and if it does not, it is unknown if it is a proxy or not.

With respect to rDNS records, 15% of the known proxies have datacentre-related keywords in there rDNS records, while only 0.2% of residential IP addresses have these keywords. With the predictive values of PPV=0.998 and NPV=0.14, under the same assumptions as the before, it is also a good test for detecting proxies, while negative test results provide little information.

Residential-related keywords are present in 46% of residential used IP addresses, while these are not present in the known proxy records. So, under the same assumptions as before, when a residential-related keyword is present in the rDNS record it is not a proxy. The prediction values of this test are PPV=0.92 and NPV=1.

Note that the above result are based on a data set that only weakly represents the real incoming connections.

5.5 Port and service scanning

This detection method is based on the active scanning of the IP address of incoming connections. It is assumed that scanning for ports and services reveals information on the use of the IP address, and on whether it is a proxy or not. An important success factor of this method is its feasibility. This is partly tested by answering:

Can OpenVPN and IPsec proxy implementations be scanned to reveal proxy-related information?

This is tested by probing known OpenVPN and IPsec servers with specialist tools.

During this test two observations were made. The first is that not all providers use the default ports. To probe multiple ports can take a long time and therefore it is helpful to know what ports the proxy service is listening, as that drastically decreases the time it takes to probe for the services ("IBM Knowledge Center - Scan Duration and Ports Scanning" n.d.).

The second observation concerns OpenVPN TLS authentication. When the service is configured to use TLS authentication on a UDP port, the service only responds to correctly authenticated probe packets. If the authentication key is known it would be possible to identify the proxy. If the key is shared for all the proxies of the proxy provider, unknown proxies of the same provider can also be identified. It is therefore helpful to know if OpenVPN proxy providers do this.

This leads to the following sub-questions:

I. *What TCP ports are used for OpenVPN services?*
II. *How often is an OpenVPN TLS authentication key used by a proxy provider, and how often are the keys shared between servers?*

5.5.1 Experiment and results

Probing known proxies

A quick search on the internet resulted in at least two tools for probing proxy servers. The tool `check_openvpn.py` is available on GitHub and can query an IP address on the presence of a OpenVPN server (liquidat [2013] 2018). For the detection of IPsec the tool `ike-scan` is used, also available on GitHub (Hills [2013] 2018). These tools were tested using the known proxy IP addresses of ExpressVPN and NordVPN.

ExpressVPN – OpenVPN test. In all observed downloadable configuration files, port 1195 is configured instead of the commonly used port 1194. In addition, the same TLS authentication key is used on all observed configuration files. Probing the OpenVPN service on port 1195 with the shared TLS authentication key revealed all IP addresses as proxies.

ExpressVPN – IPsec test. According to the website, the ports 500/UDP and 4500/UDP are used for L2TP/IPsec and IKEv2. Using the `ike-scan` tool, all 302 servers were revealed by returning a handshake.

NordVPN – OpenVPN test. All NordVPN proxy servers are configured to port 1194/UDP and port 443/TCP. Examining the configuration files, 3087 of the 4750 servers use a shared tls-auth key. The remaining 1663 servers use an individual tls-auth key. Probing the servers revealed all as proxies.

NordVPN – IPsec test. All 22 servers configured for L2TP over IPsec returned a handshake using the `ike-scan` tool. In addition, all 4601 servers configured for IKEv2 IPsec returned a notify, revealing all the as proxies.

Ports used by proxy providers

According to the documentation on OpenVPN, the default ports of OpenVPN are the port numbers 1194/UPD and 1194/TCP. The port TCP/443 is also given as an example in the official OpenVPN manual ("Manuals" n.d.). An extensive search on the internet revealed the following default port configuration of 37 VPN providers (for details see Appendix II):

Table 24 Default OpenVPN and IPsec ports of known providers

OpenVPN Default UDP port	53, 80, 88, 111, 118, 123, 149, 150, 151, 443, 553, 992, 1149, 1150, 1151, 1191, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1301, 1302, 1912, 2049, 2460, 3389, 4000, 5005, 5555, 8292, 9201, 12200, 26000
OpenVPN Default TCP port	22, 80, 88, 443, 992, 1194, 1912, 2460, 3389, 5005, 5555, 8008, 12200, 26000
IPsec default UDP port	500, 1701, 4500

OpenVPN Authentication

An inspection of the configuration of 37 VPN providers revealed the following on the use of authentication (for details see Appendix II):

Table 25 OpenVPN authentication of known providers

OpenVPN Authentication	**n=37**
None	21 (57%)
Shared Key	14 (38%)
Key not shared	2 (5%)

5.5.2 Conclusion

Probing for services and open ports is a good technique for detecting proxies. All known IP addresses of proxies were revealed when scanning for OpenVPN and IPsec services. Based on the default port configuration of 37 commercial providers, the time to probe the non-default ports can be drastically reduced by only probing the 53 commonly used ports. OpenVPN services of commercial proxy providers can be successfully probed 95% of the time, because the authentication is off or the key is shared. The remaining 5% use individual keys for each server.

5.6 TIMING INTERNET-LAYER PROXIES

This method assumes that there is a relation the usage of an IP (proxy or direct) and the time difference between the full-round-trip time of the three-way handshake and the ICMP-ping time from the server to the originating IP address. During the study, no proof-of-concept projects or literature was found on this technique. Therefore, rudimentary explorative study has been performed to answer the question:

Can a distinguishing be made between a proxy and a direct connection by comparing the full-round-trip time of the three-way handshake, and the ICMP-ping time between the server and the originating IP address?

To answer this question, two experiments have been conducted. In the first experiment, time measurements have been made using known direct and proxy connections. In the second experiment various techniques described in the previous paragraphs have been combined to identify incoming connections.

5.6.1 Test setup

A test server has been setup on the internet hosting a Secure Shell (SSH) service. When the test server receives an incoming connection, the server automatically performs an ICMP ping to the originating IP address. For each incoming connection the time difference (Δ_{time}) is determined between the three-way handshake (the full-round-trip time, as is described in Paragraph 4.5.2) and the ICMP-ping time. In addition, the WHOIS and rDNS record are retrieved for the incoming connection, and the originating IP address is probed for OpenVPN and IPsec services on the default port numbers.

5.6.2 Test using known originators

For this experiment, connections are made to the test server with and without using a proxy. For the direct connections three different originating addresses are used: a residential Digital Subscriber Line (DSL), a mobile device using 4G, and a Virtual Private Server (VPS). For the proxy connections the ExpressVPN client is used, using eight different proxy locations. In order to circumvent any possible caching, all the connections used the SSH service to connect to.

Test results

The results of the tests with known direct and proxy connections:

Figure 17 The Δ_{time} of known direct and proxy connections (times in ms)

Direct/ proxy	Conn. Source	# Meas.	Min.	Average	Max.	Std.dev.
Proxy	Hong Kong	3	0.2129	**0.2211**	0.2356	0.0103
Proxy	Vietnam	1054	0.1738	**0.2184**	2.3432	0.1731
Proxy	Mexico	13	0.1557	**0.2162**	0.4650	0.0954
Proxy	US, New York-2	799	0.0925	**0.1713**	3.2123	0.1899
Proxy	US, New York-1	1209	0.0923	**0.1485**	4.331	0.1776
Proxy	ES, Madrid	41	0.0510	**0.0657**	0.1215	0.0172
Proxy	NL, Den Haag	3	0.0235	**0.0352**	0.0482	0.0101
Proxy	UK, Kent	3	0.0292	**0.0317**	0.0331	0.0018
Direct	VPS	11	-0.0060	**0.0094**	0.0299	0.0123
Direct	DSL connection	20	-0.0224	**-0.0142**	0.0144	0.0112
Direct	Mobile 4G	16	-[****]	-	-	-

The number of connections that were measured is displayed in column '# Meas.'. Of these measurements, the minimum, average, maximum, and standard deviation were calculated.

Based on the above results, the following thresholds is determined:

$$threshold_{proxy} = 0.03$$
$$threshold_{direct} = 0.01$$

[****] The IP address did not respond to ICMP-ping probes.

This can be used in the following classification algorithm:

$$\Delta_{time} > threshold_{proxy} \rightarrow IP\ is\ likely\ a\ proxy$$
$$\Delta_{time} < threshold_{direct} \rightarrow IP\ is\ likely\ a\ direct\ connection$$

Conclusion

It is possible to distinguish an internet-layer proxy and a direct connection based on the three-way handshake and ping time. It is a direct connection if the time difference is negative or close to zero. A relatively large positive time difference indicates a proxy. The results also shows there is a relation between the time difference and the distance between the client and the server. The test server is located in New York (US) and the client in Leiden (NL). The greater the distance between the client and the proxy, the larger the time difference is.

5.6.3 Test using combined methods

For this experiment, the test server collected data on all incoming connections during a period of five months. During this time, approximately 291,000 three-way handshakes where collected originating from approximately 15,500 unique IP addresses. A selection of these IP addresses are labelled as proxy using the following methods:

- Lookup in lists of known proxies for Tor, NordVPN, and ExpressVPN;
- Provider-specific keywords in WHOIS and rDNS records (for the providers Tor, NordVPN, and ExpressVPN);
- Proxy-specific keywords in WHOIS and rDNS records;
- Probing for OpenVPN services on the IP addresses;
- Probing for IPsec services on the IP addresses.

After these IP addresses were identified, the timing results of these labelled IP addresses are collected and analysed.

Test results

Searching through the collected incoming connections produced the following results:

Figure 18 Time differences of identified proxies by using different methods

Detection method	# identified IPs	# Meas.	Min.	**Average**	Max.	Std.dev.
Known proxies	18	3133	-0.0007	**0.1764**	4.3311	0.1808
Provider-specific keywords in WHOIS and rDNS records	12	3078	0.0002	**0.1784**	4.3311	0.1817
Proxy-specific keywords in WHOIS and rDNS records	83	3079	7e-05	**0.1784**	4.3312	0.1816
Probe OpenVPN	130	75	-0.0182	**0.0053**	0.4980	0.0586
Probe IPsec	147	3217	-0.3031	**0.1133**	11.6936	0.5143
All methods combined	**371**	**6069**	**-0.3031**	**0.1308**	**11.6936**	**0.3629**

Each method identified a number of IP addresses, included in the column '# identified IPs'. Of these IP addresses, a number of time measurements were collected (in column '# Meas.'). Of these measurements the minimum, average, maximum, and standard deviation were calculated. Only the connections to the SSH server were used. The results of the individual methods are combined, eliminating any overlap in the IP addresses and measurements.

Taking the previous determined thresholds, the identified proxies are categorised as followed:

Figure 19 Categorised measurements based on timing thresholds

Detection method	# Meas.	Proxy ($\Delta_{time} > 0.03$)	Direct ($\Delta_{time} < 0.01$)	Unknown
Known proxies	3133	3123	8	2
Provider-specific keywords in WHOIS and rDNS records	3078	3075	3	0
Proxy-specific keywords in WHOIS and rDNS records	3079	3077	2	0
Probe OpenVPN	75	2	69	4
Probe IPsec	3217	798	2168	251
Methods combined	**6069**	**3772**	**2053**	**244**

Conclusion

The results can be viewed from two perspectives. If it is assumed that all the used individual detection methods are correct, the true positive rate (TPR) and false negative rate (FNR) of the timing method are TPR=0.62 (3772/6069), and FNR=0.39 (2053+244/6069). This would make the timing method a reasonable good method to identify proxies. On the other hand, if it is assumed that the timing method is correct, the true positive and false positive rates of the individual methods are:

Figure 20 the TPRs and FNRs of the methods assuming the timing methods is correct

Detection method	# Meas.	TPR	FPR
Known proxies	3133	0.997	0.003
Provider-specific keywords in WHOIS and rDNS records	3078	0.999	0.001
Proxy-specific keywords in WHOIS and rDNS records	3079	0.999	0.001
Probe OpenVPN	75	0.027	0.92
Probe IPsec	3217	0.25	0.67

This shows that the probing method produces a lot of false positives, while the other methods have a very low false positive rate.

5.6.4 Other observations

Negative time difference

In Figure 21 the Δ_{time} is shown of all the random incoming connections to the test server during the test period. The time difference of most of the connections is near zero. However, it was observed that the three-way handshake can be faster than the ICMP-ping time. This is indicated by the negative time difference. It was expected that the ICMP-ping time would always be faster than the three-way handshake, or at least equally fast. This might be an indication that the ICMP ping is not a good time reference. More study is needed to reach a definitive conclusion.

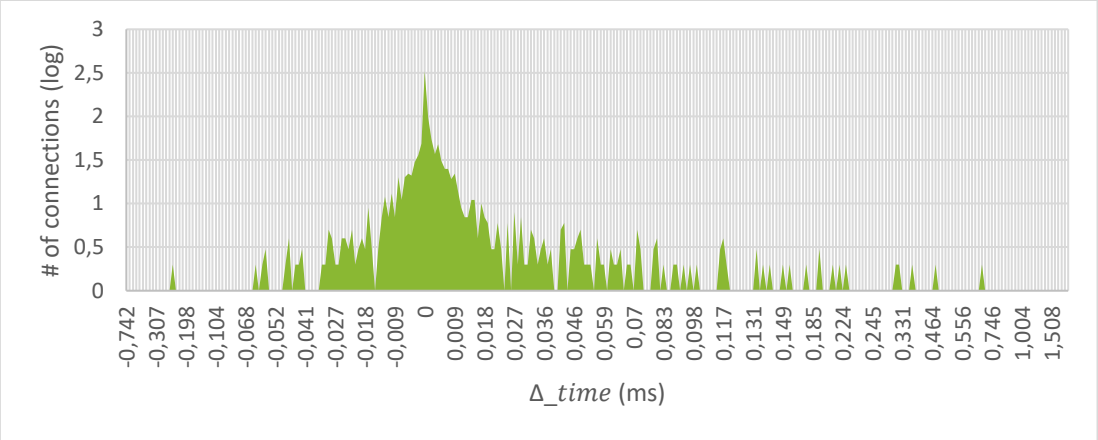

Figure 21 The Δ_{time} of 13,878 random connections.

Proxy-direct a priori odds

During the test period 27,758 connections were made to the test server. Assuming the timing method is correct and using the above determined thresholds, the odds an incoming IP address is a proxy over a direct connection can be calculated by (n=27,758):

$$\frac{P(\mathcal{H}_p)}{P(\mathcal{H}_d)} = \frac{4{,}412}{22{,}220} = 1{:}5.0 \approx 0.20$$

The total incoming connections originate from 1,985 unique IP addresses. Using the previous timing method, 123 of these IPs are classified as proxy, 1,599 as direct, and 185 are classified as both. Based on this, the odds can be calculated by (n=1985):

$$\frac{P(\mathcal{H}_p)}{P(\mathcal{H}_d)} = \frac{123}{1{,}599} = 1{:}13 \approx 0.078$$

This shows that the original estimation of the odds (1:184, see Paragraph 2.2.2) might be an incorrect estimation.

Processing time

The time difference Δ_{time} is not only influenced by the propagation times of the packets in transit, but also by the processing time of the client, proxy and the server (see Paragraph 4.5). In Figure 22 the time differences is shown of a known originating address over a 24-hour period. During this period, the server was making a five backups, making the server more busy. The five largest spikes in the figure coincide perfectly with the creation time of the backups. This effect can make the time measurement of a direct connection exceed the threshold and incorrectly classifying it as a proxy.

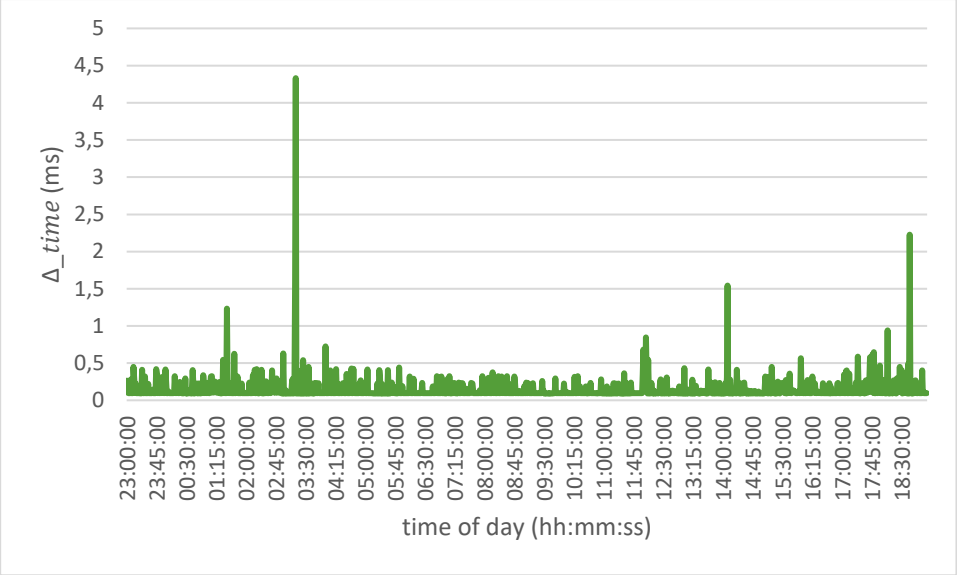

Figure 22 Time difference of a connection over a 24-hour period

5.6.5 Conclusion

Time measurements can identify internet-layer proxies. Measuring the full-round-trip time of the three-way handshake and comparing it to the ICMP-ping time can distinguish a proxy from a direct connection. However, the number of direct connections that were used in this test was limited. More experiments are needed with a representative dataset to confirm the conclusion. In addition, when the client, proxy or server is busy, the increase in processing time causes the timing measurement to be inaccurate.

5.7 MSS FINGERPRINTING

This detection method assumes there is a correlation between the advertised MSS value in the TCP three-way handshake and the use of a proxy. Therefore, one of the success factors is that this relation exists. This is partially tested for OpenVPN:

What OpenVPN settings influence the advertised TCP MSS option and how?

In addition, to get an impression on how unique MSS values are in random traffic:

What are the MSS values of a large number of random connections?

5.7.1 MSS value and OpenVPN

In a research project the internet user ValdikSS discovered that the MSS size of OpenVPN connections depends on the OpenVPN settings (ValdikSS 2015). A relation was found between the advertised MSS values in the TCP three-way handshake and the following settings of OpenVPN:

- Protocol used (UDP or TCP)
- Block size of encryption method
- MAC algorithm used
- Compression (on or off)

Various settings of OpenVPN were tested in a lab environment. The results by ValdikSS were verified and additional MSS values for new settings were found:

Table 26 TCP MSS values for various OpenVPN settings

Protocol	Block size	MAC	Compression	MSS
UDP	64	SHA1	Off	1369
UDP	64	SHA1	On	1368
TCP	64	SHA1	Off	1367
TCP	64	SHA1	On	1366
UDP	128	SHA1	Off	1353
UDP	128	SHA1	On	1352
TCP	128	SHA1	Off	1351
TCP	128	SHA1	On	1350
UDP	128	SHA256	Off	1341
UDP	128	SHA256	On	1340
TCP	128	SHA256	Off	1339
TCP	128	SHA256	On	1338
UDP	128	SHA512	Off	1309
UDP	128	SHA512	On	1308
TCP	128	SHA512	Off	1307
TCP	128	SHA512	On	1306

Unfortunately, when the OpenVPN option `--mss-fix` is used, every TCP connection to the server is advertised with the MSS value in this setting. The default value is `--mss-fix 1450`, making it indistinguishable from direct connections using Ethernet. This setting is used by the tested commercial proxy providers ExpressVPN and NordVPN. The default installation of OpenVPN (on Ubuntu) does not set this option, making it making it possible to detect it by MSS values. Therefore, the detection using the MSS value is more likely to work for proprietary proxy setups.

5.7.2 MSS Reference values

To create an indication on the spread and range of MSS values, a test is performed with data collected from $2x10^6$ random unique IP addresses. For each of these IP addresses the advertised MSS values were extracted from the three-way handshake. This resulted in 302 different MSS values ranging from 60 to 65516 bytes. The top 20 values are:

Table 27 Top 20 MSS values

MSS	Percentage of total	MSS	Percentage of total
1460	36.6%	1260	0.7%
1452	18.1%	1424	0.6%
1440	10.3%	1300	0.6%
1412	9.3%	536	0.6%
1420	4.0%	1402	0.5%
1400	3.9%	1370	0.5%
1360	3.5%	1414	0.4%
1380	1.9%	1358	0.4%
1410	1.7%	1432	0.3%
1398	1.2%	1448	0.3%

In order to determine if the MSS value is a good indicator for determining proxies or direct connections, some reference MSS values must be known. It turns out that this is not an easy task, as there is a large variation in the advertised MSS value and how it is determined. Some experiments show that MSS values can vary per application on the same computer. Mobile devices are also known for using a wide range of MSS values.

5.7.3 Conclusion

The following OpenVPN settings influence the MSS value of connections: protocol used (UDP or TCP), block size of encryption method, MAC algorithm used, and compression (on or off). This results in 16 different MSS values. Further study is needed to interpreted these MSS values and correlate them to the values found for specific OpenVPN settings. A preliminary conclusion is that when an MSS value is set in the TCP three-way handshake that equals the specific values for OpenVPN, it is more likely a proxy.

5.8 SUMMARY

Experiments were performed with some of the identified proxy detection methods in Chapter 4, in order to test some critical success factors. The results are presented below:

Table 28 Summary of all tests

Method Success factors tested	Results
List of known proxies	
- The information can be harvested from proxy providers	Yes
- The data harvested are complete	This is not always the case
- The harvesting of data is not difficult	It is somewhat labour-intensive. In some cases a subscription is needed to get the information.
Proxy databases	
- The proxy database is accurate	The true positive rate (TPR) of known public proxy providers is high (0.96). The overall TPR is 0.65. It is expected to be lower when tested with more commercial providers.
Provider- and proxy-related keywords in WHOIS and rDNS records	
- The records for a searched IP are available	WHOIS are always available, however specific records of the smaller subrange do not always exist. Only 37% of the tested rDNS existed.
- Keywords can be easily determined	This is somewhat difficult because it can lead to false positives.
- The keywords are a good indication for the use of an IP (low false positives)	This is moderately for WHOIS records; the provider-specific keywords have a TPR=0.13, and the general proxy-related keywords a TPR=0.16. This is fairly low for rDNS records; only a handful of IP addresses show it is a proxy (TPR=0.0074). The Tor IP addresses form a relative exception with a TPR=0.018. Implementation-specific keywords are likely to have few false positives. General proxy-related keywords are likely to have some false positives.
Datacentre- vs. residential-related keywords in WHOIS and rDNS records	
- The keywords are a good indication for the use of an IP (low false positives)	Yes. Of the tested known proxies, 66% have a datacentre-related keyword is present in their WHOIS records. These keyword are relatively rare in residential IP records (9.6%). Residential-related keywords where not seen in the WHOIS records of known proxies.

	Datacentre-related keywords are present in 15% of the rDNS records of known proxies. These keywords are only present in 0.2% of rDNS records of residential IP's. Residential-related keywords where not seen in the rDNS records of known proxies, while they are present in 46% of the known residential records.
Proxy port and service scanning	
- The technique to scan a proxy is available	Yes
- The proxy is susceptible to probing	Yes, all OpenVPN and IPsec services of known proxy providers included in the test were revealed.
- The port number of the listening proxy software is known	Testing 37 commercial OpenVPN providers showed 36 different UDP ports and 14 different TCP ports were used. However, exceptions exist were random ports are used.
- The authentication is turned off or the key is known	Testing 37 commercial OpenVPN providers showed 57% use no authentication, 38% a shared key, and 5% individual keys. This means that 95% can be probed.
Latency and timing	
- Measuring the TCP-handshake time and ICMP-ping time distinguishes a proxy from a direct connection	Yes, although more study is needed to confirm conclusion.
MSS fingerprinting	
- There is a relation between MSS value and the use of a proxy	For OpenVPN: yes. The following OpenVPN settings influence the MSS value of connections: protocol used (UDP or TCP), block size of encryption method, MAC algorithm used, and compression (on or off). This results in 16 different MSS values.
- The identified MSS values are unique for proxies	Unclear. Incoming connections can have a wide range of MSS values. More research is needed to determine this.

6 CONCLUSIONS AND FURTHER RESEARCH

It is difficult to identify the origin of an incoming internet connection. Does it originate from a proxy, or directly from a client? This study shows that it is possible to detect proxies by analysing incoming connections by means of specific detection methods. Despite this, a connection is not necessarily direct if no proxy is detected. This is similar to a doctor diagnosing a patient; eliminating known diseases does not mean the patient does not have a disease. To fully appreciate the result of the diagnosis, it is essential to know the capabilities and limitations of each detection method. Hence, this study presents a structured overview of these characteristics.

An exception to this elimination process is the timing of packets. Detection methods based on the timing of connections (e.g. measuring the timing of incoming connections) can distinguish proxy connections from direct connections.

6.1 DETAILED CONCLUSIONS

The study of proxy detection techniques requires proper conceptualisation of the problem and extensive study of the context. This context is provided in Chapter 2 along with the definitions for the related terms. There are many proxy techniques, and numerous ways of implementing them. A list of well-known and frequently used proxy techniques is provided in Chapter 3.

It is observed that the detection of proxies is a practical problem that very few scholars have examined. Most useful information was found on engineers' fora and from relevant proof-of-concept projects. Based on these sources, six basic detection techniques were identified. These techniques are applied in eleven more practical detection methods. A list of general properties was derived that characterises proxy detection methods. These properties can be found in Paragraph 4.1.1.

Chapter 4 presents a structured overview of the available detection methods with their properties, theoretical capabilities and limitations, as well as the critical success factors for them to work in practice (Paragraphs 4.2.3, 4.3.4, 4.4.6, 4.5.8, 4.6.3, and 4.7.1).

Finally, experiments on six selected detection methods show that all these methods can successfully detect proxies. The results on the tested accuracy and other critical success factors differ from method to method. A summary of capabilities and limitations of the six tested detection methods is provided below. A detailed overview of these results can be found in Paragraph 5.8.

6.2 SUMMARY OF SELECTED METHODS

List of known proxies

A simple method to detect proxies is to create **lists of known proxies** by harvesting IP addresses from proxy providers. Experiments have shown that this can be done for most proxy providers, although for some providers a subscription is necessary to access the information. A detection method using this technique is somewhat labour-intensive, because scripting and research is required for each provider in the preparation phase. When the detector produces a positive result (a 'hit'), it is almost certain that the incoming connection is a proxy. The biggest omission of this method is that unknown proxy providers and privately owned proxies are not detected. An important

advantage of this method is that the lists of known proxy IP addresses can be used for research and reference to other detection methods.

Proxy databases

Another simple method is the use of **proxy databases**. The IP address of an incoming connection can be queried in commercial databases that hold information on proxies. Experiments showed that a proxy database correctly detected 47% of the test set of known commercial IP addresses of proxies. The largest concern of this method is its dependence on third-party information; it is unknown how the information is acquired and how accurate it is. The omissions of this method are therefore also not known. It is also assumed that this method produces false positives.

Provider- and proxy-related keywords in WHOIS and rDNS records

Information about an IP address can be retrieved using WHOIS and DNS protocols. The retrieved records can contain information about its owner and its use. A method for detecting proxies is by searching for **provider- and proxy-related keywords in WHOIS and rDNS records**. Experiments with a number of known proxy IP addresses showed that proxies can be detected with this method, although only 16% were identified by using WHOIS records, and around 1% using rDNS records. This method is somewhat labour-intensive in the preparation, because keywords must be identified for each provider, and false positives must be eliminated manually. The biggest omission of this method is that privately owned proxies are not detected. An advantage over previous methods is that besides proxy providers, organisations using proxies that update WHOIS and rDNS records can also be detected.

Residential- vs. datacentre-related keywords in WHOIS and rDNS records

A common assumption is that proxy servers are more frequently located in datacentres than hosted on home users' systems. Assuming this is true, one way to detect proxies is by searching for **residential- and datacentre-related keywords in WHOIS and rDNS records**. Experiments have shown that if a datacentre-related keyword is present, it is more likely a proxy. In addition, if a residential-related keyword is present, it is most likely *not* a proxy. It is essential for this method to identify keywords that produce few false results. Its largest omission is that it is not conclusive; the IP address of a residential user can still be used as a proxy, and a hosted server can still connect directly. In addition, this method does not detect privately owned proxies.

Port and service scanning

An active detection method is to probe the originating IP address for information. Most proxies give away information on the presence of a proxy service in response to a prepared probing packet. Experiments with this **port and service scanning** technique identified *all* known proxies of the test set. A specific configuration of proxies using OpenVPN software was identified that does not respond to probing. This configuration is used by 5% of commercial proxy providers. The major advantage of this method over other methods is that privately owned proxies can also be detected. Its largest omission is proxy implementations where the pseudo-source address is not the same as the tunnel-endpoint address. In addition, it does not detect proxy implementations where ports and

services are obfuscated, for example where ports are closed immediately after connection, or when uncommon ports are used.

Timing internet-layer proxies

When the server-side packets can be captured, the timing of the communication can be measured. This method assumes that the **timing of internet-layer proxies** differs from direct communication, because of the extra communication path between the proxy and the client. This time difference is calculated by comparing the time it takes to complete a TCP three-way handshake and the timing of an ICMP-ping probe to the originator. Experiments have correctly identified all known proxies of the test set. In addition, it was shown that the time difference is related to the distance between the proxy and the client. An advantage of this method is that it can detect any *internet*-layer proxy, including privately owned proxies. Another advantage is that the method not only detects proxies, it can also differentiate between proxies and direct connections. The largest omission of this method is that *application*-layer proxies are falsely identified as direct connections. In addition, when clients are busy and respond to server packets with some delay, they are falsely identified as proxies. The method relies on thresholds that need to be determined experimentally. More research is needed to investigate this method in more detail.

MSS fingerprinting

A passive method, using captured network traffic, is to examine the advertised Maximum Segment Size (MSS) in the three-way handshake of incoming TCP connections. Experiments have shown that there is a correlation between identified settings of the OpenVPN proxy software and the advertised MSS value in the three-way TCP handshake. The major omission of this method is that only specific OpenVPN configurations can be detected. An additional complication is the wide range of MSS values random connections can have. Further research is needed to determine the uniqueness of these typical MSS values compared to other traffic.

6.3 FUTURE WORK

This work is the result of an extensive study in detecting proxies from the server-side. Still, there is much work to be done to complete the overview. The largest omission of this work is that its focus is limited to internet-layer proxies. Further studies on the detection of application-layer and link-layer proxies would complete the work.

Furthermore, the methods developed in this work are proof-of-concept methods, and a first step in creating a real detection application. More work is needed to create a more mature detection method that can be tested and validated. In addition, the dataset used for the experiments lacked a good representation of direct connections. More experiments are needed with direct connection IP addresses to substantiate the results.

Because the six identified detection techniques are very different in their approach, the combination of techniques can potentially lead to better detection than the sum of the individual methods. Thus the combination of the results of different methods could be studied further.

It was noted that there is no good classification scheme to classify the type of proxy in respect to the TCP/IP protocol layer. The common classification in link-layer, internet-layer and application-

layer proxy turned out to be inappropriate. This is because various (contradicting) definitions exist, and because some proxy protocols can proxy traffic at different TCP/IP protocol layers while they are labelled as application-layer proxies. Based on this work, a classification scheme can be created in the context of detecting proxies.

7 BIBLIOGRAPHY

Borovicka, Tomas, Marcel Jirina, Pavel Kordik, and Marcel Jiri. 2012. “Selecting Representative Data Sets.” In *Advances in Data Mining Knowledge Discovery and Applications*, edited by Adem Karahoca. InTech. https://doi.org/10.5772/50787.

Buchanan, Mark. 2007. “Statistics: Conviction by Numbers.” News. Nature. January 17, 2007. https://doi.org/10.1038/445254a.

Constantin, Lucian. 2012. “Tools Released at Defcon Can Crack Widely Used PPTP Encryption in under a Day.” Computerworld. July 29, 2012. https://www.computerworld.com/article/2505117/cyberwarfare/tools-released-at-defcon-can-crack-widely-used-pptp-encryption-in-under-a-day.html.

Fielding, Roy, and Julian Reschke. n.d. “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.” Accessed May 27, 2018. https://tools.ietf.org/html/rfc7230.

foo.com, Jasper packet-. 2014. “Determining TCP Initial Round Trip Time.” *Packet-Foo.Com* (blog). July 25, 2014. https://blog.packet-foo.com/2014/07/determining-tcp-initial-round-trip-time/.

“Global Internet Testing | Speedchecker Ltd.” n.d. Accessed September 18, 2018. https://probeapi.speedchecker.xyz/.

Hills, Roy. (2013) 2018. *The IKE Scanner. Contribute to Royhills/Ike-Scan Development by Creating an Account on GitHub*. C. https://github.com/royhills/ike-scan.

“IBM Knowledge Center - Scan Duration and Ports Scanning.” n.d. Accessed September 26, 2018. https://www.ibm.com/support/knowledgecenter/en/SS42VS_7.2.6/com.ibm.qradar.doc/c_qvm_scan_times_ports.html.

“ICMP Tunnel.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=ICMP_tunnel&oldid=846302017.

IP2Location. n.d. “IP2Proxy LITE IP-COUNTRY Database.” IP2Location. Accessed June 3, 2018. https://lite.ip2location.com/database/px1-ip-country.

“IPsec.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=IPsec&oldid=863034235.

Koblas, David. 1992. “SOCKS.” In .

“Layer 2 Tunneling Protocol.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Layer_2_Tunneling_Protocol&oldid=862304783.

“Layer Two Tunneling Protocol.” n.d. Accessed October 11, 2018. http://www.chimica.unipd.it/luigino.feltre/pubblica/unix/winnt_doc/2000/inbe_vpn_njvv.html.

Leech, Marcus. 1996. “SOCKS Protocol Version 5.” March 1996. https://tools.ietf.org/html/rfc1928.

liquidat. (2013) 2018. *Nagios/Icinga Check for OpenVPN Availability Monitoring: Liquidat/Nagios-Icinga-Openvpn*. Python. https://github.com/liquidat/nagios-icinga-openvpn.

“Manuals.” n.d. Accessed September 26, 2018. https://openvpn.net/index.php/manuals.html.

“Meet Algo, the VPN That Works.” 2016. *Trail of Bits Blog* (blog). December 12, 2016. https://blog.trailofbits.com/2016/12/12/meet-algo-the-vpn-that-works/.

“[MS-SSTP]: Relationship to Other Protocols.” n.d. Accessed October 11, 2018. https://msdn.microsoft.com/en-us/library/cc247424.aspx.

“[MS-SSTP]: Secure Socket Tunneling Protocol (SSTP).” n.d. Accessed October 11, 2018. https://msdn.microsoft.com/en-us/library/cc247338.aspx.

Murray, David, Terry Koziniec, Kevin Lee, and Michael Dixon. 2012. “Large MTUs and Internet Performance.” In *2012 IEEE 13th International Conference on High Performance Switching and Routing*, 82–87. Belgrade, Serbia: IEEE. https://doi.org/10.1109/HPSR.2012.6260832.

“OpenVPN.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=OpenVPN&oldid=846543167.

Pannu, Mandeep, Bob Gill, Robert Bird, Kai Yang, and Ben Farrel. 2016. “Exploring Proxy Detection Methodology.” In , 1–6. IEEE. https://doi.org/10.1109/ICCCF.2016.7740438.

Petersson, A., and M. Nilsson. 2014. “Forwarded HTTP Extension.” https://www.rfc-editor.org/info/rfc7239.
“Proxy Anonymity Check - ProxyDB.” n.d. Accessed June 9, 2018. http://proxydb.net/anon.
“Proxy Server.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Proxy_server&oldid=863308888.
R. Fielding, Ed, and Ed J. Reschke. 2014. “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.” https://www.rfc-editor.org/info/rfc7230.
“Reserved IP Addresses.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Reserved_IP_addresses&oldid=841692738.
“Reverse DNS Lookup.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Reverse_DNS_lookup&oldid=855578883.
“Round-Trip Delay Time.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Round-trip_delay_time&oldid=827838151.
Savola, P. 2006. “MTU and Fragmentation Issues with In-the-Network Tunneling.” RFC4459. RFC Editor. https://doi.org/10.17487/rfc4459.
“Secure Socket Tunneling Protocol.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Secure_Socket_Tunneling_Protocol&oldid=843095314.
“SOCKS.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=SOCKS.
Staff, Ars. 2008. “How the ‘Net Works: An Introduction to Peering and Transit.” Ars Technica. February 9, 2008. https://arstechnica.com/features/2008/09/peering-and-transit/.
“Standard Normal Table.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Standard_normal_table&oldid=863582997.
“Stunnel.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Stunnel&oldid=863295380.
Tanenbaum, Andrew S., and D. Wetherall. 2011. *Computer Networks*. 5th ed. Boston: Pearson Prentice Hall.
“Tor (Anonymity Network).” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Tor_(anonymity_network)&oldid=863440129.
“Udger.” n.d. Udger.Com. Accessed September 17, 2018. https://udger.com/resources/datacenter-list.
ValdikSS. 2015. “Detecting VPN (and Its Configuration!) And Proxy Users on the Server Side.” *Medium* (blog). July 25, 2015. https://medium.com/@ValdikSS/detecting-vpn-and-its-configuration-and-proxy-users-on-the-server-side-1bcc59742413.
Webb, A. T., and A. L. N. Reddy. 2016. “Finding Proxy Users at the Service Using Anomaly Detection.” In *2016 IEEE Conference on Communications and Network Security (CNS)*, 82–90. https://doi.org/10.1109/CNS.2016.7860473.
“What Is DNS Tunneling?” 2016. Plixer. September 28, 2016. https://www.plixer.com/blog/network-security-forensics/what-is-dns-tunneling/.
“WHOIS.” 2018. *Wikipedia*. https://en.wikipedia.org/w/index.php?title=WHOIS&oldid=856084792.

Appendix I. HTTP PROXY HEADERS

A list of HTTP header lines added by a http proxy (taken from ("Proxy Anonymity Check - ProxyDB" n.d.))

HTTP-Headers that can reveal your IP-address:

ACCPROXYWS	Cdn-Src-Ip	Client-IP
client_ip	CUDA_CLIIP	Forwarded
Forwarded-For	REMOTE-HOST	X-Client-Ip
X-Coming-From	X-Forwarded	X-Forwarded-For
X-Forwarded-For-IP	X-Forwarded-Host	X-Forwarded-Server
X-Host	X-Network-Info	X-Nokia-RemoteSocket
X-ProxyUser-IP	X-QIHOO-IP	X-Real-IP
XCnool_forwarded_for	XCnool_remote_addr	

HTTP-Headers that can reveal that can reveal that you are using a proxy:

Mt-Proxy-ID	Proxy-agent	Proxy-Connection	Surrogate-Capability
Via	X-Accept-Encoding	X-ARR-LOG-ID	X-Authenticated-User
X-BlueCoat-Via	X-Cache	X-CID-HASH	X-Content-Opt
X-D-Forwarder	X-Fikker	X-Forwarded-Port	X-Forwarded-Proto
X-IMForwards	X-Loop-Control	X-MATO-PARAM	X-NAI-ID
X-Nokia-Gateway-Id	X-Nokia-LocalSocket	X-Original-URL	X-Proxy-ID
X-Roaming	x-teamsite-preremap	X-Tinyproxy	X-TurboPage
X-Varnish	X-Via	X-WAP-Profile	X-WrProxy-ID
X-XFF-0	Xroxy-Connection	And all other headers that begin with X-*	

Appendix II. DEFAULT PORT AND AUTH OPENVPN PROVIDERS

Provider	UDP Port	TCP port	Auth.
AirVPN	443	443	Shared key
blackbox	1194	1194	None
BTGuard	1194	443	None
BulletVPN	1194	443	Shared key
Celo	1194	443	Non-shared key
CyberGhost	443	443	None
ExpressVPN	1195	443	Shared key
HideIPVPN	443, 992, 1194, 2460, 3389, 5005, 5555	443, 992, 1194, 2460, 3389, 5005, 5555	None
HideMe	4000	-	Shared key
HMA	553	443	None
ibVPN	1196	-	None
IPVanish	443	443	None
IVPN	2049	443	Shared key
LimeVPN	1195	-	None
LiquidVPN	53, 118, 1194, 123, 443, 9201	22, 443, 80, 88, 8008	Shared key
Mullvad	1194, 1195, 1196, 1197, 1301, 1302	443	None
NordVPN	1194	443	Mixed[††††]
PerfectPrivacy	149, 151, 150, 1149, 1150, 1151	-	Shared key
PIA	1197, 1198	443	None
PrivateVPN	1194	443	Shared key
proXPN	443	443	None
RA4WVPN	1194	443	Shared key
SaferVPN	1194	443	None
SecureVPN	80, 443	-	Non-shared key
SmartDNSProxy	1194	443	None
tigerVPN	1194	443	None
TorGuard	1912	1912	Shared key
TotalVPN	1194	443	None
VanishedVPN	1194	-	None
VPN.ac	53, 88, 12200, 26000	80, 88, 12200, 26000	Shared key
VPN.ht	1194, 1195, 1196, 1197, 1198, 1199, 1200	443	Shared key

[††††] A shared key is used by 3087 of the 4750 servers. The remaining 1663 servers use a non-shared key.

VPNArea	53, 111, 123, 443, 1194, 1195, 8292	1194	Shared key
VPNSecure	1191	443	None
VPNUnlimited	1194	80	None
VyprVPN	1194	-	None
Windscribe	443	443	Shared key
WiTopia	1194	-	None