TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

GAINING RELIABLE ENDPOINT AWARENESS IN A NETWORK SECURITY SOLUTION

Jenny Heino

TURUN YLIOPISTON JULKAISUJA ANNALES UNIVERSITATIS TURKUENSIS
SARJA SER. F OSA TOM. 53 | TECHNICA INFORMATICA | TURKU 2025

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

GAINING RELIABLE ENDPOINT AWARENESS IN A NETWORK SECURITY SOLUTION

Jenny Heino

TURUN YLIOPISTON JULKAISUJA ANNALES UNIVERSITATIS TURKUENSIS
SARJA SER. F OSA TOM. 53 | TECHNICA INFORMATICA | TURKU 2025

University of Turku

Faculty of Technology
Department of Computing
Information and Communication Technology
Doctoral Programme in Technology

Supervised by

Professor Seppo Virtanen
Department of Computing
University of Turku, Finland

Dr. Antti Hakkala
Department of Computing
University of Turku, Finland

Reviewed by

Professor Olaf Maennel
University of Adelaide, Australia

Professor Dragos Truscan
Åbo Akademi, Finland

Opponent

Professor Kimmo Halunen
University of Oulu, National Defence University, Finland

ISBN 978-952-02-0084-8 (PRINT)
ISBN 978-952-02-0085-5 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)
Painosalama, Turku, Finland, 2025

To all the children I have had the privilege to share my life with.
The future is yours.

UNIVERSITY OF TURKU
Faculty of Technology
Department of Computing
Information and Communication Technology
HEINO, JENNY: Gaining Reliable Endpoint Awareness in a Network Security Solution
Doctoral dissertation, 158 pp.
Doctoral Programme in Technology
March 2025

ABSTRACT

The field of network security has been going through a significant evolution during recent years. Services that used to be run locally by organizations, such as email servers and office solutions, have largely been transformed into cloud services. In addition, the amount of remote work has increased considerably, mostly due to the COVID-19 pandemic which forced network users to become remote almost overnight. The separation of good and bad network traffic has become increasingly difficult, and the appearance of false positive and false negative security events is unacceptably frequent. Network security solutions are forced to produce innovative approaches for providing reliable protection for their users.

This thesis focuses on the concept of improving the traffic inspection process of a network security solution with endpoint awareness. There are three main contributions in this thesis. The first contribution is in providing a comprehensive understanding of how a network security solution can gain endpoint awareness. A patent is included in the thesis, introducing a novel, concrete way of gaining further awareness of the endpoint based on the information stored in the extensions included in the handshake process of an encrypted TLS connection. This method has already been implemented into the Forcepoint Network Security Platform and has proven to be a valuable addition to the product. In addition, a study is performed on existing methods of gaining endpoint awareness where both active and passive methods are examined, as well as the state-of-the-art in different network security solutions. The second contribution is in introducing well rationalized improvements for the existing hash fingerprinting algorithms. An update is proposed for these algorithms where the pre-hash string is used as the fingerprint instead of taking the final hash value. Experiments are performed using machine learning on the pre-hash strings for endpoint awareness, showing promising results. The third contribution is in defining two concrete methodologies for implementing endpoint awareness into a network security solution. The efficacy of the second methodology, entitled JAPPI, is evaluated in a larger-scale experiment. The model performed exceptionally well, with 99.5% coverage, demonstrating that it provides an excellent means for introducing endpoint awareness into the inspection process of a network security solution.

KEYWORDS: Network security, Intrusion prevention, Traffic identification, Machine learning

TURUN YLIOPISTO
Teknillinen tiedekunta
Tietotekniikan laitos
Tietotekniikka
HEINO, JENNY: Gaining Reliable Endpoint Awareness in a Network Security Solution
Väitöskirja, 158 s.
Teknologian tohtoriohjelma
Maaliskuu 2025

TIIVISTELMÄ

Verkkotietoturva-alan kehitys on ollut lähivuosina merkittävää. Aiemmin paikallisina asennuksina toteutetut palvelut, kuten sähköpostipalvelimet ja toimisto-ohjelmistot, ovat pääosin siirtyneet pilvipalveluiksi. Etätyön määrä on kasvanut merkittävästi COVID-19-pandemian myötä, joka pakotti työntekijät etätyöhön lähes yhdessä yössä. Toivotun ja ei-toivotun verkkoliikenteen erottamisesta on tullut yhä vaikeampaa, ja väärien positiivisen ja väärien negatiivisten tunnistusten suurta määrää ei voida hyväksyä. On välttämätöntä keksiä uusia ja innovatiivisia lähestymistapoja luotettavan tietoturvan tarjoamiseksi verkkotietoturvatuotteiden käyttäjille.

Tämä väitöskirjatutkimus keskittyy liikenteen tarkastelun tehostamiseen lisäämällä verkkotietoturvalaitteeseen tietoisuus päätelaitteistosta. Tutkimuksella on kolme päätulosta. Ensimmäisenä tuloksena esitellään kokonaisvaltainen ymmärrys siitä, miten päätelaitteistotietoisuus voidaan saavuttaa verkkotietoturvalaitteessa. Väitöskirja esittelee patentoidun, uudenlaisen ja selkeän keinon saada lisätietoja päätelaitteistosta hyödyntäen TLS-protokollalla salatun verkkoliikenteen kättelyvaiheeseen sisällytettyjä tarkennuksia. Tämä tekniikka on jo toteutettu Forcepoint Network Security Platform- ohjelmistoon, jossa sen on todettu tuovan merkittävää lisäarvoa. Väitöskirjassa tutkitaan olemassa olevia keinoja päätelaitteistotietoisuuden saavuttamiseksi, käydään läpi sekä aktiivisia että passiivisia tekniikoita ja tutustutaan viimeisintä tekniikkaa edustavissa verkkotietoturvalaitteissa käytössä oleviin toteutuksiin. Toisena tuloksena esitetään parannuksia olemassa oleviin hajautustunnistealgoritmeihin. Keskeinen parannusehdotus on esihajautusmerkkijonon (engl. prehash string) käyttö tunnisteena lopullisen hajautustunnisteen sijaan. Koneoppimisen soveltamisesta esihajautusmerkkijonoon päätelaitteistotietoisuuden saavuttamiseksi saadaan lupaavia tuloksia. Kolmantena tuloksena määritellään kaksi konkreettista metodologiaa päätelaitteistotietoisuuden toteuttamiseksi verkkotietoturvalaitteeseen. Näistä jälkimmäinen JAPPI-metodologia suoriutui laajamittaisessa kokeessa erinomaisesti saavuttaen 99,5% kattavuuden. Tämä osoittaa sen tarjoavan oivallisen keinon tuoda päätelaitteistotietoisuus verkkotietoturvalaitteen liikenteentarkasteluprosessin.

ASIASANAT: Verkkotietoturva, Hyökkäyksen esto, Liikenteen tunnistaminen, Koneoppiminen

Acknowledgements

My deepest gratitude goes to my supervisors Professor Seppo Virtanen and Dr. Antti Hakkala. The quality of your guidance has always been first-class, and I have heartfelt appreciation for the way you found time for me in your busy schedule even during the evening or weekend if the time pressure was high. It was an absolute privilege and pleasure to collaborate with you.

I am also grateful to Professor Olaf Maennel from University of Adelaide and Professor Dragos Truscan from Åbo Akademi for taking the time to review my thesis and for providing such positive and encouraging comments.

An indispensable contributor in getting this thesis where it is now is my co-author, collaborator, mentor and life companion Christian Jalio. Your profound knowledge of and interest in network security has been an invaluable inspiration throughout and beyond this project.

I would also like to express my gratitude to my other co-workers at the network security side at Forcepoint. Being able to work every day with such brilliant and dedicated people is a continuous source of inspiration. Without you this dissertation would have remained a distant dream.

Finally, I would not be here, writing the final words for my doctoral dissertation, without the lifelong support of my friends and family. My friends have always been there for me, giving me courage and sometimes the needed nudge to reach toward the next ladder in life. And, above all, my family has always given their wholehearted support for everything I do, without any pressure but with the continuous belief that anything is possible.

March 6, 2025
Jenny Heino

Table of Contents

Abbreviations

ALPN	Application Layer Protocol Negotiation
API	Application Programming Interface
CASB	Cloud Access Security Broker
CMDB	Configuration Management Database
DLP	Data Loss Prevention
DNS	Domain Name System
DPI	Deep Packet Inspection
ECA	Endpoint Context Agent
ECH	Encrypted Client Hello
EIA	Endpoint Intelligence Agent
EVE	Encrypted Visibility Engine
FTP	File Transfer Protocol
FW	Firewall
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IDS	Instrusion Detection System
IOC	Indicators Of Compromise
IPS	Intrusion Prevention System
LDA	Latent Dirichlet Allocation
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NGFW	Next Generation Firewall
NPN	Next Protocol Negotiation
OSI	Open Systems Interconnection
RBF	Radial Basis Function
SASE	Secure Access Service Edge
SDN	Software-Defined Networking
SD-WAN	Software-Defined Wide-Area Network
SMB	Server Message Block
SNI	Server Name Indication
SSE	Secure Service Edge
SVM	Support Vector Machine
SWG	Secure Web Gateway

TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAN	Wide-Area Network
ZTNA	Zero Trust Network Access

List of Original Publications

This dissertation is based on the following original publications, which are referred to in the text by their Roman numerals:

I Jenny Heino, Tuomo Syvänne, Christian Jalio and Olli-Pekka Niemi. Identification of unknown traffic based on transport layer security extensions. US Patent US11418542B2, 2022.

II Jenny Heino, Antti Hakkala and Seppo Virtanen. Study of methods for endpoint aware inspection in a Next Generation Firewall. Cybersecurity, Volume 5, Article no 25, 2022.

III Jenny Heino, Christian Jalio, Antti Hakkala and Seppo Virtanen. A Method for Endpoint Aware Inspection in a Network Security Solution. IEEE Access, Volume 10, pp. 44517–44530, 2022.

IV Jenny Heino, Ayush Gupta, Antti Hakkala and Seppo Virtanen. On Usability of Hash Fingerprinting for Endpoint Application Identification. In proceedings of 2022 IEEE International Conference on Cyber Security and Resilience (CSR), pp. 38–43, Virtual Conference July 27-29, 2022.

V Jenny Heino, Antti Hakkala and Seppo Virtanen. Categorizing TLS traffic based on JA3 pre-hash values. Procedia Computer Science, Issue 220, pp. 94–101, 2023.

VI Jenny Heino, Christian Jalio, Antti Hakkala and Seppo Virtanen. JAPPI: An unsupervised endpoint application identification methodology for improved Zero Trust models, risk score calculations and threat detection. Computer Networks, Volume 250, Article no 110606, 2024.

1 Introduction

A network connection between two endpoints is established when two devices, such as personal computers, server devices, smartphones or even home appliances, are connected to the same network, and one of them decides to communicate with the other. The endpoint initiating the network connection, referred to as the *client*, typically either wants to receive information that the other endpoint, the *server*, has, or wishes to report some information to the other endpoint. A simple example is a personal computer used for browsing the web: the personal computer works as a client and establishes a network connection with a remote server which hosts the contents for a particular website.

More specifically, the network connection is initiated by, and on the other side received by, a specific *endpoint application*. An endpoint application is a software component installed on the endpoint. There are numerous different types of endpoint applications. Some are used actively by the user of the endpoint. Such endpoint applications include, for example, web browsers used for browsing web sites, email applications used for sending and receiving email, and chat applications. Other endpoint applications work in the background and may even be a part of the operating system itself. These background applications are often responsible for actions such as looking for updates, performing other timed tasks, and sending telemetry about the device. A simplistic visualization showing a personal computer as an endpoint, a web browser as an endpoint application and a network connection accessing the web

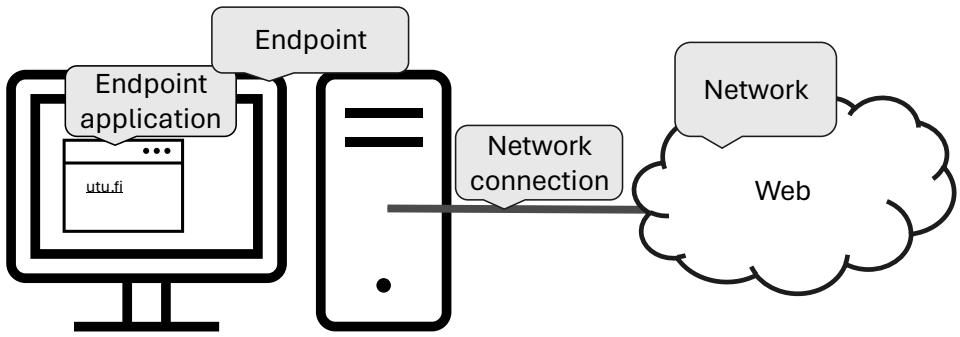

Figure 1. A simplified visualization of the concepts of endpoint, endpoint application and a network connection.

is shown in figure 1.

Information security is typically implemented in a way that separates endpoint security and network level security from each other. An endpoint can have anti-malware software running along with a local firewall which controls what traffic can enter and leave the device. A network security solution resides between the endpoint and external resources, where it sees the network traffic leaving and entering the endpoints. It can detect and stop network level attacks before they enter the endpoint, control where and what kind of traffic is permitted, and form a big picture about the protected network. It is generally an accepted fact that these two remain separate with no awareness of the other. Their functionality is based on the fact that they work blindly in their own context.

This approach has its advantages. It offers an onion like security structure. If one layer does not function properly, the other layers continue to work without interruption. If, for example, a laptop gets compromised when it is taken out of the office into an unsafe public network, a network security solution can still identify the traffic initiated by the installed malware and alert the administrator. Alternatively, if the network security solution is unable to identify a new type of an attack, an endpoint where all applications are up to date with the latest security patches may be safe from the attack which targets an older, vulnerable version of an application.

Despite this, the detachment of these security solutions makes the functionality of a singular security solution weaker. Without the additional context received from the other layer, the overall image of an event is left incomplete. Due to this, one security solution can never provide maximal security.

Without any awareness of the endpoints of a network connection, a network security solution must make security decisions in a generic way that applies to all traffic. In many instances this approach works well enough. When a virus is being downloaded, it often does not matter what the source or the target is - the download should be blocked. Similarly, if an untrusted entity tries to access a restricted resource, it does not matter what endpoint application was used for initiating the connection - it should not be let through no matter what. But there are cases when a better awareness of the endpoints can help a network security solution to make a more informed decision.

The purpose of a network security solution is to identify and deny malicious network activities while letting all normal network activities proceed undisturbed. To achieve this, a network security solution typically has a set of pre-determined detection logic patterns, or *fingerprints*, configured. These fingerprints can vary greatly depending on the network security solution and the traffic being processed. They can be something as simple as a string match, or a complex combination of patterns ranging over several network connections. As the network security solution processes network traffic, it applies these fingerprints on the traffic, and if the traffic matches to a fingerprint, a *security event* is produced.

In many cases, however, it is not trivial to see if a pattern in a network connection is benign or if it is intended for malicious purposes. An attacker does usually not leave clear indicators in the traffic of its malicious intent - they will instead try to make the traffic look as innocent as possible. A network security solution often needs to balance between two negatives: accidentally terminating an innocent network connection, which causes disturbances to benign traffic, or letting something malicious potentially go through undetected.

A security event which is produced from network traffic is considered a *false positive* event if the traffic is not what the original fingerprint was intended to catch. A typical example of this is when a network security device observes a traffic pattern which it assumes to be an attempted attack and blocks the connection, when in fact the traffic was not malicious. False positive events are usually caused by the fact that some attacks are difficult or even impossible to identify reliably, and the fingerprints used by the network security device are thus left incomplete.

A *false negative* event happens when something that should have been identified and blocked is, instead, let through. An example of a false negative event is when a network security solution has fingerprints for a specific type of an attack, but the attack traffic differs from these fingerprints enough to get past the network security solution. Similarly as with a false positive events, a false negative event is typically caused by the fact that a fingerprint which reliably detects all attack variants is often difficult, or sometimes impossible, to create.

There are cases where an attack is very simple, and so is creating a fingerprint for it. In such cases false positive and false negative events are practically nonexistent. But many attack types are not as clear. An example is when a vulnerability in an endpoint application is triggered by specific URL parameters, for example a negative value for a specific parameter. Identification of an attack seems easy - block any requests where this specific parameter has a negative value. But a URL parameter with the same name might be used by other endpoint applications as well, and negative values might be normal for them. Thus, blocking all negative values for this parameter could cause false positives when the traffic is not targeted against the vulnerable endpoint application.

Some attacks can also be more complex, requiring multiple steps to succeed, and a fingerprint intended for identifying such an attack can be complicated and require that several different detection patterns are matched from the traffic for a security event to be produced. Sometimes this can lead to a situation where a benign connection happens to contain patterns that match to the fingerprint without it being an indication of an attack, and a false positive security event is produced. And vice versa, sometimes the fingerprint can be lacking and miss some crucial steps of the attack, thus failing to produce a security event when it sees an actual attack.

One approach to reducing false positive security events and improving the accuracy of a network security solution is to gain additional metadata about the connec-

tion. This can include methods such as verifying the reputation of the destination IP address, or with some network protocols, the reputation of the URL. If the destination is known to be a trustworthy server, a network security solution can make the decision not to produce security events that are considered less accurate. Similarly, if the destination is considered unknown or suspicious, a network security solution can decide to take a stricter approach and produce even the less reliable security events. In case of an encrypted connection, a network security solution can assess the signer of the security certificate. If the signer is considered trustworthy, the network security solution can again decide to leave the less accurate security events out, while security events produced from a connection which has been signed by an untrusted signer can be produced even when they are considered less accurate.

An interesting piece of additional metadata that a network security solution can benefit from when considering whether to produce a less accurate security event or not is metadata about the endpoints of the connection. When a network security solution observes an anomaly in a network connection where it is unsure if the connection is an attack or not, this kind of additional metadata about the endpoint may provide the needed step for deciding if the connection should be discarded or not. As an example, a network security solution may observe a network connection which demonstrates patterns based on which it might be an attack against Internet Explorer 8, an old version of a web browser only implemented for the Windows operating system. If the protected endpoint is known to be a device with the macOS operating system, it is unlikely that it will be in any danger. In this case the network security solution may decide to let the traffic pass. If the protected endpoint was known to have an out-of-date Windows 7 operating system, the network security solution may decide to terminate the connection.

But what about when the potential vulnerability is in an application that could be installed on the operating system that the protected endpoint has, but it is not known to the network security solution whether it has been installed there or not? If the potential attack would be against an old version of the Chrome browser but the protected endpoint does not have any version of Chrome installed, there is no point for the network security solution to produce less reliable security events against attacks on Chrome. But if it does have the Chrome browser installed, and the installed version even happens to be vulnerable against the attack, the network security solution should raise its alert level and even less reliable security events for attacks against Chrome should be produced.

One solution for this would be for the network security solution to have awareness of the exact endpoint applications of a network connection. In the above example, the network security solution would not have to guess whether the protected endpoint might have the vulnerable endpoint application installed or not. It can instead see which endpoint application is in the receiving end of the traffic and base its decision on that information, letting the connection through if the receiving applica-

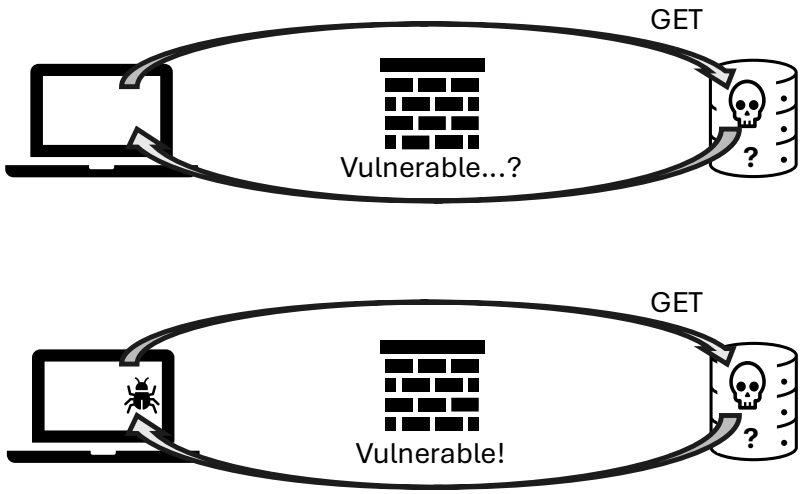

Figure 2. A simplified visualization demonstrating two network security use cases. In both use cases an endpoint application accesses a remote server through a network security solution, and the server responds with a potentially malicious response. In the first use case the network security solution does not have endpoint awareness, but in the second one it does. In the second use case the network security solution can make the decision to terminate the connection with high confidence, due to being able to verify whether the receiving endpoint application is vulnerable against the potential attack or not.

tion is something completely different, or producing a security event and terminating the connection if it happens to be the vulnerable endpoint application. Figure 2 shows a simplistic visualization of two use cases. An endpoint application is accessing a remote server over a network connection where the first network security solution does not know if the endpoint application is vulnerable for the potential attack, whereas the second network security solution has endpoint awareness and knows that the endpoint application would be vulnerable against the potential attack. The server responds with a reply which matches to a fingerprint in the network security solution indicating a potential attack. In the first use case the network security solution needs to make its decision without further information and to potentially create a false positive security event or let an actual attack through. In the second use case the network security solution has more confidence to make the decision whether to terminate the connection or not as it can verify if the receiving endpoint application is vulnerable to the potential attack.

This level of endpoint awareness brings many other advantages for a network security solution as well. For one, it offers the ability to enforce access policies based on the endpoint application. This means that it is possible to determine which endpoint applications are permitted to access which resources, and anything not specifically permitted can be dropped. One benefit from this level of granularity can be seen when considering a device, such as a work laptop, that can be carried away

from the protected network by the user and then brought back infected with a malicious software component. When the access policy has been configured to only permit network connections from a specific set of trusted endpoint applications, the malicious software component is unable to spread to the rest of the network or do any other damage inside the protected network. An alert about the attempts can be sent to the network administrator who will then be able to confiscate the infected device.

Another advantage can be observed when the network security solution is used for maintaining a Zero Trust architecture. In a typical Zero Trust architecture the identity of the user is verified, after which the device will be given access to suitable resources. But what if, after the authentication, the restricted resource is also being accessed by an unknown endpoint application on the same device? Something malicious might have found its way on the endpoint, and once the proper access rights have been confirmed it also has access to the restricted resource. If the network security solution has endpoint awareness as described above, it can also restrict access to restricted resources based on the endpoint application. For example, a database containing personal information about employees should only be accessed by HR personnel using the proper software component. Anything else trying to access the resource should be dropped, and an alert should be sent to the network administrator about the event.

Awareness of the endpoint can on a high level be gained through two means. The first is using an active method. An active method indicates that some level of interaction is required between the network security solution and the protected endpoint to receive information about the endpoint. This might include active scanning of the endpoint, maintaining a Configuration Management Database or *CMDB* where information is stored about the endpoint, or even installing an external component on the endpoint which reports information about the endpoint directly to the network security solution. Active methods can provide the most accurate picture about the endpoint for a network security solution, especially if the endpoint reports very granular information, such as the endpoint application initiating each new network connection. But they also require that the endpoint is cooperative or maintained by the administrators of the network. In many protected networks this level of control on every protected device is not achievable.

The second means of gaining endpoint awareness is through passive methods. A passive method is something that can be performed without any interaction with the protected endpoint, only by observing network traffic. Passive methods can be less reliable than active methods, as the patterns in the network traffic may be dynamic, or they might change after an update to an endpoint application. They are, however, more useful in a general sense, as they can be applied on any network traffic, not only on the endpoints that can be controlled by the administrator.

With passive methods a question of privacy often surfaces. If a network security solution is able for example to deduce the endpoint applications that have been

installed on an endpoint purely based on the network traffic, sometimes even as precisely as identifying the exact versions of the applications, then this same ability is available to any other entity who is able to observe the network traffic as well. It could be a malicious actor who is looking for targets with a vulnerable version of a particular endpoint application, or it could be a software publisher looking for targets to advertise their competing endpoint application for. In any case, the privacy of the endpoint, and the privacy of the user, is breached.

There are ways in which protocol and software designers are trying to battle against this issue. An extension, called Encrypted Client Hello or *ECH*, has been proposed for the TLS protocol [1]. The purpose of this extension is to encrypt sensitive fields present in the Client Hello message, such as the Server Name Indication extension or *SNI*, or the Application Layer Protocol Negotiation list or *ALPN*. In addition, the Google Chrome web browser has introduced a feature to randomize the order in which the extensions in a Client Hello message are presented [2], which renders some passive fingerprinting mechanisms useless.

While these improvements enhance the privacy of the user, they hinder the work of a network security solution attempting to provide better security for an endpoint. This aspect of these improvements is rarely if ever mentioned when justifying them. Nevertheless, a network security solution should not prevent enhancements that increase privacy. Instead, a network security solution could be at a unique spot to increase user privacy when considering passive methods. It would be possible for a network security solution to utilise the information gained from passive methods to increase security but obfuscate that same information when sending the traffic forward. This would increase the privacy of the user while maintaining the ability to provide more accurate security features.

The research presented in this thesis introduces a novel method of gaining endpoint awareness in a network security solution. To achieve this, various approaches are explored. As part of the research, a study is performed to assess existing methods and existing implementations in network security solutions, based on which an initial methodology for integrating endpoint awareness into a network security solution is proposed. After identifying certain deficiencies in the methods used for gaining endpoint awareness, further research is conducted. Based on the new findings made, a new methodology, named JAPPI, is introduced. The efficacy of JAPPI is evaluated in two large networks where it is found to be highly efficient. The results are also compared to ones from our previous methodology, and the improvements introduced in JAPPI are immense.

1.1 Motivation and Objectives

The core of the research, gaining an understanding or *awareness* of the protected endpoint in a network security solution, rises from a concrete real-life need. Each

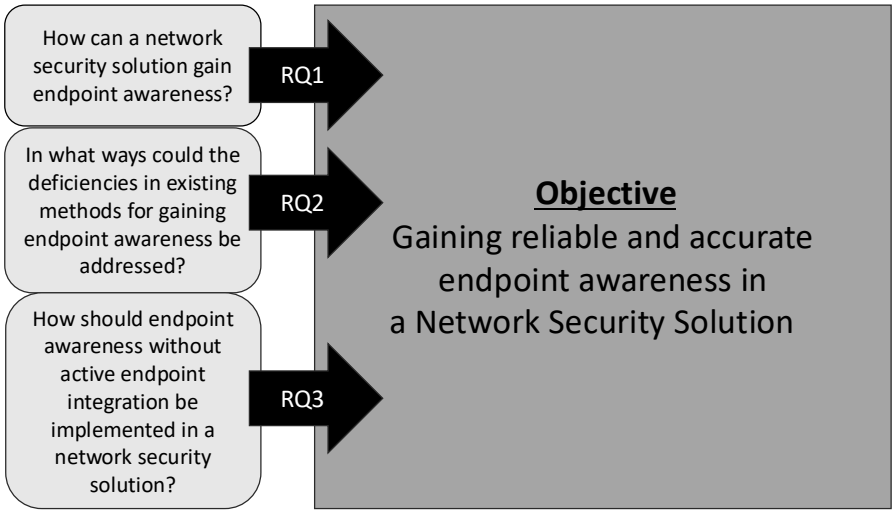

Figure 3. Structure of the research in this dissertation

false positive security event which hinders normal network flow brings a negative user experience to anyone behind a network security solution or responsible for maintaining one. Even worse are false negative security events that let something malicious through. The consequences of a false negative security event range from mild inconvenience, such as from an adware, to the compromise of the entire network. Minimizing the risk for each false positive or false negative security event is a core purpose for a network security solution and any new approach can be extremely valuable. Awareness of the protected endpoint, especially the endpoint application, can provide a network security solution with crucial metadata which helps in making the correct inspection decision for each network connection.

The objective of this dissertation was twofold in essence: firstly, to identify reliable and accurate methods for a network security solution of gaining awareness of the endpoint application initiating a given network connection, and secondly, to find optimal ways of integrating the suitable methods into the traffic inspection process of a network security solution. The emphasis was on identification methods that could be used purely on network traffic, without a requirement for endpoint integration or active interaction on all protected endpoints.

1.2 Research Questions

To approach the objectives of our research, the following 3 research questions were formulated. Figure 3 visualizes the structure of the research.

RQ1 How can a network security solution gain endpoint awareness?

This research question lays groundwork for all of the research presented in this

thesis. To understand what the best methods for bringing endpoint awareness into the inspection process of a network security solution are, we need to understand what methods exist. To find an answer, a thorough study is needed on existing research and implementations in current network security solutions.

RQ2 In what ways could the deficiencies in existing methods for gaining endpoint awareness be addressed?

This research question focuses on defining and designing new and improved techniques for gaining endpoint awareness with a review and analysis of existing methods as the starting point. This requires the identification of the most advanced existing methods and the recognition of their features and functionalities that could act as a basis for creating new methods in gaining endpoint awareness. An important goal in the review and analysis is the identification of techniques where the application of machine learning could be introduced with an expectation of greater reliability and accuracy.

RQ3 How should endpoint awareness without active endpoint integration be implemented in a network security solution?

The purpose of the third question is to provide a concrete proposal of how endpoint awareness should be implemented in a network security solution. After a suitable method of gaining endpoint awareness is identified, we expect to be able to formulate a methodology to describe the necessary steps for bringing endpoint awareness into a network security solution and to validate its efficacy in a real life network environment.

1.3 Structure of the Thesis

This dissertation consists of two parts. The first part provides an introduction to the research and the concepts the original research papers focus on. The second part consists of the original research papers themselves.

The introductory part consists of the following Chapters:

Chapter 1 provides a high-level summary of the dissertation and introduces the research questions that the dissertation provides answers for.

Chapter 2 gives an overview of existing research on the main concepts of the dissertation.

Chapter 3 introduces the concept of network security solutions, starting from a basic Firewall and ending with the more recent concepts of Zero Trust Network Access or *ZTNA*, and Secure Access Service Edge or *SASE*.

Chapter 4 describes the concept of traffic identification from the perspective of a network security solution, with focus on receiving information about the endpoint, specifically the endpoint applications.

Chapter 5 introduces the original publications and lists their contributions to the dissertation as well as the future work the publication introduces.

Chapter 6 provides a conclusion of the dissertation, presenting the results in a clear manner through the research questions, comparing the research presented in this thesis to other related research, listing the limitations of the dissertation and considering the future work that the dissertation introduces.

2 Related Research

This chapter highlights relevant existing research on bringing endpoint awareness into a network security solution. First, in section 2.1 we go through existing research on ways for identifying endpoint applications based on network traffic. Next, in section 2.2 we introduce two relevant research papers where machine learning is applied for endpoint application identification from network traffic. Finally, in section 2.3 we briefly go through existing information about integrating endpoint awareness into a network security solution.

2.1 Identifying the Endpoint Application Based on Network Traffic

A considerable amount of research has been conducted on various methods of classifying network traffic. In this section we focus on the research which has explored identifying the endpoint applications based on network traffic. The focus of the existing research has been on encrypted TLS traffic, which has also been the main interest of this dissertation due to its current popularity on the internet. The focus on encrypted TLS is explicable: the other common web protocol, plain text HTTP, contains a request header field called User-Agent, which reports the endpoint application initiating the network connection and can thus directly be used as an identification method [3]. It should still be noted that many web browsers have implemented a feature for intentionally spoofing the User-Agent value [4; 5] and thus it depends on the use case how this information should be utilised by a network security solution. The User-Agent field of the HTTP protocol is explored in more detail in section 4.3.1.

Husák et al. [6] explored how well the list of supported cipher suites presented in a TLS Client Hello message correlate with the HTTP User-Agent field. By first collecting samples and then using them to identify other traffic, the authors were able to identify the endpoint application for 84% of the monitored traffic. In a similar fashion, Muehlstein et al. [7] explored identifying the web browser from HTTPS traffic by collecting a set of features from a TLS connection. In addition to the list of supported cipher suites, they collected other information from the connection, such as the compression methods and the number of extensions, but also more general information about the network connection such as amount of network packets and bytes. Using machine learning on these values, they achieved an identification accuracy of

96.06%.

Some research focuses specifically on identifying mobile applications based on network traffic. Protecting mobile devices can be an important function for many network security solutions, even though the focus usually is on local workstations, laptops, and server machines. Taylor et al. [8] developed a tool called AppScanner for identifying traffic from mobile applications. They used UI fuzzing to collect different traffic samples from different mobile applications and automatically created fingerprints based on the collected information. AppScanner does not, however, look into the application layer of the traffic, but focuses on more generic information such as traffic bursts to the same destination IP address. They explore several different classification methods and present performance numbers for them. The same researchers later expanded on their previous research by reviewing how well AppScanner works between different application versions and with the passage of time [9]. In this research, they also experimented with excluding traffic common between different applications in post-processing of the collected data. They discovered that without post processing, the maximal detection accuracy they were able to reach was 40.9% when the versions remained the same during the monitoring period. After the common traffic was excluded during post processing, the accuracy increased to 96% in the best-case scenario.

The most relevant research regarding traffic identification identified during the dissertation was the concept of hash fingerprinting algorithms. These algorithms are explored in more detail in chapter 4 due to their importance to this dissertation. We will explain here the history of these algorithms and related research.

The concept is based on TLS fingerprinting research by Brotherston [10]. He demonstrated that it was possible to use the information presented in the TLS Client Hello to identify a client application. Inspired by this presentation, Althouse et al. [11] created and published an algorithm for fingerprinting TLS Clients called JA3. JA3 was the first hash fingerprinting algorithm, but it was followed by several others for other protocols: HASSH for fingerprinting SSH [12], GQUIC for fingerprinting the original Google-QUIC [13], RDFP for fingerprinting RDP [14] and SMBFP for fingerprinting SMB [15]. The concept in each algorithm is the same: they pick up a set of relevant and unique parameters that are included in the initial handshake message of the protocol and present them in a structured string format, concatenating values given for the same parameter with a dash, and separating values given for different parameters with a comma. Finally, an MD5 hash value is calculated from this string, which comprises the final fingerprint. Since their publication, many of the hash fingerprinting algorithms, especially the JA3 algorithm, have been implemented in several network security solutions and the information collected by these fingerprints is considered a valuable addition to any threat intelligence information shared among security researchers about a novel or potent attack.

2.2 Using Machine Learning for Endpoint Application Identification

Various research articles have been published using different machine learning methods for categorizing network traffic. In the context of this dissertation, the most relevant research was considered to be the publications that used machine learning on the various values collected from a TLS connection. The hash fingerprinting algorithms provide a structured and well-defined method of collecting such information about a network connection for multiple network protocols, and thus they provide a fruitful ground for expanding this kind of research also outside of the context of TLS.

As previously mentioned, Muehlstein et al. [7] used machine learning on various information collected from TLS connections. The researchers used Support Vector Machines (SVM) and Radial Basis Function (RBF) in their research. Since their method only worked when the traffic was being post-processed after the connection had already been closed, the research was not considered as relevant for this dissertation due to the focus here being more on providing in-line security in a network security solution.

In the course of the research presented in this thesis the pre-hash strings of the hash fingerprinting algorithms became a primary focus. While looking for ways of calculating the distance between two different pre-hash strings, the Levenshtein distance algorithm was identified as an intriguing method. At this stage, the existing research was reviewed to see if similar methods had previously been used for the same purpose. Two interesting publications were identified.

In the first, Frolov and Wustrow [16] used Levenshtein distance when performing clustering on TLS traffic. They collected similar information from the TLS Client Hello message as the JA3 algorithm does, but the information was not exactly the same. Their results were very promising, and they were for example able to discover two loosely connected clusters that turned out to be connections from the Google Chrome browser where one cluster contained connections using TLS versions 1.2 and older and the other contained connections from then experimental draft of TLS 1.3.

Later during the same year, Anderson and McGrew [17] published a similar analysis. Their research also collected similar, but not exactly the same, values from the TLS Client Hello as the JA3 algorithm does. By using the Levenshtein distance algorithm to see if a previously unknown fingerprint was considered to be *close* to a known fingerprint, the researchers discovered that nearly all observed TLS connections were covered after a year of data collection.

2.3 Integrating Endpoint Application Awareness into the Inspection Process of a Network Security Solution

In relation to the research objective to identify how endpoint awareness should be introduced to the inspection process of a network security solution, no existing research publications were identified that could have provided a basis for the research. In addition, no other similar problem was identified that would have already been solved, and that could thus have been referred to during the dissertation.

During the research process, one network security solution was identified that had implemented a method for identifying endpoint applications using machine learning. The Cisco Secure Firewall has implemented a proprietary method, called *Encrypted Visibility Engine* or *EVE*, for identifying endpoint applications from TLS traffic [18]. Due to the proprietary nature of the feature, its exact functionality is not known, but based on the public information it seems to utilise similar methods machine learning methods as the publications referenced in the previous section. It does not seem, however, like the endpoint application information would have been integrated into the inspection process of the Cisco Secure Firewall.

3 Network Security Solutions

Network security encompasses a wide variety of different security solutions. A network security solution is typically something that is intentionally separate from the protected endpoints. It resides somewhere on the route from a client to a server, observing the communications between the two endpoints. Traditionally, a network security solution used to be installed on the premises of an office or a data center, where it stood between the protected internal network and the hostile external network, the internet. But as the various assets of an organization started moving to the cloud and the users started to work from remote offices, the network security solutions have also started to move more towards cloud-based instances [19].

As a comparison, an endpoint security solution is something that is installed on a protected endpoint. As with network security, there are various different endpoint security solutions. The most common one is anti-malware software, which can monitor the disk for any existing or incoming files that are considered malicious. A more recent addition to the endpoint security landscape are Endpoint Detection and Response or *EDR* solutions, that are able to identify more complex attacks, including zero-day attacks and fileless malware [20; 21; 22].

In addition, some security solutions may take a more hybrid form. This type of security solution typically performs a limited set of network security features locally on the endpoint but can also redirect the traffic to a remote network security solution. Many Secure Web Gateway products, or *SWG*s, function this way [23; 24].

The most complete security system can be achieved by utilising multiple security layers. Such an architecture can better tolerate a case where one layer fails to function properly - the remaining layers can still detect and prevent an attack, or any other unwanted event. Still, strictly separating endpoint and network security from each other provides weaker overall security, as neither is able to form a perfect picture of the general situation.

This chapter provides a high-level introduction to various types of network security solutions. The concept of a basic firewall is presented in section 3.1. Section 3.2 describes how deep packet inspection brought the evolution from a firewall into next-generation firewalls. Other types of network security solutions are reviewed in section 3.3. The concept of Zero Trust is explored in section 3.4. Finally, we describe the concept of SASE and SSE in section 3.5.

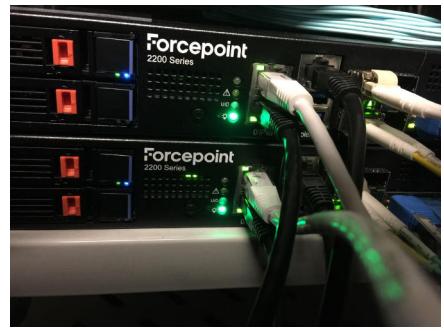

Figure 4. On the left: Electrical Substation Firewall installed between substation transformers. (Credit: Jackson Bishop. This image was marked with a CC BY-SA 3.0 license (https://creativecommons.org/licenses/by-sa/3.0/deed.en)) . On the right: A network firewall cluster with cables attached, connecting the device to internal and external networks. (Credit: Kimmo Talvitie.)

3.1 Firewalls

A firewall is a network security component located at the boundary of a protected internal network and an external network. All traffic passing between these two networks must go through the firewall, and the firewall needs to be able to configure rules that permit some traffic while dropping other traffic. The collection of rules is referred to as a firewall policy [25].

The term *Firewall* has historically been used for many purposes. Before it was used in a computer setting, it was used for describing a physical structure that prevents fire from spreading from one area to another. A wall separating a room with a high risk of a fire from other rooms has been referred to as a firewall already in 1774 [26]. So has a metal layer typically separating a combustion engine from the passenger area. Figure 4 provides a visual comparison between a physical firewall separating two electrical substation transformers and a network firewall separating internal and external networks.

The first time the term was used in a computer security setting was most likely already in 1974 [27]. It was, however, not used to reference a network level software component, but instead as a figure of speech to describe something that prevents software bugs from being exploited. There have been claims that the term was first used to reference a network security solution in the movie WarGames, released in 1983, and this claim has been established as a fact in some sources [28]. This claim is, however, difficult to verify. It could be true, since the first academic references to a network firewall in the sense that we use it today seem to date to sometime around 1992. The initial references seem to, again, be figurative, referencing router configurations as a type of a firewall [29; 30]. But the term quickly establishes itself to denote a separate network security solution and is referenced to as such for example

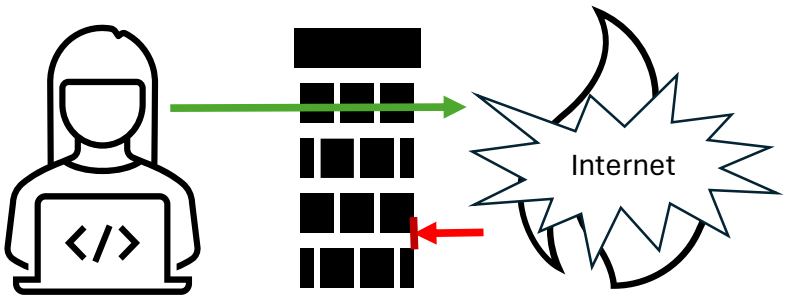

Figure 5. A network firewall protecting a user from attacks originating from the internet.

in RFC 1636, published in 1994 [31]. The concept of a network firewall existed already before the term was used to describe it. The first academic paper presenting the concept of such a packet filter was published in 1987 [32] and the first commercial firewall product was DEC SEAL, released in 1991 [25]. Figure 5 provides a simplified visualization of a network firewall protecting a user in the protected internal network from attacks originating from the internet.

These types of firewalls are able to perform actions only using simple rules. They can permit and discard traffic based on information available on the network and transport layer protocols, but that is usually the limit. They are sometimes referred to as packet filter firewalls or first-generation firewalls [33]. During the early days of the internet these types of network security solutions provided enough protection. But with the inevitable growth of the internet the concept of a first generation firewall became too simple, and further security measures were needed. Especially the lack of visibility into the application layer became a considerable shortcoming, paving the way for more complex network security solutions.

3.2 Next-Generation Firewalls

The concept of a firewall experienced an evolution when Palo Alto introduced the first next-generation firewall, or *NGFW*, in 2010. Initially their strategy was to highlight the application control capabilities of their new network security solution, which enabled them to sell the product to customers who already had a firewall and found it too large of an investment to replace it altogether with a new solution. This gave them a market share without the requirement of complex network refactoring for the customers [34]. Since then, the next-generation firewalls have become a core component of the network security architecture of most companies.

A core improvement that a next-generation firewall brings on top of a traditional firewall is deep packet inspection, or *DPI*. This refers to the capability to process the application layer information of a network connection. Whereas a traditional firewall can control traffic based on the IP address and port, a next-generation firewall is able

to perform additional controls based on information derived from application layer protocols such as HTTP, DNS, FTP and SMB.

The feature set of a next-generation firewall is not set in stone. Still, there are many features that are considered core functionalities of a next-generation firewall. These include:

Control of network applications. This capability enables the user to control the traffic based on the network application. These types of applications typically include web applications, such as Facebook, Youtube and LinkedIn. A next-generation firewall provides the ability to granularly define what kind of traffic is permitted and discarded in the protected network. In addition, this includes identification of protocols based on application layer information. This means that a next-generation firewall will be able to identify, and thus permit or deny, various protocol traffic in any port, providing a better evasion detection capability.

Intrusion Prevention capabilities. A next-generation firewall typically includes the capability to identify and prevent threats. This is done using the same deep packet inspection feature which is used for application control. Depending on the product, the level of intrusion prevention can vary. At the most basic form, it may include simple string matching on the observed application layer data to identify malicious strings in it. More advanced features may include a deeper level of protocol awareness and more complicated pattern detection mechanisms. There are also differences in the tolerance different next-generation firewall implementations have for network level evasions [35].

User level access control. Another typical feature of a next-generation firewall is the ability to control traffic based on user information. It provides the ability to control which users are allowed to access which restricted resources and to form groups of users with various permission or security levels. The authentication method usually depends from product to product and from use case to use case but might include integration to external authentication platforms like Active Directory.

Decrypting TLS traffic. Due to the current popularity of TLS, which at the time of writing comprises approximately 95% of the web traffic on the internet [36], the ability to decrypt TLS has become one of the core features expected from a next-generation firewall. There are often two ways of decryption, depending on whether the protected endpoint is a client or a server. When protecting a TLS server, the server certificates can be imported to the next-generation firewall which will then be able to decrypt the TLS traffic that the server receives. When protecting a client, the next-generation firewall can replace the server

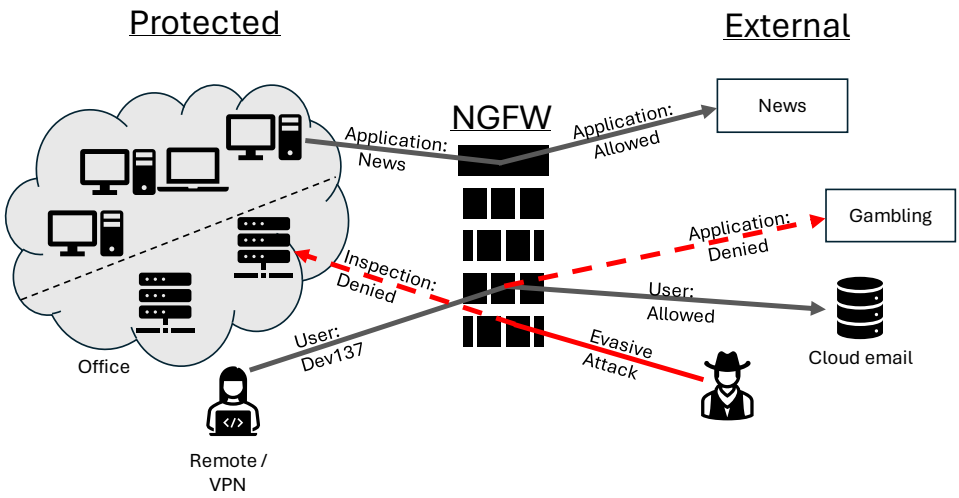

Figure 6. A next-generation firewall providing a range of different access control and security functionalities.

certificate with its own certificate. This will show on the client machine as an untrusted certificate unless it is imported as a trusted certificate in the system.

Despite offering a large variety of features on top of the so-called first-generation firewalls, the next-generation firewalls are typically installed at a similar place in a network: at the boundary between a protected network and an unsafe network. It may also separate two different protected networks from each other to provide better network segmentation. But the typical location is at the organization's own premises, separating the internet from the internal network.

A next-generation firewall is typically able to control network traffic on layer 3 of the OSI model. This means that it is able to, and usually used for, routing traffic between multiple networks and performing NAT. Many next-generation firewall vendors also offer a product with a similar functionality set as their next-generation firewall, but which controls network traffic on the layer 2 of the OSI model and is thus only able to perform packet switching without routing or NAT capability. These products are referred to as intrusion prevention systems, or *IPS*.

Figure 6 provides a simplified visualization of a next-generation firewall controlling the traffic flowing through it. It depicts different use cases where traffic is allowed or denied based on network application and user information, and where an attack is stopped by deep packet inspection. The protected network includes an office network as well as a remote user connecting to the NGFW using a VPN. The office network is segmented to a server network and a user network.

As services started migrating increasingly to the cloud instead of everything being run locally at the office premises, the requirements for network security solutions like a next-generation firewall also became more complex. A part of the protected

network now might reside in the cloud instead of a fully controlled static location. This brought again the need for another evolution for the concept of a firewall. Different next-generation firewall companies started to introduce cloud-based firewall solutions as part of their portfolio. At the time of writing, the evolution is still ongoing, moving towards a larger architectural change into Zero Trust models and Secure Access Service Edge or *SASE* solutions. These are covered in more detail in sections 3.4 and 3.5.

3.3 Other Network Security Solutions

There is a large variety of different network security solutions in addition to firewalls and next generation firewalls. Some organizations may not have a need for a full next-generation firewall, and may require something more specific. To the same extent, many of the network security solutions listed here may also be provided as a sub-component of a next-generation firewall. This section provides a high-level overview of other network security solutions and their purposes. The list is not intended to be a comprehensive list of all available network security solutions, but rather to provide an overview of the most common solutions.

IPS. An intrusion prevention system, or *IPS*, is a network security solution dedicated to identifying and terminating various attack attempts. It is located in the protected network at a place where all traffic will pass through it, which gives it an ability to block any malicious traffic. This differentiates it from its predecessor, intrusion detection system or *IDS*, which only received mirrored traffic and produced alerts from it but was not able to directly terminate an attack. An intrusion prevention system controls network traffic on layer 2 of the OSI model. Thus, it does not have the ability to perform routing between multiple networks, or NAT, which are functions performed by a next-generation firewall [37; 38; 39].

VPN. A virtual private network solution, or *VPN*, provides a secure way to access a protected network or restricted resources. It can be used for example for connecting one endpoint to a protected network, or connecting multiple protected networks, such as multiple remote offices, together in a secure way. The VPN traffic will be transported inside a secure tunnel. A VPN can be configured so that only a part of the traffic is directed through it, or so that all traffic goes through the VPN. Besides providing a secure way of transporting network traffic between two endpoints or sites, a VPN solution does not typically include any additional security features. There are VPN solutions marketed for individual users that typically obfuscate the source of the connection and can for example be used for making it look like the client is from another country. They are usually intended for providing privacy and protection for the user, or

access to resources that have a restriction based on location. VPN solutions marketed for companies are typically more large scale and used for providing access to the organization's resources [40; 41; 42].

Sandboxing. Sandboxing solutions are used for ensuring that downloaded files are safe. Simple anti-malware software typically runs some predetermined checks for a file, and if no malicious intent is found in the file that would match to the checks, the file is considered clean. By contrast, a sandboxing solution is usually a virtual environment which is aimed to mimic the protected endpoints as well as possible. When a new file is being downloaded, it is first sent to the sandbox where it is executed. The sandboxing solution is able to identify any malicious activities that executing the file causes and can thus block the download if it is considered malicious. If no malicious activity is found, the file is considered safe and can be delivered to the target machine [43; 44; 45].

DLP. Data loss prevention, or *DLP*, is a security solution which monitors the network for data breaches. DLP solutions typically have various methods of detecting data leaks, which include the ability to identify sensitive data, monitor outgoing data and securing data that is being stored. DLP solutions help organizations ensure compliance of many data regulations. DLP solutions can be deployed directly on the protected endpoints, as local servers at the organization's premises, or as cloud solutions [46; 47; 48].

SD-WAN. A wide-area network, or *WAN*, is a collection of several networks that are located at different geographic locations. The traffic between the networks is typically transported using dedicated multiprotocol label switching or *MPLS* circuits, which enables the traffic to be transported via a pre-determined route without the need for routing. These traditional WAN solutions are typically implemented using hardware, such as routers and switches. An SD-WAN solution provides WAN functionality using software-defined networking technology, or *SDN*. An SD-WAN solution is considerably less expensive to use than a traditional WAN solution while providing improved performance and typically more granular control of the traffic, such as routing the network traffic differently depending on the identified network application. In addition, a traditional WAN solution does not support the increasing adoption of cloud services, which is much better supported by SD-WAN solutions due to the improved routing capabilities [49; 50; 51].

SWG. A secure web gateway, or *SWG*, is a network security solution focused on providing protection for web traffic. An SWG usually has the ability to filter traffic using URL categories, to control traffic based on the applications, and to perform some level of file security activities such as running an anti-malware on a downloaded file or checking an uploaded file for potential information

leakage using data leak prevention or *DLP*. Decrypting HTTPS traffic is also a common feature in an SWG. An SWG can be a physical appliance installed on the premises of the protected organization, or a cloud-based solution where all web traffic is directed from the protected network or from the protected endpoints [52; 53; 54].

CASB. A cloud access security broker, or *CASB*, is a cloud-native network security solution. It is specifically dedicated for providing secure access and granular control of traffic targeted for cloud-based solutions. Typical features included in a CASB solution include authentication, authorization, threat protection and data security. CASB solutions also provide companies with the ability to maintain compliance with many regulations posed by various laws. A CASB solution also provides a good visibility into what types of services and functionalities are being used by the employees. A CASB solution can be installed as a local installation at the organization's premises, as a cloud-based solution where the target traffic is forwarded prior to access to the cloud resources, or the CASB solution can be outsourced to a dedicated CASB provider [55; 56; 57].

FWaaS. Firewall as a Service, or *FWaaS*, is a cloud-based implementation of a firewall. Despite its name, an FWaaS solution typically contains a full stack of next-generation firewall features, not just traditional firewalling. Instead of a need for installing a local hardware based next-generation firewall at the organization's premises, an FWaaS is located in the cloud. An FWaaS solution often provides a more flexible solution for companies with a more mobile workforce. In addition, it enables an organization to have a simpler IT infrastructure by removing the need of maintaining a hardware-based next-generation firewall appliance. They are also typically more flexible to deploy and control than hardware-based solutions [58; 59; 60].

3.4 Zero Trust

The concept of Zero Trust network architecture was first introduced in 2010 by John Kindervag and the Forrester Market Research organization [61]. This architecture contests the prevalent idea of a trusted network. The original network concept consists of networks that are considered secure and insecure, depending on whether they are located in the internal network or on the internet. A network security solution is installed in between the two, protecting the endpoints in the secure network from attacks originating from outside. Zero Trust network architecture does not instill trust on the endpoints purely for them being located in a specific network - it considers each entity a potential threat. Zero Trust Network Access, or *ZTNA*, is a network security solution, or a collection of several solutions, which enables an organization to build their network using Zero Trust network architecture [62; 63; 64].

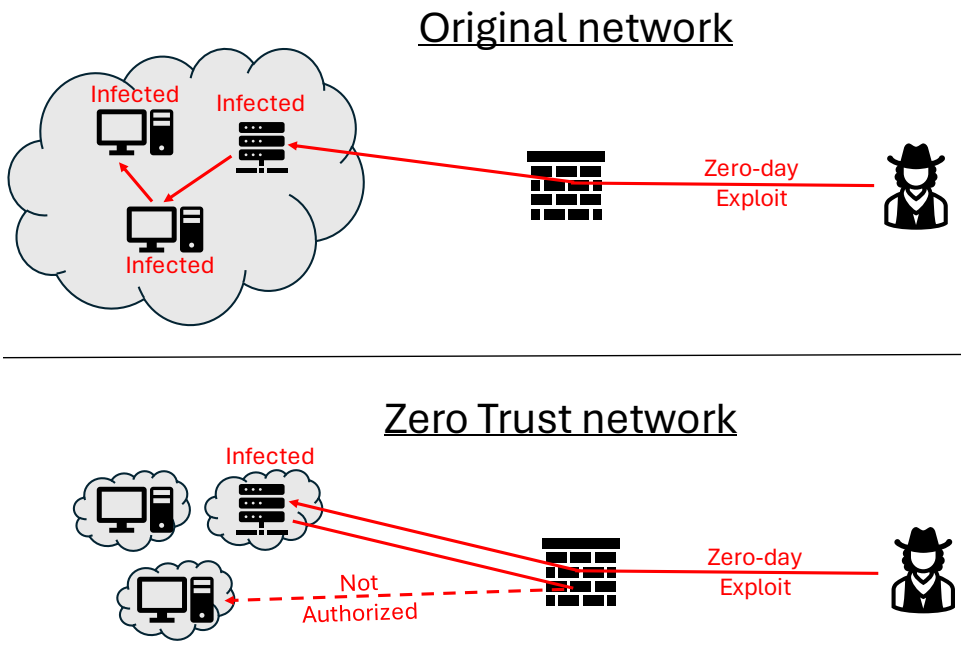

Figure 7. Top: a network built with the original concept of a trusted internal network. An attacker is able to spread an attack across the protected network once they gain access to one machine. Bottom: A network built with a zero trust network architecture. An attacker cannot progress in the network after gaining access to one machine.

In traditional network architecture a machine located in the internal network might get compromised, and because it is considered trusted, it is possible for an attacker to spread an attack to all other potentially vulnerable devices in the same trusted network. In such a scenario the network security solution has no visibility into what is happening, as the traffic does not go through it. In Zero Trust architecture all requests are authenticated and authorized, and access is only granted for the systems that are truly needed. This means that all requests will be processed by a network security solution, which prevents the above scenario from happening - the network security solution will be able to detect the attempts to spread the infection and will be able to block it [65; 66; 67].

Figure 7 depicts one simplified use case demonstrating the benefit of zero trust network architecture. It shows two successful attack scenarios: the first (top) in a network built with the original network architecture where the whole internal network is considered trusted, and the second (bottom) in a network built with zero trust architecture. In the first scenario, the attacker is able to compromise a server device located in the trusted internal network using an attack against a zero-day vulnerability that the network security solution is unable to detect. The server can directly access other devices in the internal network, and the attacker is able to gain access to

all devices in the same network with the same zero-day vulnerability. In the second scenario, the initial attack is again a success. However, this time the attacker is unable to progress to other devices as the compromised server machine does not have permission to initiate connections to the other protected devices.

There is some variation in what features a ZTNA solution provides, but there are two core features that all of them include.

Authentication. At the core of a ZTNA solution is the ability to verify the connections using authentication. This includes authenticating the users, but often also the ability to verify the identity of the device. Authentication is typically done continuously.

Limit access. The ability to limit access based on authentication is another core feature of a ZTNA solution. The aim is to fulfill the principle of least privilege, meaning that all entities in the network are able to access only the resources they truly need to reach. The granularity of access control depends on the solution.

3.5 SASE and SSE

The term secure access service edge, or *SASE*, was first introduced by the research and consulting firm Gartner in 2019 [68]. SASE is another step of evolution in the field of network security. It, again, originates from the increasing shift of services to cloud and the rising number of users becoming remote. The concept where users are located at offices and are accessing resources stored in data centers located at the organization's premises, is expired. Instead, a rising number of sensitive data is stored outside of the organization's own networks, and there are numerous complex use cases where the users, the data and the running workloads are located at various locations.

In their report, the Gartner researchers highlight the need for a new, converged approach to securing the resources of an organization. Protecting the resources had become complex, with a number of different network security solutions providing different services. The researchers defined this new market area as SASE, where the network solutions such as SD-WAN, and network security services such as SWG, CASB and cloud-based firewalls converge. In their vision, these services should be provided in one solution, providing a simpler policy configuration with a highly tailorable network fabric. Identity and context are described as being at the center of the SASE stack. Figure 8 depicts an example SASE solution on a high level.

The current available SASE solutions have slightly varying focus areas. Some highlight their advanced security features, some their data security capabilities, some the use of AI [69; 70; 71]. In general, most vendors list the following capabilities in their SASE solution:

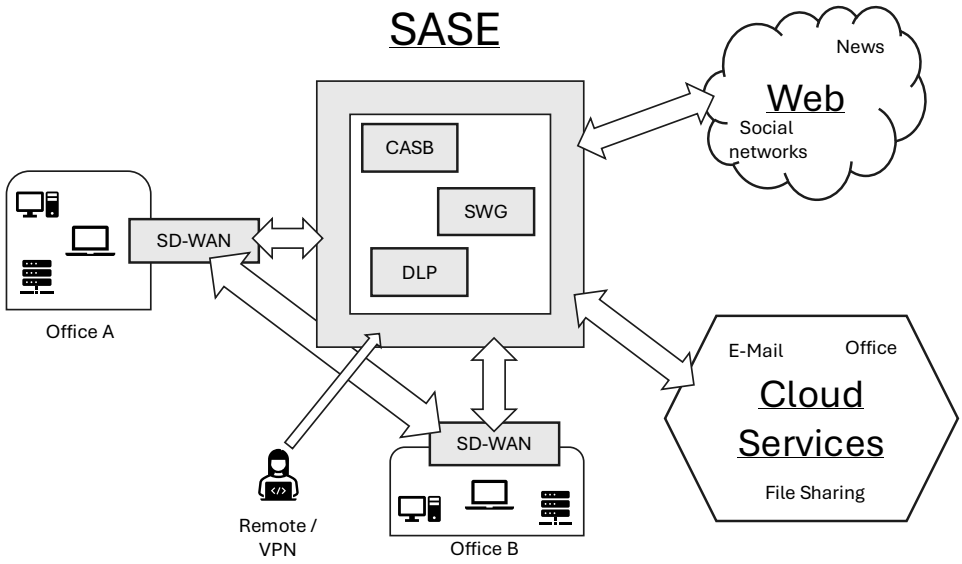

Figure 8. A SASE solution with a centralized cloud-based security service set and SD-WAN integration.

ZTNA. At the core of a SASE solution is the ability to authenticate and authorize the users with access only to the resources they need.

CASB. A CASB integration gives a SASE solution the ability to provide a strong control of and visibility into the users' web application usage.

SWG. An integrated SWG provides security for web traffic by providing URL filtering services and performing security scanning.

DLP. A SASE solution often includes support for DLP, which enables the solution to monitor and prevent data leakages.

SD-WAN. An SD-WAN capability seamlessly integrates the cloud-based services of a SASE solution into the wide-area network of an organization's offices, data centers and remote users.

Two years after introducing the concept of SASE, Gartner introduced a new network security solution concept: security service edge, or *SSE*. SSE was described as a subset of a SASE solution, consisting of the network security services included in SASE, but leaving out the networking side of the SASE concept provided by the SD-WAN integration. They described that an SSE solution can be completed with a separate SD-WAN solution to offer a complete security stack [72]. Figure 9 depicts an SSE solution without SD-WAN integration.

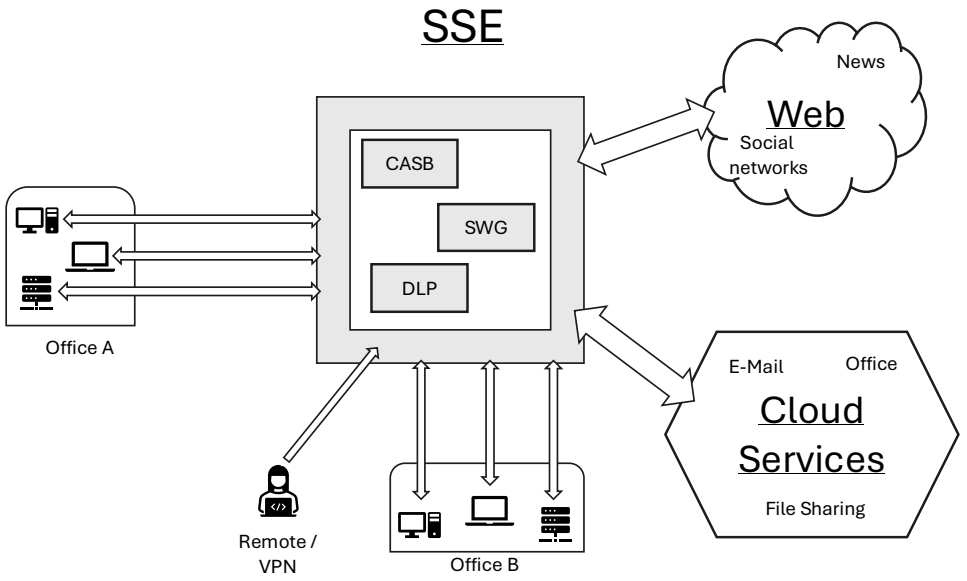

Figure 9. An SSE solution without SD-WAN integration.

Without the support for an SD-WAN, an SSE solution is typically provided as a cloud-based solution. It may also include components that are located at the organization's premises, or on the protected devices using agents. An SSE solution is often easier for new customers to adopt than a full SASE solution including SD-WAN. Most SSE solutions include all of the security features included in a SASE solution, including at least ZTNA, SWG and CASB functionalities [73; 74; 75].

4 Traffic Identification

When deep packet inspection became a core feature of a network security solution, there were several new security related areas to conquer. Deep packet inspection provided capabilities to identify network-based attacks on a completely different level as compared to a simple IP-address and port-based configuration of a first-generation firewall. In addition to improved attack detection, the visibility into the application layer of the network protocol stack brought along the ability to identify and categorize legitimate, baseline traffic.

As mentioned in section 3.2, the ability to control traffic based on the identified network level applications was the initial differentiator and selling factor used by the first next-generation firewall vendor Palo Alto. They were able to create a new market sector and gain customers for their new network security solution by introducing a need for more granular control of the network traffic that the customers did not know they had. Instead of an IP and port-based access control, the policy could now be built in a way which allowed network traffic that was essential for the employees to complete their tasks, but blocked traffic that was considered counterproductive, such as social networking or online video streaming services.

Network applications are elements that do not rely on IP addresses and ports but rather they are formed using application layer information gathered from the network traffic. Some network applications are used for identifying the application layer protocol, such as HTTP, TLS or DNS. In addition, network applications are used for identifying the service which is being used. With web traffic, the host name is used for identifying the web service at minimum. If the full URL, including the path, is available, such as with plain text HTTP or with decrypted HTTPS, a more granular network application identification may be possible. As an example, the host name may reveal that the service being accessed is a specific social networking platform. Upon seeing the path of the URL, it may be revealed that the particular network connection is associated with using the chat feature available in this platform. Such granular information can provide a network security solution with the ability of blocking the use of the chat feature of the social networking platform in the protected network while permitting other functionalities, such as browsing the organization's own page on the platform. In addition to network application identification, the URL information can also be used for URL categorization, which offers the ability to permit and block traffic based on more generic categories, such as News, Social Networking and

Gambling.

In addition to identifying the network level application and URL category, another interesting area of network traffic identification is identifying the endpoint application. An endpoint application is a software component sending or receiving a network connection. For example, an endpoint application used for accessing files using SMB protocol is typically not the same endpoint application that is used for browsing websites, and a Zoom conference is often accessed using an endpoint application specifically dedicated for Zoom conferences. Nevertheless, some endpoint applications can perform multiple different tasks. Zoom conferences can also be joined via a web browser, and some web browsers also support file transfer protocols like FTP.

The endpoint application used for a specific task can, however, affect what security features can be applied to the traffic. Some endpoint applications that have been developed for a specific purpose, such as accessing remote files in a cloud-based file storage service, may not tolerate decryption of the traffic. This means that a network security solution may not have any means of verifying that the downloaded and uploaded files are safe, or do not leak confidential information outside. If the same cloud-based file storage is accessed using a web browser, traffic decryption is typically possible, and file-based security features can be applied on the traffic. In addition, some organizations may want to ensure that restricted resources can only be accessed using dedicated software components.

In this chapter the various methods of identifying and categorizing network traffic are explored. The initial concept of network application identification is first introduced in section 4.1. URL categorization is next explained in section 4.2. The main interest of this thesis, endpoint application identification based on network traffic, is covered in section 4.3. Finally, we consider the effects of spoofing the traffic in section 4.4.

4.1 Identifying the Network Application

The ability to identify the *network application* of a network connection is a core feature of many network security solutions [76; 77; 78]. It provides the ability to build access policies in a much more granular way than simple IP-address and port-based policies of a first-generation firewall. A first-generation firewall requires that an IP address remains static, and that all IP addresses used by a given web operator are known. This is, however, not how the current internet functions. Cloud operators may serve certain operations from overlapping and dynamic IP addresses, and thus an administrator will get exhausted trying to keep their IP-address based access policy up to date with the cloud operators. This is where the concept of network applications becomes valuable and even necessary.

A network application may, for one, be used for identifying the application layer

protocol that is used: HTTP traffic has quite different use cases in transferring information about web sites than SMB or FTP traffic that are typically used for transferring files. DNS traffic is currently crucial for all internet traffic whereas streaming media over UDP may not always be a desired event. The ability to differentiate whether the traffic flowing in UDP port 53 is DNS traffic or a video stream is a valuable tool in a network administrator's toolbox, as is the ability to block file transfer attempts in TCP port 80 using FTP.

In addition to the protocol level network applications, a network application can also be used for identifying the web service which is being accessed. Examples of such web based network applications are Google, Facebook and LinkedIn. In some instances a network application may even incorporate several different protocols associated with the same service, such as audio and video streams. A high-level example is a network application used by a network security solution for controlling traffic for the network conferencing application Zoom [79]. Such a network application needs to be able to identify both the web connections associated with Zoom, such as accessing Zoom's web site, as well as the media connections transferring audio, video and other information related to a Zoom web conference session.

The ability to identify the web service can be very valuable for controlling what types of functionalities are permitted for the employees, but in many cases the ability to identify the application layer protocol provides better security features. With the visibility into the application layer protocol, the administrator can perform actions like protocol enforcement. Many UDP and TCP ports have been assigned for a specific application layer protocol by the Internet Assigned Numbers Authority, *IANA*. The most common examples are the TCP ports 80 and 443 assigned to HTTP and HTTPS, respectively, and the TCP and UDP ports 53 assigned for DNS. As these ports are typically open for outside access in any network where the internet is accessible, they are also used by many evasive tools that try to perform actions in the network that are otherwise not permitted. As an example, many software components intended for circumventing network restrictions use the open DNS ports for proxying web traffic through, and thus accessing otherwise prohibited web sites. By restricting the traffic in the assigned ports only to the assigned protocols, and dropping all other traffic in them, a network administrator can make sure that these ports are not used for any other traffic.

Figure 10 shows a simplified example of an access policy from Forcepoint Security Management Center version 7.2.1 where traffic is controlled using protocol level network applications. Rule 5.2.2 allows DNS, FTP, SMB, SMB2, SMB3 and SSH traffic through using network applications. This means that traffic in the ports assigned for each protocol is initially let through and deep packet inspection is performed to verify if the traffic follows the desired protocol. If the traffic is identified as the given protocol, the connection is allowed through. If not, the traffic is discarded by the following rule 5.2.3, which is configured to discard all traffic to the

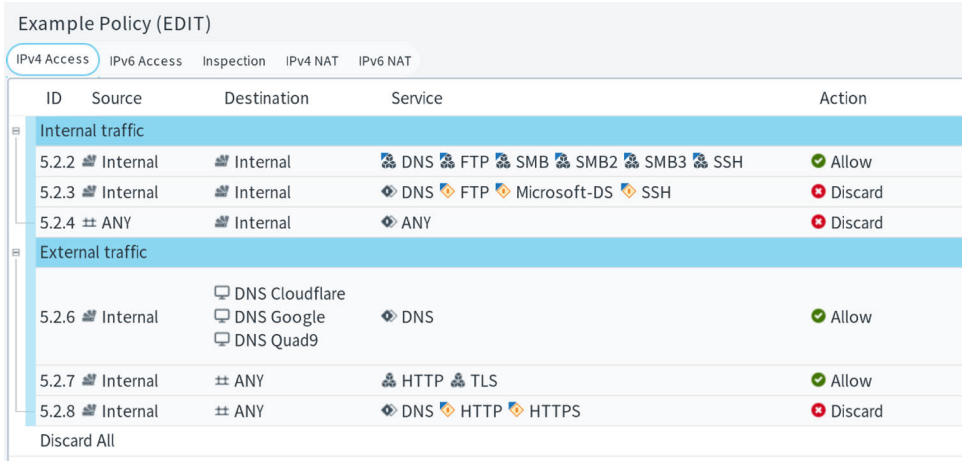

Figure 10. An example next-generation firewall access policy where traffic is controlled using network applications. Screen capture taken from a Forcepoint Network Security Platform.

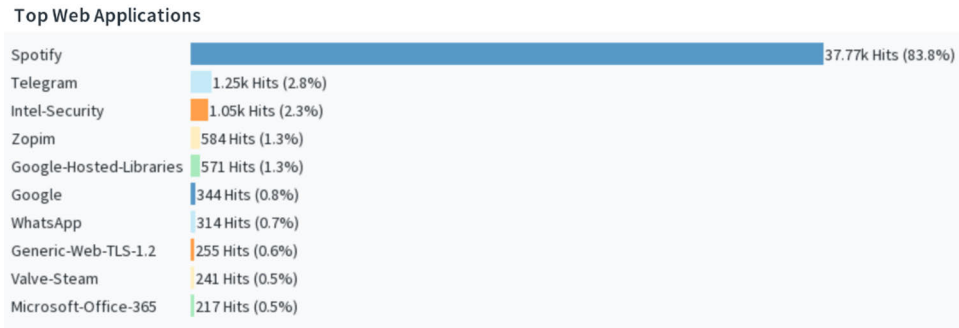

Figure 11. A report of top applications based on amount of transferred traffic. Report taken from a Forcepoint Network Security Platform monitoring a leisure network on 16th March 2024.

corresponding ports.

Most network security solutions offer identification for a set of common web services. They usually do not, however, have an individual network application for every existing host name and less-used service. This can typically lead to a situation where a part of the observed network traffic will always remain as *unknown*, or only have a protocol level identification. An example of this can be seen in figure 11. The figure is a screen capture of a top application report taken from a Forcepoint Network Security Platform which has been installed at the border of a small leisure network. It can be seen that a part of the traffic has only been identified based on the protocol as *Generic-Web-TLS-1.2*.

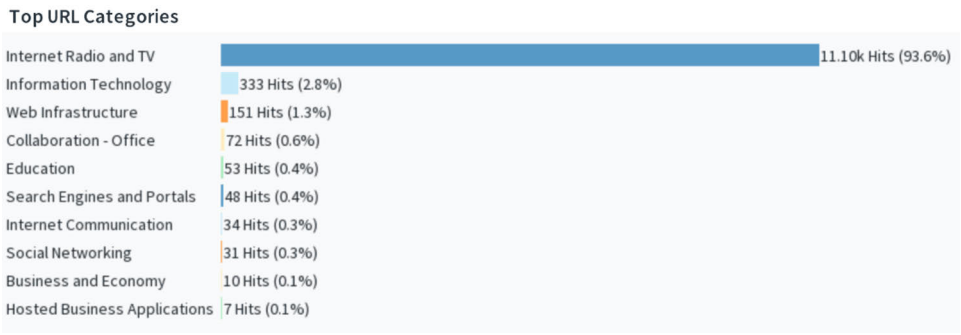

Figure 12. A report of top URL categories based on amount of transferred traffic. Report taken from a Forcepoint Network Security Platform monitoring a leisure network on 16th March 2024.

4.2 URL Categorization

As mentioned in section 4.1, the network applications typically do not cover all web hosts but focus on the most common ones. The concept of *URL categorization* is close to the concept of web type network applications, as it also focuses on web traffic. However, when categorizing a URL the exact service such as the exact social networking site is often not considered a category. Instead, the site will be categorized as a social networking site along with other similar sites. This provides the administrator with a broader scope of defining what type of traffic is permitted and what is not. And, contrary to the concept of network applications, the purpose of URL categorization often is that every host and URL has some kind of category assigned to it. URL categorization services are typically offered as cloud services.

URL categorization is usually first made based on the host name, after which a more granular categorization can be made when the full path is seen. This permits a high-level categorization to be available for HTTPS traffic even if decryption is not performed, but a more specific category can be given if the traffic is decrypted, and the full path can be seen. As an example, LinkedIn is a social media platform targeted for employed people who either wish to expand their work network or are looking for new positions, accessible by URL *https://linkedin.com*. LinkedIn also provides a learning platform with video courses for a variety of topics, which at the time of writing is accessible by URL *https://linkedin.com/learning*. As both reside under the host name *linkedin.com*, without decryption the two services will both receive a categorization based on the functionalities of LinkedIn, most typically related to social media. But if the traffic is decrypted and the path is observed to be */learning*, the category may be changed to something related to learning instead of social media. The difference may be crucial, as many organizations may wish to prevent access to social media but might wish to permit improving your work-related skills through a learning platform.

Figure 12 shows a screen capture of a report of top URL categories taken from a

Forcepoint Network Security Platform monitoring a leisure network. It demonstrates how the URL categories remain on a higher level than the network applications. The report can be compared with figure 11 which shows an application report from the same period.

In addition to the URL, URL categorization is typically made based on other information as well. This includes information in the headers and the contents of the site. If a URL categorization service does not have a categorization yet for a new site, it may visit it and apply real-time categorization based on what it sees. In addition, URL categorization services often integrate into other services that complement the categorization, such as intrusion detection databases, to better categorize malicious sites.

4.3 Identifying the Endpoint Application

Identifying either the client- or server-side endpoint application based on network traffic is not a common feature in network security solutions. Most network security solutions that do threat detection and traffic identification have some level of HTTP User-Agent parsing [80; 81; 82]. A few vendors have also implemented more advanced methods [18; 83]. Still, the ability to identify endpoint applications based on network traffic has mainly remained an academic curiosity. The academic research on the topic has been covered in chapter 2. In this section we provide a higher-level introduction to the concept.

In the context of this dissertation, the term *endpoint application* refers to a software component that communicates with other software components over a network. In a modern operating system there are numerous endpoint applications that are constantly making network connections, with or without an interaction from a user. A user may for example use a web browser for surfing various web sites, in which case the web browser is the endpoint application. In the background other endpoint applications are also making network connections, such as resolving the IP addresses of the web sites being accessed, checking for software updates, and sending telemetry. And at the other end of each of these network connections is another endpoint application - the server-side software component which either delivers the requested content or receives and processes the information sent to it.

In this section we introduce a few methods of identifying the endpoint application based on network traffic from the perspective of a network security solution. We first introduce and consider the HTTP User-Agent header field in subsection 4.3.1. We then dive into *hash fingerprinting algorithms*, a core topic in this dissertation, in subsection 4.3.2. Lastly, we go through active methods of gaining awareness of the endpoint application in a high level in subsection 4.3.3.

```
GET /online/ HTTP/1.1
Host: brightwholeserenekiss.neverssl.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/122.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,fi;q=0.8
```

Figure 13. An HTTP GET request showing a User-Agent from Chrome version 122.0.6261.128 on a Linux-based operating system.

4.3.1 HTTP User-Agent Field

With some network protocols it is easier to identify the endpoint application based on network traffic. The HTTP protocol is a prime example. Prior to the encrypted TLS protocol becoming the main transport method for web content, web traffic was transported as plain text HTTP. The HTTP protocol contains a request header field called User-Agent, which provides information about the endpoint application originating the HTTP request [3]. Thus, identifying the endpoint application from plain text HTTP is as simple as searching for specific strings from the contents of the User-Agent header. An example HTTP request showing a User-Agent from Chrome version 122.0.6261.128 on a Linux-based operating system can be seen in figure 13.

Since the use of plain text HTTP has almost entirely been overtaken by encrypted HTTPS [36], the User-Agent field is not as useful for identifying the endpoint application based on network traffic as it used to be. Now the network connection needs to be decrypted before the User-Agent can be seen. This limits its use cases, as not all traffic can be decrypted, and some functionalities of a network security solution require that the identification is made from the first observed data packet. An obvious example is that the decision whether to decrypt the traffic or not cannot be made based on this information, as receiving the information requires that the decision to decrypt was already made. In addition, the ability to identify the endpoint application already from the first data packet enables a network security solution to terminate the connection before any data has been transferred between the client and the server or make the decision to route the connection via another route.

4.3.2 Hash Fingerprinting Algorithms

The concept of hash fingerprinting algorithms was first introduced with the JA3 hash fingerprinting algorithm in 2017. The JA3 algorithm is intended for fingerprinting TLS clients, and further detailed in the following subsection. Since the publication of JA3, more algorithms were proposed for other protocols that followed a similar structure as the JA3 algorithm. In this dissertation the term *hash fingerprinting al-*

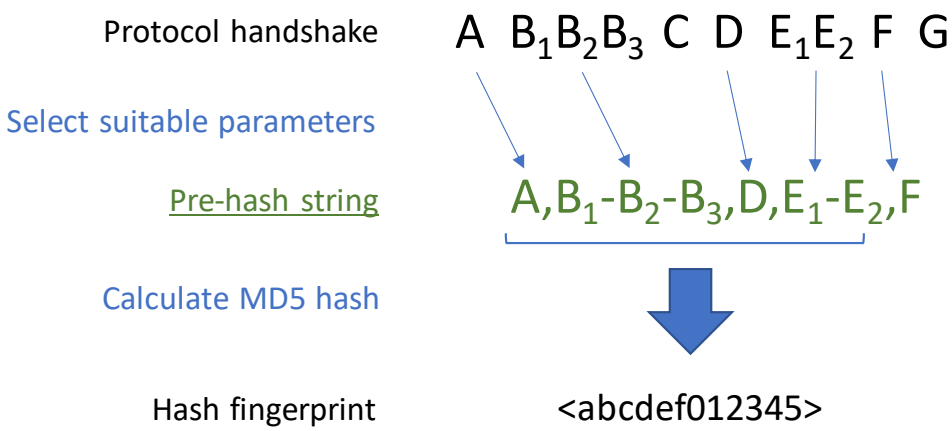

Figure 14. A high level visualization of a generic hash fingerprinting algorithm, from publication V.

gorithm is used when referring to the algorithms that follow this general structure. The structure that the hash fingerprinting algorithms follow will first be introduced here, and then further details will be given about the various algorithms proposed for different protocols.

For a network protocol to be a good candidate for a hash fingerprinting algorithm, it needs to have an initial handshake phase where the client and the server side agree upon a set of features and settings that they follow during the rest of the network connection. The supported features and the proposed settings, or the order in which they are presented, usually vary from one protocol implementation to another. The idea behind the hash fingerprinting algorithms is that by collecting these values from the handshake into a string and taking an MD5 hash of it, it is possible to generate unique fingerprints that can be used for identifying different implementations of the particular network protocol.

The purpose is to find a set of significant and distinctive parameters from the protocol handshake message. Depending on whether the purpose is to fingerprint a client or a server, the handshake message sent by the respective side is targeted by the algorithm. The values for these parameters are collected from the handshake message. There can be a different number of values proposed for different parameters: some parameters may permit more than one value, and sometimes an endpoint application might leave one parameter empty or completely out of the handshake message. Once the values have been collected, they are concatenated into a string, in an order predetermined by the particular hash fingerprinting algorithm. With most hash fingerprinting algorithms, the string is formed as follows: values for different parameters are separated by a comma, and multiple values for one parameter are separated by a dash. Some of the algorithms follow a slightly different notation, but all algorithms concatenate the calculated values into a string. In this dissertation,

this string is referred to as the *pre-hash string*. Once the pre-hash string has been formed, an MD5 hash is calculated from it, which comprises the final fingerprint and is the source for the high-level naming of these algorithms. The generic process of generating a hash fingerprint is visualized in figure 14.

JA3 Hash Fingerprinting Algorithm

The JA3 algorithm was the first algorithm to use the hash fingerprinting algorithm structure, and thus the first to introduce the hash fingerprinting algorithm concept for fingerprinting the endpoint applications. The algorithm was invented by Salesforce employees John B. Althouse, Jeff Atkinson and Josh Atkins, and published as open source in Github in 2017 [11]. Since then, the algorithm has been integrated into many network security solutions and tools, including Wireshark, a popular network packet analyzer tool [84].

The concept of fingerprinting TLS clients, which acted as the inspiration for the JA3 hash fingerprinting algorithm, came from a conference presentation by Lee Brotherston in 2015 [10]. In the presentation Brotherston introduced a set of tools which either sniffed a network or parsed an existing packet capture file for TLS Client Hello packets and collected certain information from them to identify the client application. The tools were able to export the fingerprints in C programming language structures or regular expressions to be used for further purposes. The presentation raised a lot of interest, but perhaps due to its dependency on specific software, it has not seen wide adaptation.

The Salesforce employees took this concept and created a more generic structure for presenting the information. They named the outcome JA3, which most likely originates from the three inventors' repeated initials. They did also publish tools for generating the JA3 fingerprints, but as the structure of the fingerprinting algorithm was the main publication, it did not require that the tools be used. Instead, it was possible to implement the fingerprinting algorithm into other solutions without integrating the tools.

The JA3 hash fingerprinting algorithm collects the following information from a TLS Client Hello packet: the used TLS version, a list of cipher suites that are listed as supported by the client, a list of extensions that are present in the Client Hello packet, a list of supported groups and a list of supported elliptic curve point formats. The concrete functionality of each of these fields is not relevant in this context, as only the numeric value for each field is used for the fingerprint. The values are collected into the pre-hash string in the following order: *version, cipher suites, extensions, supported groups, elliptic curve point formats*. The final JA3 fingerprint is then generated by taking an MD5 hash from this string. Figure 15 shows a screen capture from Wireshark of a captured TLS Client Hello packet, generated using OpenSSL. After the information collected from the TLS Client Hello packet, the screen capture

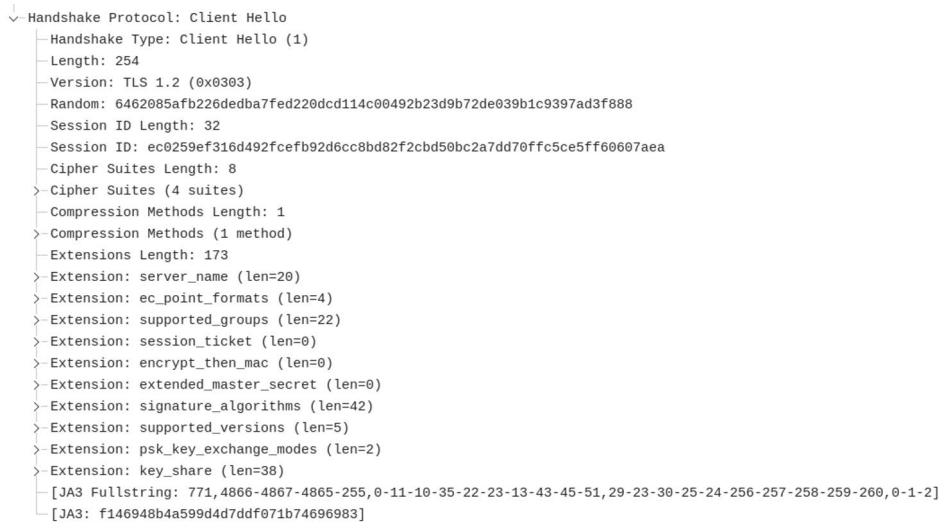

Figure 15. A screen capture of a TLS Client Hello packet from Wireshark, showing the JA3 fingerprint.

also shows the JA3 pre-hash string generated by Wireshark (as *JA3 Fullstring*), and the final JA3 fingerprint.

Since its publication, the JA3 hash fingerprinting algorithm has become a popular tool for network security analysts. As mentioned, the algorithm has been implemented in many network security solutions and tools. Public databases have been created for collecting JA3 fingerprints [85] and network security solutions support it [83]. In addition, they have become a useful addition when listing indicators of compromise information, or *IOC*s [86].

Other Hash Fingerprinting Algorithms

After the JA3 hash fingerprinting algorithm was published by Salesforce, more researchers became interested in the concept and proposed algorithms for other protocols that followed the same structure. Over the next few years, the field was active and the following hash fingerprinting algorithms were proposed:

SSH client and server: HASSH and HASSHServer. A bit over a year after the publication of the JA3 algorithm, in 2018, another Salesforce employee Ben Reardon published hash fingerprinting algorithms for identifying SSH clients and servers, called HASSH and HASSHServer respectively [12]. Both fingerprints are generated based on the SSH_MSG_KEXINIT message, sent in clear text by both the server and the client, and their structure is in essence identical. The values collected from the SSH_MSG_KEXINIT message by the algorithms are the list of supported key exchange methods, list of acceptable encryption algo-

Figure 16. A screen capture of a TLS Server Hello packet from Wireshark, showing the JA3S fingerprint.

rithms, list of acceptable MAC algorithms and list of acceptable compression methods. When forming the pre-hash string for HASSH and HASSHServer, multiple values in any of the lists are separated with a comma, and the lists themselves are separated by a semicolon. With HASSH and HASSHServer, the values are collected into the pre-hash string in the following order: *key exchange methods, encryption algorithms, MAC algorithms, compression methods*. The final fingerprint is, again, formed by taking an MD5 hash from the pre-hash string.

TLS server: JA3S. In January 2019 the inventors of the JA3 hash fingerprinting algorithm introduced JA3S, which was intended for fingerprinting TLS servers [87]. The syntax was similar to the JA3 algorithm, but the information available in the TLS Server Hello message is more limited than what is available in the TLS Client Hello message. The JA3S algorithm collects the following information from the TLS Server Hello message: the version, the selected cipher suite, and the list of extensions present in the message. The pre-hash string for JA3S is formed in the following order: *version, cipher suite, extensions*. Since its release, the inventors of the JA3S fingerprint have noted that the JA3S hash fingerprinting algorithm is not as useful as the JA3 algorithm as one server can generate so many variations of the collected values depending on what the initial TLS Client Hello message had. Nevertheless, the JA3S algorithm has also been implemented in many network security tools, like Wireshark. Figure 16 shows a screen capture from Wireshark with a captured TLS Server Hello message sent by the web page for University of Turku, where the JA3S pre-hash string (referred to as the *JA3S Fullstring*) and the final JA3S fingerprint are also shown, generated by Wireshark.

RDP client: RDFP. In the summer of 2019 an independent researcher, Adel Karimishiraz, published a hash fingerprinting algorithm for fingerprinting RDP clients, called RDFP [14]. The RDFP hash fingerprinting algorithm collects the needed information from several data blocks sent during the setup phase. It first collects information about the client's major and minor versions from the Client Core Data block. Next, the Client Security Data block is parsed, and the encryption methods are collected, as well as the extEncryptionMethods from the clients using the French locale. Then, the Client Cluster Data block is processed, and the cluster flags are collected. Finally, the Channel Definition Structure in the Client Network Data block is processed, and the channel options are collected. The values are added to the RDFP pre-hash string in the following order: *major version, minor version, cluster flags, encryption methods, extEncryptionMethods, channel options*. The RDFP hash fingerprinting algorithm follows the typical hash fingerprinting structure, where the values for different parameters are separated by a comma, and several values for the same parameter are separated by a dash. The MD5 hash taken of the pre-hash string forms the final RDFP fingerprint.

gQUIC client: CYU. In August 2019, soon after the publication of the RDFP algorithm, another hash fingerprinting algorithm was released by Salesforce employee Caleb Yu. The algorithm was called CYU, and it was intended for fingerprinting the gQUIC, or Google QUIC, protocol [13]. At this time, there was no official RFC version of QUIC, and the preliminary version of the QUIC protocol was the mainly used one. It was fairly different from the current RFC version of QUIC [88], and CYU does not work on the RFC version. However, the RFC version of QUIC heavily uses TLS, and the JA3 hash fingerprinting algorithm works for it. The CYU algorithm collects the version and the list of tags included in the client hello message of a gQUIC connection. The pre-hash string for CYU is formed as *version, tags*. CYU also follows the typical hash fingerprinting algorithm structure, where the version is separated from the tags by a comma, and the different tags are separated from each other by a dash.

DHCP client: KYD or DHCPFP. Also in August 2019 another hash fingerprinting algorithm was published by independent researcher Fatema Bannatwala [89]. It was called KYD or DHCPFP and was intended for fingerprinting DHCP clients. The DHCPFP algorithm collects the options presented by the client in the parameters list of a DHCP request, and the pre-hash string is formed by separating the collected options by a comma. The DHCPFP fingerprint is then received by taking an MD5 hash of the pre-hash string. The researcher suggested that the best use case for DHCP fingerprinting was for organizations to collect a fingerprint database of the devices in their network, which can then be used as a list of trusted devices.

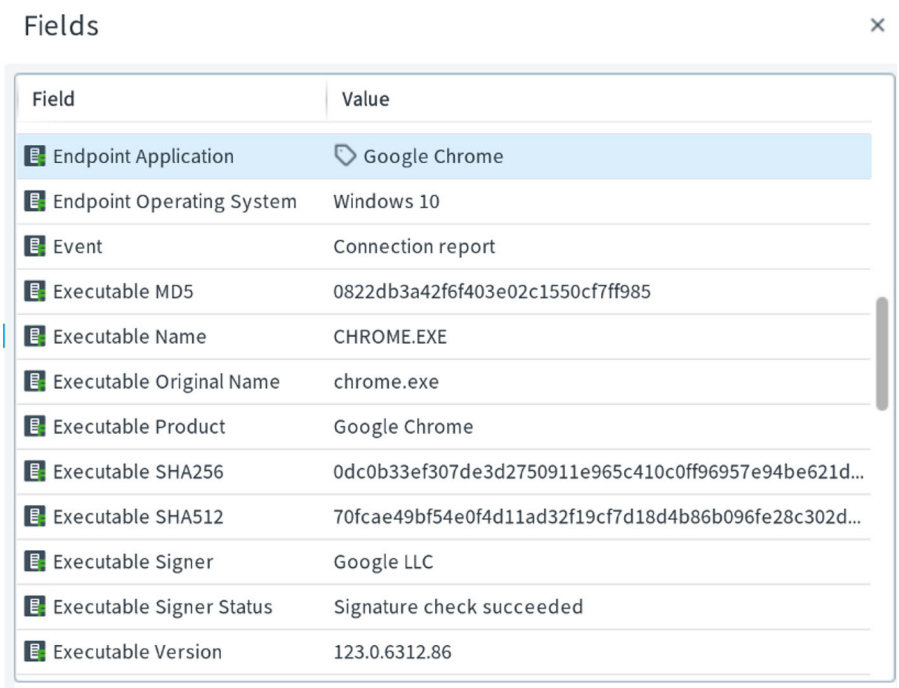

Figure 17. Example endpoint metadata from Forcepoint Network Security Platform.

SMB client: SMBFP. In May 2020 another independent researcher, Michael R. Torres, published a hash fingerprinting algorithm for fingerprinting SMB clients [15]. The algorithm was called SMBFP and it remains at a very preliminary stage. The work has not been continued since its release. The following fields are, however, mentioned as collected by the algorithm: the list of versions supported by the client, the maximum supported buffer size, the maximum amount of simultaneous SMB commands, the OS of the client, the client's LAN manager type, the primary domain, Unicode support and the support for read-only opportunistic locking.

4.3.3 Active Methods

The methods for identifying endpoint applications introduced thus far have been passive, meaning that no interaction with the protected endpoint is needed to gain the information. In addition to passive identification methods, there are also active methods that can be used for gaining information about the endpoint application. In this subsection the most relevant active methods in the context of a network security solution are explored. These methods either require an additional software component to be installed on the protected endpoint, active interaction with the protected endpoint,

```
jenny@hal9001:~$ nmap -sV finite.group
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-03 17:36 EEST
Nmap scan report for finite.group (46.101.166.123)
Host is up (0.030s latency).
Not shown: 996 closed ports
PORT   STATE    SERVICE VERSION
22/tcp open     ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
25/tcp filtered smtp
53/tcp filtered domain
80/tcp open     http    lighttpd 1.4.35
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.37 seconds
jenny@hal9001:~$
```

Figure 18. A screen capture of an active scan performed using nmap on a remote server.

or maintaining a separate asset database. Some of the methods can be very powerful, but they cannot usually be applied on all protected devices due to their more invasive nature.

Endpoint metadata. The most accurate approach for receiving information about the endpoint application initiating a network connection is to receive trustworthy metadata from the endpoint itself. Some network security solutions have implemented a feature which enables a protected endpoint to communicate with the network security solution [90; 91]. This can be achieved by installing an additional software component on the protected device which is able to send metadata to the network security solution about each network connection. In this case, the network security solution does not have to guess the endpoint application based on the network traffic - the endpoint reports what the actual endpoint application is. Example endpoint metadata received by a Forcepoint Network Security Platform about a network connection initiated by the Chrome browser can be seen in figure 17.

Active scanning. Active scanning is the process of actively initiating connections and making identifications based on the received information. Active scanning can typically only be applied to server-side endpoint applications, as forcing a client to start a network connection cannot generically be done in the same way. But using active scanning, combined with server-side fingerprinting methods, can provide a good understanding about the protected server endpoint applications. The most common and well-known active scanner is Nmap [92]. Figure 18 shows an example scan performed using Nmap on a remote server, which has SSH and HTTP servers installed.

CMDB. A configuration management database, or *CMDB*, is a database which contains details about an organization's information systems. This information can be as granular as to include what endpoint applications are installed on the

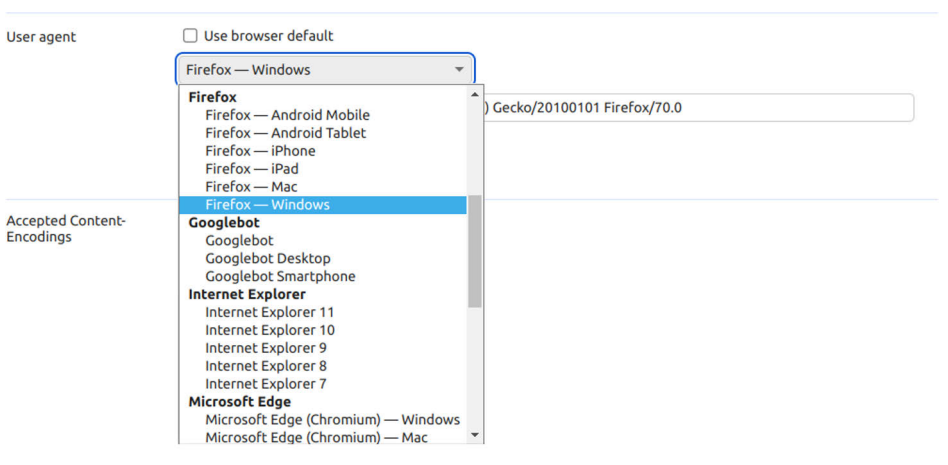

Figure 19. A screen capture of the different User-Agent values that can be selected in a Google Chrome web browser version 122.0.6261.128.

protected endpoints, although it is not a typical use case. If a CMDB included this level of granular information, a network security solution could use it to find out if the protected endpoint has a potentially vulnerable endpoint application or not.

4.4 Spoofing the Traffic

The traffic identification methods presented in this chapter, especially the passive methods, are prone to misuse as it is possible for a sophisticated attacker to modify the traffic in a way which gets identified as something else. As easy as it is to use the User-Agent to identify the endpoint application, it is similarly easy to provide a false value for it. In fact, spoofing the User-Agent is a common feature in modern web browsers [93]. Figure 19 shows example User-Agent values that can be selected in a Google Chrome web browser version 122.0.6261.128 when accessing a web site.

This feature sometimes does have its uses. Certain web services may offer slightly different content depending on which endpoint application, based on the User-Agent string, is used for accessing the service. And because of this, certain content may only be reachable when using the right endpoint application, or by spoofing the User-Agent. Because of this, the value in the User-Agent field is not something that should be blindly trusted when performing endpoint application identification based on network traffic.

In a similar way, it is possible for a HTTPS client and server to collaborate so that the initial TLS layer shows that the traffic is something while the decrypted HTTP traffic contains something entirely different. Based on the information available without decryption the client may seem to be accessing something permitted by

the organization, while the server is actually offering something that is prohibited. The Encrypted Client Hello extension for TLS provides a simple and efficient way for implementing something like this [94]. A generic service provider can host various different services, such as news and gambling services. If the network security solution does not decrypt the traffic, it can only verify what is shown to it on the TLS layer, which in this case would be the host name of the generic service provider.

Spoofing the parameters listed by a hash fingerprinting algorithm and used for identifying the client or the server endpoint application is trickier. The parameters are typically something that the specific endpoint needs to support for the connection to succeed. Adding support for a list of features needed for mimicking a specific endpoint application may require a lot of resources. It is still possible to spoof these values. This should be taken into account when considering what the information can be used for, and how much trust should be applied to it.

One potential use for the endpoint application identification is to provide more accurate identification of potential attacks by targeting the protection only for the vulnerable endpoint applications. If some network connection then gets terminated by a network security solution because the traffic contained a potential attack against the endpoint application which the actual application was only pretending to be, the produced false positive can be considered self-inflicted. As an example, a Google Chrome pretending to be an Internet Explorer 11 might not be able to download a file which might trigger a vulnerability in an Internet Explorer 11. This can be considered an acceptable false positive. Of course, it is also possible for a Google Chrome pretending to be an Internet Explorer 11 to download something that triggers a vulnerability in Google Chrome, thus deteriorating the threat detection capabilities of a network security solution. Because of this, an organization should not encourage spoofing of the endpoint application.

The endpoint application identification can also be used for ensuring that restricted resources are not accessed by unintended software components, such as for enhancing a Zero Trust network architecture. When the identification is made based on the network traffic alone, it needs to be understood that the information can be spoofed. If the identified endpoint application is something that clearly should not be permitted to access the resource, the connection can be dropped. But the identification should not be used as a basis for reducing security features for the connection. Similarly, changing the risk score of a network event should be considered well if the endpoint application is identified purely based on network traffic. The risk score can be raised if the identified endpoint application is something suspicious or unknown, but lowering the risk score, and thus potentially lowering security features applied to the connection, should not be done.

Despite the deficiencies mentioned above, the ability to identify various aspects of network traffic introduces remarkable benefits for a network security solution. As long as the shortcomings introduced here are properly addressed during feature im-

plementation and access policy design, traffic identification provides the user with significantly improved security capabilities. Especially the ability to identify endpoint applications and bringing endpoint awareness into the inspection process of a network security solution is a feature which has been left for very little attention among network security vendors considering its potential benefits.

This dissertation focuses on discovering the most efficient ways of identifying the endpoint application in a network security solution and thus gaining endpoint awareness in the traffic inspection process. The following chapter introduces five original publications that comprise this dissertation and summarizes their key results. These publications explore both existing methods introduced in this chapter as well as new, novel methods of identifying the endpoint application. Finally, a methodology for bringing endpoint awareness into the inspection process of a network security solution, entitled JAPPI, is introduced, which utilises clustering algorithms on the prehash strings of hash fingerprinting algorithms.

5 Summary of Key Results in Original Publications

When making an inspection-based decision in a network security solution, any additional context about the network connection can be valuable. It will provide the network security solution with an improved ability to decide whether a security event should be produced or if it can be silenced for that particular connection. With this additional context, a network security solution can reduce the amount of false positive and false negative security events, providing better overall value for its users.

The objective of this dissertation was to identify ways of bringing endpoint awareness into the inspection process of a network security solution. Focus was on finding methods of identifying the endpoint applications of a network connection and understanding how they should be integrated into the inspection process. In this chapter we give a summary of the key research findings of the original publications comprising the dissertation. We begin by providing a high-level summary of the structure of the dissertation where we shortly describe each publication's connection to the rest of the research. We then go through each publication individually, providing an outline of its contents, followed by its contributions and how it correlates to the research questions we formulated in section 1.2. We also describe future research that each publication introduces.

5.1 Research Structure

The research consists of six publications, which we will hereon refer to using roman numerals. Publication I is a patent filed for a novel method of identifying unknown traffic. This first publication brought forward the need of a better understanding of other existing methods for identifying unknown traffic, especially of gaining a better awareness of the endpoint which originated the network connection. Publication II answered to this need by providing a thorough study of existing research on the topic of gaining endpoint awareness in a network security solution, as well as a review of current implementations in network security solutions. Based on the study, publication III selected one topical method, hash fingerprinting algorithms, and introduced a method for adding endpoint awareness into the inspection process of a network security solution based on these algorithms.

After the method was published, it started to become clear that the final step of

the hash fingerprinting algorithms, taking the MD5 hash of the pre-hash string, was making the algorithms less useful. This step caused all information contained in the pre-hash string to be lost, and even minimal variations in the pre-hash string caused the final fingerprint to be entirely different. This was addressed in publication IV. This publication indicated that these weaknesses in the pre-hash algorithms could be bypassed by removing the final step of the algorithms and using the original pre-hash strings instead. This provided a ground for experimenting with machine learning for improved identification, which was done in publication V.

After publication V, the Chrome browser introduced an update which randomized the order in which the TLS Client Hello extensions were presented during the TLS handshake [2]. This update made the risk brought forward by publication IV a reality - in practice it was now impossible to fingerprint the Chrome browser as each connection produced a different MD5 hash. Since publication V demonstrated that machine learning provided superior results when using the pre-hash string of the hash fingerprinting algorithms, the final publication VI formulated an improved, novel methodology for gaining endpoint awareness in a network security solution using unsupervised machine learning on the pre-hash strings. This publication also presented an experiment on two different real life network environments and showed that the methodology was efficient in providing endpoint awareness.

5.2 Publication I: Identification of unknown traffic based on transport layer security extensions

Publication I is a patent filed for defining how the information in TLS extensions can be used for identifying previously unknown traffic. The original problem which introduced the need for the patented solution was the presence of traffic where only the protocol was known, but no additional information or context was available. This kind of traffic ends up in practice being categorized as unknown in a network security solution. The problem was especially emphasized with TLS traffic, which at that point comprised by far the largest portion of internet traffic. With TLS it was observed that in some cases a considerably large portion of the network traffic was left identified as unknown TLS.

When further considering the issue, it was recognized that the TLS extensions included in the protocol handshake procedure often contain useful additional information about the source endpoint. As an example, the Next Protocol Negotiation extension or *NPN*, and the Application Layer Protocol Negotiation extension or *ALPN*, provide information about the protocol that will be used inside the encrypted TLS layer. This often provides an indication whether the TLS connection is a web connection, such as a human browsing the internet using a web browser, or something else, such as a script fetching information using an API. This kind of additional information about the network connection can be very valuable when making security-

based decisions or actions in a network security solution. An example of how this can be done in a network security solution is presented in figure 20.

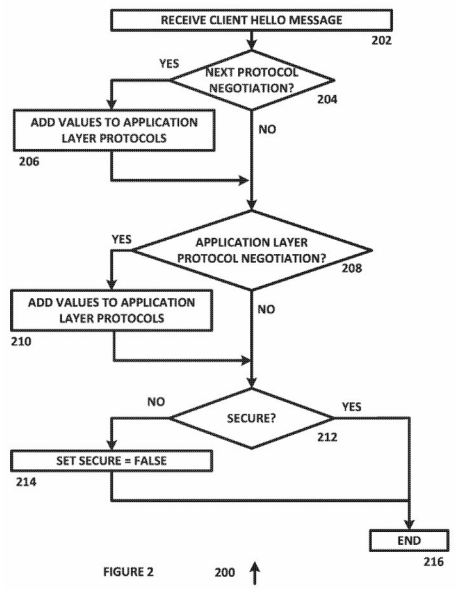

Figure 20. An example procedure of processing a TLS Client Hello message as described in publication I.

The concept has since been implemented in the Forcepoint Network Security Platform, which at the time of writing uses this information for better categorization of unknown TLS traffic. Currently the information is used for separating unknown web traffic from other unknown TLS traffic. The concept has been proven in real life environments to provide an improved ability to decide whether the traffic should be processed further, for example by a web proxy, or potentially discarded. Other potential use cases have also been identified and are under consideration for future improvements.

5.2.1 Contribution and Future Work

This publication introduced a novel method of receiving additional information about a previously unknown TLS connection. It provided an initial answer for research question **RQ1**. It also highlighted the need for additional research on potential methods of gaining more information about the network connection. By introducing the ability to separate web traffic from non-web traffic, this publication provided an initial spark for the research of finding ways to introduce a better endpoint awareness into the inspection process of a network security solution. This publication also

brought a concrete real-life benefit by presenting something that has since been implemented in a commercial network security solution.

5.3 Publication II: Study of methods for endpoint aware inspection in a Next Generation Firewall

Publication II is a journal article written as a review of literature and current state of the art. It performs a literature review of various methods for gaining information about an endpoint in a network security solution, exploring both historical and more recent methods. The various methods are separated into passive and active methods. The taxonomy of the considered methods is shown in figure 21.

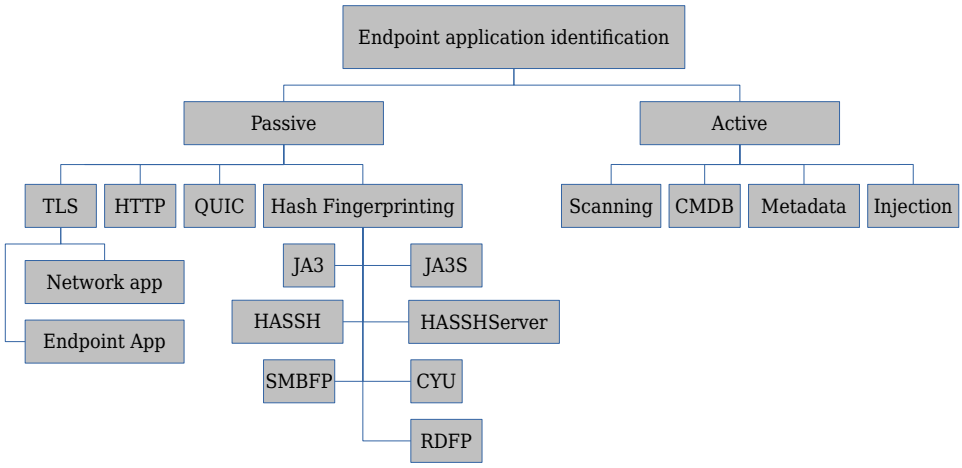

Figure 21. Endpoint application identification taxonomy as covered in publication II.

An active method requires that an active component be involved in the process of gaining endpoint awareness, be it a network scanner, a CMDB, or an external component installed on each protected endpoint. Out of these methods, the external component installed on a protected endpoint provides the most accurate method for gaining awareness of the endpoint applications. Such a component can for example let the network security solution know the original endpoint application of every network connection initiated by the endpoint. The downside is that such an approach requires that each protected endpoint device is in full control of the network administrator. In many network setups this cannot be achieved - bringing personal devices into the network may be allowed, and some endpoints in the network may even be temporary virtual environments that are only used for a limited period of time.

A passive method is something that does not need any additional interaction be-

tween the network security solution and the protected endpoint - the method can be applied purely based on the observed network traffic. Several different passive methods were identified during the study. The concept of hash fingerprinting algorithms rose up as the most prominent method. These algorithms were considered topical, and the first hash fingerprinting algorithm, JA3 which is intended for fingerprinting TLS clients, had gained popularity among the security research community.

In addition, the publication examines existing implementations of the various methods of introducing endpoint awareness in current NGFW products. Five prominent NGFW vendors were selected for the review, and the methods supported by each vendor were listed. Only publicly available information was used for the review, and thus it was left partially unclear how well each feature was integrated into the inspection process of the products.

5.3.1 Contribution and Future Work

This publication provided a comprehensive understanding about the current status of providing endpoint awareness in both academic research and state-of-the-art implementations of Next Generation Firewalls. It provided a more thorough answer for research question **RQ1**. Publication II highlighted which methods seemed the most topical and provided the most fruitful ground for future research. In particular, it suggested that despite being potentially the most powerful tool for gaining endpoint awareness in an NGFW, active methods can often not be used to cover the entire protected network. This led the research to focus more on the passive identification methods. Out of the passive methods this publication brought into light the concept of hash fingerprinting algorithms, which were identified as a highly topical area for research and the most interesting method for gaining awareness of the endpoint.

5.4 Publication III: A Method for Endpoint Aware Inspection in a Network Security Solution

Publication III is a journal article detailing a method for integrating endpoint awareness into the inspection process of a network security solution. The method utilises a trusted subset of the protected endpoints, called the Source, to collect hash fingerprints and endpoint metadata about each suitable network connection. This information is stored in a database so each hash fingerprint can be mapped to all endpoint applications that have produced it. After a pre-determined period of time of populating it, the database will start to be utilised by the network security solution on the entire protected network, called the Target. When the network security solution sees a connection from an endpoint which does not belong to the Source, it calculates the hash fingerprint from the connection and compares it to the database. If the database contains entries where one or more endpoint applications have been seen to produce

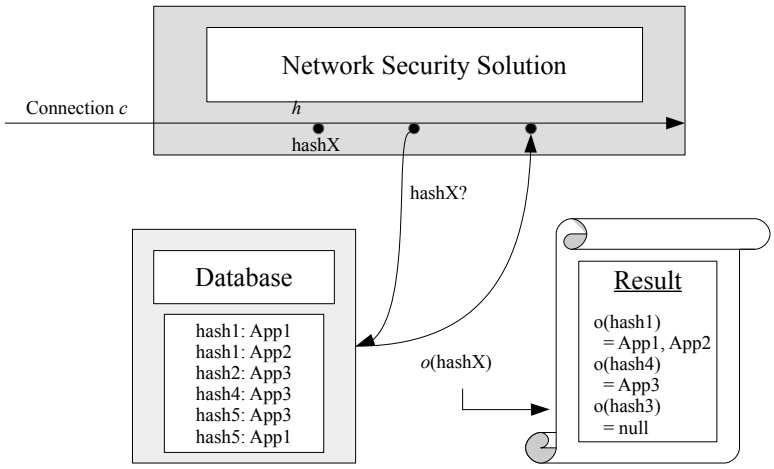

Figure 22. A visualization of the utilisation of the method from publication III

the same hash fingerprint, the network security solution can assume that the network connection has been produced by one of these applications, and the information can be further utilised in the inspection process. A visualization depicting how a NGFW should utilise the database on the entire Target network is shown in figure 22.

In addition to formulating the method, this publication contains a small-scale Proof of Concept experiment to validate the method. A small network segment containing endpoints with Windows and Linux based operating systems was monitored, and two Windows machines were selected as the initial Source. The database was first populated for two weeks, after which the database was utilised on all traffic. After this period the accuracy of the method was assessed by verifying if the database contained the hash fingerprints observed during the utilisation period. It was observed that the coverage of the database was perfect for all connections produced by the Windows machines where the endpoint application was also installed on the Source machines when the database was being populated. In addition, the coverage of the connections from the Linux machines was surprisingly good when the endpoint application was also installed on the Source machines, despite the Source machines using a different operating system.

As part of the experiment, a false positive test was also conducted. In this test, a set of sample files were accessed from one machine using the Firefox web browser. The sample files contained exploits against certain versions of Internet Explorer, Microsoft Outlook, and Android native web browser, but did not pose any risk for the Firefox web browser. The false positive test demonstrated that without using this method, the NGFW terminated the downloads. It also showed that had the method been utilised by the inspection process of the NGFW, the endpoint application would

have been identified as something that was not at risk, and the download would have succeeded.

5.4.1 Contribution and Future Work

This publication introduced a novel method of bringing endpoint awareness into a network security solution without the need of installing an external component on all monitored endpoints. It provides an answer for research question **RQ3**. It includes a small-scale proof of concept experiment which demonstrated promising results. The experiment was, however, very light weight, and thus left room for a future publication where the effectiveness of the method is validated on a larger scale. These results from the experiment did, however, indicate that the same endpoint application can produce the same hash fingerprint on different operating systems, and thus that the method may be useful even in cases where all operating systems used in the Target are not covered by the original Source.

5.5 Publication IV: On Usability of Hash Fingerprinting for Endpoint Application Identification

Publication IV is a conference research paper. It addresses an issue with the hash fingerprinting algorithms which was identified after publication III. Publication IV notes that the information presented in the pre-hash string, which contains a list of the parameter values presented by the endpoint during the handshake process, is considerably more valuable than the final MD5 hash taken from it. In addition, it argues that taking the MD5 hash does not provide any considerable benefit over using the pre-hash string, but instead leaves the hash fingerprinting algorithms vulnerable for attacks such as randomizing the order of the parameter values, which could make them nearly useless. An example of the process of a hash fingerprinting algorithm, in this case the JA3 algorithm for fingerprinting the TLS Client Hello message, is shown in figure 23.

The publication includes a small example demonstrating the issue. Sample pre-hash strings and hash fingerprints are collected from four endpoint applications, two samples from each. The pre-hash strings produced by the same endpoint application are seen to be very close to identical, adding or removing one value from the string, but the MD5 hashes taken from the pre-hash strings are entirely different, losing all information about the closeness of the original values. In addition, the pre-hash strings produced by two different endpoint applications that are known to use the same underlying TLS library are seen to be similarly close to identical, with one sample being exactly the same between Google Chrome and Microsoft Edge.

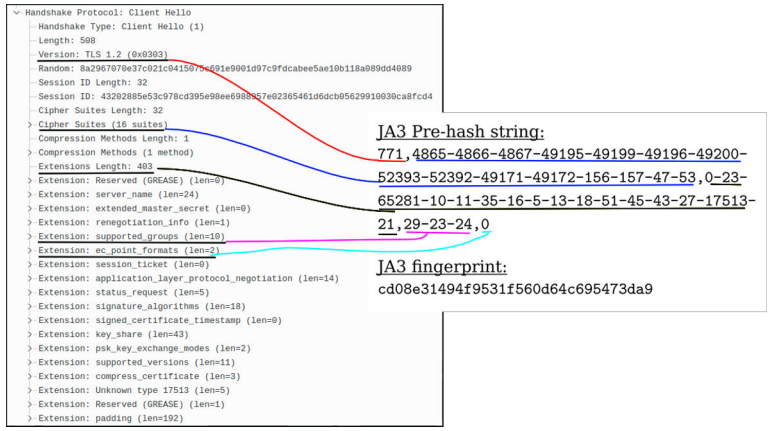

Figure 23. A screen capture from Wireshark showing a TLS Client Hello message and demonstrating how a JA3 hash fingerprint is formed, from publication IV

5.5.1 Contribution and Future Work

This publication brings light to the issue with the final step of the hash fingerprinting algorithms: taking the actual MD5 hash of the pre-hash string, which contains a lot of valuable information about the connection. It provides an initial answer for **RQ2**. The publication suggests that the last step should be left out, and the pre-hash string should instead be stored and used for analysis. The publication suggests that the pre-hash strings could be used as a basis for future research, where machine learning techniques could be applied on the values contained in the pre-hash string.

5.6 Publication V: Categorizing TLS traffic based on JA3 pre-hash values

Publication V is a conference research paper. Building upon the arguments made in publication IV, publication V explores utilising two different machine learning methods, K-Means and LDA, on the pre-hash strings of the JA3 algorithm. The approach is conceptually different with these two algorithms, but both algorithms are widely used in their own field. Despite focusing on the JA3 algorithm, the methods used in this publication could be utilised with any of the hash fingerprinting algorithms.

With K-Means the distance between two pre-hash strings is calculated using the Levenshtein distance algorithm with small modifications made to it to suite the purpose. The process of calculating the modified Levenshtein distance between two pre-hash strings is presented in figure 24. Using this approach, the order in which the values are presented by the endpoint in the original handshake message, and thus in the pre-hash string, is significant. K-Means clustering is performed on the collected pre-hash strings to find clusters among the values, and each cluster is considered one

$$JA3_1 = A_1,B_1\text{-}B_2\text{-}B_3,C_1,D_1,E_1\text{-}E_2$$
$$JA3_2 = A_1,B_1\text{-}B_2,C_1,D_2,E_3$$

$$\begin{aligned} lev(\, JA3_1, JA3_2\,) = \; & lev(\, A_1, A_1\,) \\ & + lev(\, B_1\text{-}B_2\text{-}B_3,\, B_1\text{-}B_2\,) \\ & + lev(\, C_1, C_1) \\ & + lev(\, D_1, D_2\,) \\ & + lev(\, E_1\text{-}E_2, E_3\,) \\ & = 0 + 1 + 0 + 1 + 2 = \underline{4} \end{aligned}$$

Figure 24. Levenshtein distance between two JA3 pre-hash strings, as presented in publication V

category.

With LDA the approach is different. Each pre-hash string is considered a document, and each value listed in the pre-hash string is considered a word. The values in the pre-hash string are first separated into a bag-of-words model and topic modeling using LDA is performed on it. With LDA the order in which the values are presented in the pre-hash string does not matter. Each found topic is considered a category with this approach.

The models were built and validated on samples collected from a small, controlled leisure network over a period of three months. Both machine learning methods produced convincing results, but it was again visible that endpoint applications using the same underlying TLS library were categorized similarly by both methods. The Levenshtein distance algorithm was further utilised in a small experiment where parts of the pre-hash string were presented as two-dimensional Cartesian coordinates by calculating the distance from origo. This visualization was able to demonstrate that the clusters found using K-Means could be visualized in multidimensional space.

5.6.1 Contribution and Future Work

This publication explores using two different machine learning methods on the pre-hash strings of a hash fingerprinting algorithm. The publication demonstrates that both methods have potential in identifying the endpoint applications based on the pre-hash strings, providing an answer for research question **RQ2**. The experiment remains superficial due to the small sample size, and thus leaves room for a more extensive experiment.

5.7 Publication VI: JAPPI: An unsupervised endpoint application identification methodology for improved Zero Trust models, risk score calculations and threat detection

Publication VI is a journal article detailing a novel methodology for identifying endpoint applications from network traffic using unsupervised learning, and thus introducing endpoint awareness into a network security solution. This publication builds upon the findings published in publications IV and V and utilises the pre-hash strings of the hash fingerprinting algorithms instead of the original hash fingerprint. The publication details a methodology called *JAPPI*, which specifies steps for building and utilising a JAPPI model. A visualization of the methodology is presented in figure 25.

A Source is first established, containing a set of trusted endpoints with the ability to report pre-hash strings and endpoint application metadata about each new network connection initiated by the endpoint. This information is then collected into a database over a pre-determined period of time. After this time, a JAPPI model is formed by finding clusters among the collected pre-hash strings, using a suitable clustering algorithm. The paper uses two suitable algorithms, K-Means and DBSCAN, but does not leave out the possibility of using another algorithm.

After the clusters have been found, the collected endpoint application metadata will be used for adding labels to the clusters, providing information about the endpoint applications producing the pre-hash strings included in this cluster. A primary label can also be manually added to a cluster, for example indicating the underlying TLS library which all these endpoint applications share. One sample, a centroid, is selected from each cluster to represent the cluster. After this, the model can be used by a network security solution for categorizing other network traffic, which does not provide endpoint application metadata for each connection. When a new suitable network connection is seen, the pre-hash string should be calculated and its distance to the centroids should be calculated. When the closest centroid has been found, the associated cluster should be considered the identified JAPPI cluster for the connection. While the current JAPPI model is being utilised on the whole protected network, new data should constantly be collected from the Source, and the JAPPI model should periodically be updated based on the new data.

As part of publication VI, a larger scale experiment was performed to verify how well the methodology works in a real-life network environment. Two distinct network environments were selected for the experiment. One network was a corporate network, which contained a more controlled network profile. The other network consisted of traffic from student housing dormitories. A subset of endpoints in the corporate network was used as the Source because endpoint metadata was available from these endpoints. Several JAPPI models were constructed based on pre-hash

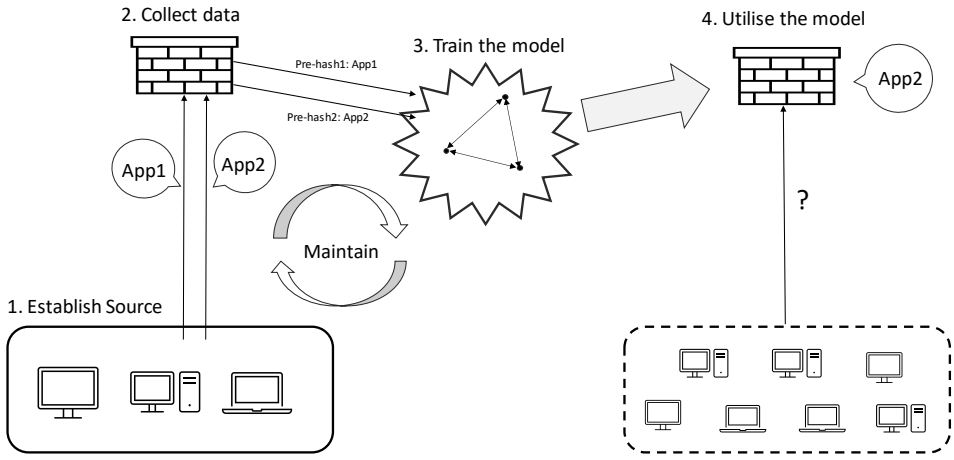

Figure 25. Visualization of the JAPPI methodology process from publication VI

strings and endpoint metadata collected from the Source using different clustering methods, and the models were applied both on the rest of the corporate network as well as on the student network. A model using our earlier method introduced in publication III was also constructed from the same Source, and the results of the different models were compared.

After publication V, the Chrome browser released an update to randomize the extensions in the TLS Client Hello message [2], which was raised as a potential risk for the hash fingerprinting algorithms in publication IV. The deterioration of the previous model, caused to a large extent by this update, was clearly visible in the results from our previous model, which was able to cover less than half a percent of the traffic. On the contrary, the best JAPPI model, produced using DBSCAN, was able to categorize over 99% of the corporate traffic with a *Strong* identification, meaning that the distance to the closest centroid was less than 10. In addition, when applied to the student network, the same model was able to categorize around 86% of the traffic with a Strong identification. When also considering the *Medium* identifications where the distance to the closest centroid was 10 or more but less than 20, the JAPPI model was able to categorize almost 94% of the student traffic.

5.7.1 Contribution and Future Work

The publication introduced a considerably improved methodology for bringing endpoint awareness to a network security solution. It also performed a larger scale experiment where its functionality was shown in two highly different network environments. Publication VI brings an answer for research question **RQ3**. The methodology has already been experimentally implemented in the Forcepoint Network Security Platform where its value has already been demonstrated with real traffic. The

next step for future research would be performing a long-term experiment to see how well the methodology can be maintained over a longer period of time. Another topic for future research would be to experiment how well the methodology performs in different types of network security solutions.

6 Conclusion

Introducing endpoint awareness into the inspection process of a network security solution can improve its functionality in many ways. If a network security solution knows one or both endpoint applications of a network connection it can use the information for making more accurate and better targeted security decisions. An accurate assessment of the endpoint application can be achieved using an external component on the protected endpoint which reports endpoint application information about each network connection to the network security solution. This is not, however, always possible to achieve in a protected network, which is why passive methods that do not need any interaction with the endpoint are needed. By taking advantage of both methods, it is possible to form a thorough system for gaining endpoint awareness in a network security solution.

There are multiple benefits for a network security solution in the ability to identify one or both endpoint applications of a network connection. One prominent benefit is the ability to produce more accurate security events. If the endpoint application receiving a network connection is known, and a potential security event is made from the connection, the network security solution can further consider whether it is relevant for the receiving endpoint application or not. On one hand, if the security event is not relevant, the network security solution can decide not to produce it, reducing false positive security events. On the other hand, if the security event would not have been produced otherwise, but it is considered highly relevant for this particular endpoint application, the network security solution can make the decision to exceptionally produce the security event, potentially catching an attack which would otherwise been left unnoticed, and thus reducing false negative security events.

Another benefit of having better visibility into the endpoint applications is the ability to raise or reduce the risk score of the network event. If the connection is initiated by a trusted endpoint application, the network security solution can decide to lower the risk score of the event or leave it as is. If, however, an unknown, unwanted, or otherwise unexpected endpoint application is seen initiating a network connection, the risk score of the network event can be raised.

Awareness of the endpoint applications can also help when building a Zero Trust architecture. It gives the network security solution an even more granular ability to restrict access to certain resources. This approach eliminates the risk of an unwanted application on an otherwise properly authenticated device making connections to

restricted resources and gives a network security solution the ability to only grant access for desired endpoint applications.

This dissertation addressed the issue of introducing endpoint awareness to a network security solution. A study was conducted where existing methods of bringing awareness of the endpoint to a network security solution were reviewed. This study laid groundwork for the following research, where a specific prominent passive identification method was identified and further improvements for it were offered and validated. Based on the findings, multiple methods of bringing better awareness of the endpoint and introducing endpoint awareness to the inspection process of a network security solution were proposed. The methods were validated in real life environments, and in some instances implemented as part of a commercial network security solution.

In this dissertation, the problem of gaining endpoint awareness in a network security solution was approached through the following three research questions:

RQ1 How can a network security solution gain endpoint awareness?

RQ2 In what ways could the deficiencies in existing methods for gaining endpoint awareness be addressed?

RQ3 How should endpoint awareness without active endpoint integration be implemented in a network security solution?

In this chapter, the results of the research are presented in a concise form. The overall contribution of the research regarding each research question is established, and the contribution of each publication for the research questions is clarified. The structure and contributions of the research is visualized in figure 26. In addition, we compare the research presented in this thesis to the related research presented in chapter 2. Finally, we discuss the limitations of this research, and go through the future work that this research introduces.

6.1 Results and Contributions

There are different ways of introducing endpoint awareness into a network security solution. As part of the research, a study was conducted on the different methods, listed in this thesis as publication II. In this study the pros and cons of each method were assessed. It was observed that it depends on the protected network whether an active or a passive method is more suitable. In the context of this research the passive methods were considered more beneficial. Out of the identified passive methods, the focus of the research centered on the hash fingerprinting algorithms that were considered potent and topical.

During this research, further analysis was conducted on the hash fingerprinting algorithms. It was determined that the final step of the hash fingerprinting algorithms,

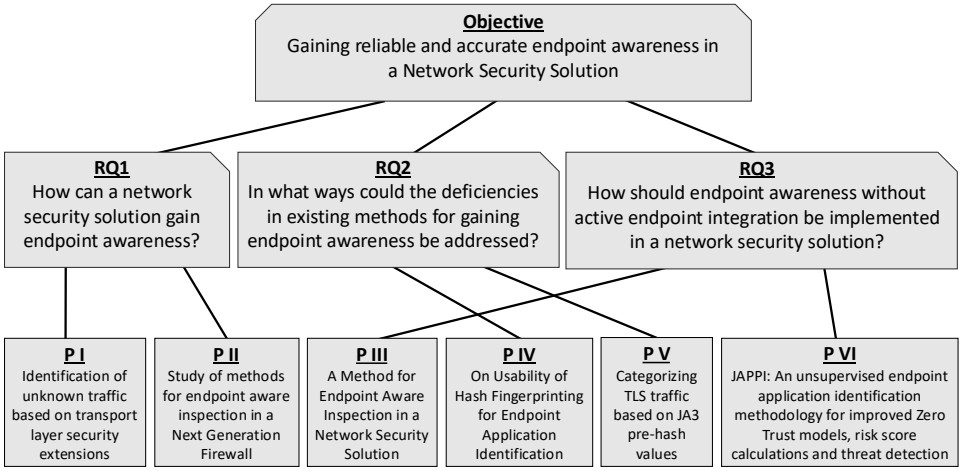

Figure 26. Structure of the research in this dissertation

the process of taking an MD5 hash of the collected pre-hash string, was unnecessary and in general made the algorithms less useful. This analysis was published in a conference paper, listed in this thesis as publication IV. Due to this assessment, the pre-hash string of the hash fingerprinting algorithms was taken into further consideration. The pre-hash string consists of a list of values for parameters presented during the handshake process of the particular protocol that are considered significant. Depending on the protocol, some parameters may list several values while others list only one. This information is presented in the pre-hash string in a standardized way. As the pre-hash string lists significant and distinct information about the handshake process, which can vary depending on the underlying protocol implementation, it was determined that this string provides a fruitful source for experimenting with various machine learning methods to gain better awareness of the endpoint.

Two distinct machine learning methods were experimented with for the purpose of identifying the underlying endpoint application based on the pre-hash string of a hash fingerprinting algorithm. This experiment was published in a conference paper, listed in this thesis as publication V. The two methods experimented with were a clustering algorithm, K-Means, and a topic modeling method, LDA. Both methods produced promising results when run on a sample data set.

Two methodologies for introducing endpoint awareness into the inspection process of a network security solution were proposed as part of this dissertation. The first methodology is listed in this thesis as publication III. It utilises the hash fingerprinting algorithms directly. The core of the methodology is establishing a trusted subset of the protected network which provides a good sample of the entire protected network. This subset is used for populating a database that contains information about which endpoint applications have been observed to produce which hash fin-

gerprints. This database is then later used on the entire protected network to gain awareness of the endpoint applications.

Following the first methodology, the fundamental issues with the hash fingerprinting algorithms were identified. After further research was conducted on the topic in publications IV and V, another methodology was proposed which utilised these findings. The second methodology is listed in this thesis as publication VI. This methodology is named JAPPI, for *JA3-based Application Identification*. JAPPI also makes use of a trusted subset of the protected network, where information about the pre-hash strings produced by different endpoint applications can be collected into a database. As opposed to the first methodology, this second methodology uses the database for constructing a machine learning model, using a suitable clustering algorithm such as K-Means or DBSCAN. This model is then applied on the entire protected network for gaining awareness of the endpoint applications. The JAPPI model will constantly be maintained using latest information collected from the trusted subset of the network.

During the dissertation process, a novel passive method for gaining additional information about the protected endpoint was also introduced. This method was published in the form of a patent, and listed in this thesis as publication I. This method provides a network security solution with the ability to better categorize observed network traffic based on the information listed in the extensions of a TLS handshake. This information can, as an example, be used for separating web traffic from other traffic using the TLS protocol, providing the ability to perform certain security procedures only on a suitable subset of the observed traffic.

Several concepts discovered during this dissertation have already been experimented with or implemented in the Forcepoint Network Security Platform. The patented method presented in publication I is actively used by the Forcepoint Network Security Platform for better categorization of otherwise uncategorized TLS traffic. In addition, the methodology presented in publication VI has been experimentally implemented in the Forcepoint Network Security Platform, and its value has been confirmed in several experiments. Thus, the research presented in this thesis has already provided solutions and concrete value for a real-life need.

In summary of the research and results presented in this thesis, all the research questions were successfully and comprehensively answered and the objective of the research was successfully reached. In the following sections, we will summarize the answers found to the research questions and conclude the thesis with discussing the limitations of the results and defining directions for potential future work.

6.1.1 RQ1: How can a network security solution gain endpoint awareness?

This research question was a fundamental question upon which the rest of the research was based on. To reach the overall objective of this dissertation, introducing endpoint awareness into the inspection process of a network security solution, it was crucial to begin by identifying existing methods and potentially finding new ones for gaining further awareness of the endpoint. During the research process there were two publications that specifically addressed this research question.

Publication I directly provided one answer to this question by presenting a novel way of gaining further awareness of the endpoint. The method involves analyzing the information stored in the extensions of a TLS handshake. One example of utilising this method is using the information stored in the ALPN and NPN extensions in the TLS Client and Server Hello messages. These extensions present information about the protocol which is transported inside the encrypted TLS layer. This information can further be used when determining whether the connection contains web traffic or something else, such as scripted API calls. This method has already been implemented into the Forcepoint Network Security Platform and is in active use.

A more extensive answer for this research question was given by publication II. This publication performed a thorough study of different methods that can be used for gaining endpoint awareness in a network security solution. It recognized that there are active and passive methods, and both have different uses. Active methods can be more useful in a firmly controlled network. An especially thorough awareness can be achieved if an external component can be installed on each protected endpoint which reports metadata about every new network connection to the network security device. This enables a network security solution to receive trusted and accurate information about the endpoint application. This level of control is, however, rarely possible in a protected network. Some endpoints in the protected network may be devices not controlled by the network administrator, and some endpoints may even have a rare operating system which cannot be monitored using the same tools. Passive methods, where no interaction is needed with the endpoint, provide a more generic solution. They can be applied on all network traffic. The downside with passive methods is that they are not as reliable as the most powerful active methods are, and they are often possible to evade by forming a protocol message which mimics the message structure of a different endpoint application.

6.1.2 RQ2: In what ways could the deficiencies in existing methods for gaining endpoint awareness be addressed?

After existing methods of gaining endpoint awareness had been identified, the second research question aspired to identify if there were ways they could be further

improved. This dissertation focused on the hash fingerprinting algorithms, and thus the focus for this research question was in finding ways to improve their usability. Two publications in this dissertation addressed this research question.

Publication IV highlighted a deficiency in the hash fingerprinting algorithms and proposed a solution for avoiding it. The final step of the hash fingerprinting algorithm, which produces the actual fingerprint, is taking an MD5 hash of the pre-hash string. The main deficiency that was highlighted is that the MD5 hash will obfuscate all information stored in the pre-hash string. The pre-hash string, on the other hand, provides visibility into key features of the connection as it lists features used during the connection and supported by the endpoint application. In addition, the information stored in the pre-hash string provides ground for further research, such as automatic endpoint application identification using machine learning. Due to these findings, this publication proposed that the pre-hash string should be used as the main fingerprint instead of the final MD5 hash. The publication included a short sample which enforced the claim.

After publication IV suggested the concept of using machine learning on the pre-hash strings for automatic endpoint application identification, publication V addressed this. This publication performed experiments using two distinct machine learning algorithms on the pre-hash strings. The chosen algorithms, K-Means and LDA, were different in how they approached the issue. With K-Means a distance algorithm, a modified version of the Levenshtein distance algorithm, was used for finding clusters among the collected values. With LDA, the approach was to consider each pre-hash string as a document, and each separate value included in the pre-hash string to be a word. LDA was then used for topic modeling on the values. A clear difference in these two approaches is that with K-Means the order in which the values are presented in the pre-hash string has significance, whereas the approach with LDA loses this information. Despite this difference, both approaches showed potential with the sample set used in this publication, demonstrating that the usage of the pre-hash string in collaboration with machine learning provides an improvement to the hash fingerprinting algorithms.

6.1.3 RQ3: How should endpoint awareness without active endpoint integration be implemented in a network security solution?

Once the existing methods for gaining endpoint awareness in a network security solution had been identified, the purpose of this research question was to identify ways how the information can best be utilised in a network security solution. The intention was to specify how a network security solution can receive accurate information about the endpoint applications, and how it can be utilised in the inspection process. During the dissertation process, two publications were written that addressed

the final research question.

After publication II, hash fingerprinting algorithms had been identified as the most prominent area for future research. Thus, a new methodology was formulated, which utilised them in gaining endpoint awareness in a network security solution. This methodology was presented in publication III. The methodology included several steps, during which a trusted source group was formed among the protected devices, a database containing mappings of hash fingerprint to endpoint application was populated, and the database was then utilised on the entire protected network for gaining endpoint awareness. In the light-weight case study performed as part of this publication it was deemed to perform well, and it showed promise in providing endpoint awareness to a network security solution. This publication also contained a small demonstration about how the methodology could improve accuracy of the inspection process of a network security solution by reducing false positive security events.

Following publication III, the deficiencies of the hash fingerprinting algorithms were identified and later realized by an update introduced by the Google Chrome browser, described in more detail in 5.7. Due to this, another methodology was introduced in publication VI, which addressed these deficiencies and utilised the findings of publications IV and V. This methodology was named JAPPI. Instead of using the hash fingerprints directly, JAPPI uses the pre-hash strings. Similarly, as with the previous methodology, a trusted source of endpoints is established and used for collecting mappings of pre-hash string to endpoint applications that have produced the string in the source group. Instead of directly using this mapping, JAPPI performs clustering on the pre-hash strings to find common clusters among the values. These clusters are then labeled based on the endpoint applications that produced pre-hash strings included in a cluster. One sample is then selected from each cluster, and these samples form a JAPPI model. When a network security solution utilises JAPPI on a new network connection, it first calculates a pre-hash string from the handshake, and then compares it to the samples in the JAPPI model. The closest one is considered the identified JAPPI cluster.

In publication VI the efficacy of the JAPPI methodology is also assessed in a larger scale experiment, and it is compared to the methodology presented in publication III which utilised the hash fingerprints directly. The experiment included two large network environments with different traffic profiles. In this experiment it was shown that JAPPI performed exceptionally well, while the previous methodology performed extremely poorly. Examples were also collected that showed how JAPPI would have improved inspection accuracy during the experiment period. This experiment validated that JAPPI provides a functional and efficient methodology for introducing endpoint awareness into a network security solution, without the requirement of active endpoint integration for all protected devices.

6.2 Comparison to Related Research

Chapter 2 introduced related research in the domain of bringing endpoint awareness into a network security solution. Although none of the related research works focused specifically on bringing endpoint awareness into the traffic inspection process of a network security solution, they used similar methods as the research presented in this thesis. In this section we will briefly cover the contribution of this dissertation as compared to the methods presented in chapter 2.

The research presented in this thesis focuses on hash fingerprinting algorithms, such as the JA3 algorithm for TLS clients. None of the related works used the same algorithms in their methods. All of them did, however, collect information from the traffic that is similar to what the JA3 algorithm collects. The advantage this dissertation has over the other methods is the use of a well-defined, widely used algorithm instead of a unique collection of various values. Implementing the proposed methodologies into a network security solution and repeating the experiments is considerably easier when a well-known algorithm is used.

Some researchers, such as Husák et al. [6] and Frolov and Wustrow [16], use a static data structure for presenting the collected data, similar to the MD5 based hash value that is used in the hash fingerprinting algorithms. This introduces the same problems that were brought forward in publication IV regarding the hash fingerprinting algorithms. On the other hand, Anderson and McGrew [17] use a data structure more closely resembling the pre-hash string used in the research presented in this thesis. The disadvantage of their approach is that they note that their fingerprinting syntax may change if the open-source code base is updated. This approach would most likely lead to additional updates into the code base of the network security solution to support the updated syntax. In addition, as the fingerprints may differ from version to version, backward compatibility with fingerprints collected using the previous version would be lost. In comparison, the JA3 fingerprinting algorithm is well defined and not subject to change, making it more suitable in the context of a network security solution.

Several methods collect information from both client and server side of the connection, as well as information from the entire duration of the connection, such as traffic bursts and packet sizes. This results in a method which can be useful for a network security solution for reporting purposes. However, since the identification can only be done either when the connection has been ongoing for a while or after the connection has already been closed, the usefulness of such a method for a network security solution is limited. In comparison, the hash fingerprinting algorithms used in this dissertation collect information from the handshake stage of the connection, often from the very first data packet that is sent in the connection. This provides a network security solution with the advantage of having the information practically for the entire duration of the connection, even offering the ability to end

the connection before any data has been transferred between the endpoints.

6.3 Limitations and Future Work

This dissertation has certain limitations that introduce potent areas for future research. One aspect limiting the research is that all experiments have been performed using the same type of network security solution, which is the Forcepoint Network Security Platform, previously also known as Forcepoint NGFW and Forcepoint SD-WAN. This limitation was introduced by the ease of access to this type of network security solution during the research process due to the author's affiliation with Forcepoint. Experimental implementation additions were possible for the Forcepoint Network Security Platform, and access to sample networks where this type of device had already been installed was available. There is, however, nothing specific to Forcepoint in the concepts or implementations brought forward during the research. As a future research project, experiments with different types of network security solutions would be considered valuable.

In addition, the experiments focused on the JA3 hash fingerprinting algorithm and TLS traffic, despite the research highlighting the hash fingerprints in general. In addition to the JA3 algorithm for TLS traffic, several other hash fingerprinting algorithms have been proposed for other protocols, as described in 4.3.2. This limitation was also introduced due to ease of access, as the Forcepoint Network Security Platform had integrated support for the JA3 algorithm. A valuable topic for future research would be validating the efficacy of JAPPI with other network protocols as well.

In publication II, multiple techniques of gaining endpoint awareness in a network security solution were introduced. This dissertation considered the hash fingerprinting algorithms the most potent method and focused on them. Nevertheless, future research on the applicability of the other identified methods on a network security solution could produce different results than this research.

Publication V chose two distinct machine learning methods, K-Means and LDA, for performing categorization on the pre-hash strings. Despite these algorithms being chosen, other potential methods for approaching this issue exist. Performing JAPPI with other machine learning methods could provide interesting results.

All real-life experiments conducted during this research were conducted over a fairly brief period of time. Thus, none of them were able to show how well the proposed methodologies function over a longer period, such as a year or more. Future research about the longevity of the proposed solutions would be valuable. A special interest is in the JAPPI methodology, where the JAPPI model is trained using information collected from the Source. When a long time passes, endpoint application implementations may go through large refactoring procedures, where the behavior of the application changes drastically. The current experiments were not able to cap-

ture such a situation, and thus it is not clear how the methodology behaves when this happens.

This dissertation focuses on the improvements that can be gained in a network security solution by having additional context about the endpoint. The collaboration of endpoint and a network security solution could also be approached from the perspective of a security solution residing in the endpoint. The potential benefits that can be achieved in an endpoint security solution from receiving additional context from a network security solution could be an interesting topic for additional research.

This research references passive methods that are performed on data received over a longer period of time. Some earlier research has experimented on using machine learning with network traffic patterns like bursts and packet sizes. Such methods are, however, only touched briefly, and the focus is instead moved towards methods that provide information about the endpoint already from the first few packets of a connection. This decision was made due to the benefits received from an early identification, such as the ability to reroute the connection and, in case the connection is terminated, a minimal amount of information about the endpoint is leaked with an early identification. But there can be a lot of value in information received after post processing. Future research on what kind of additional context can be received from post processing logs collected by a network security solution could introduce new powerful ways of providing additional security.

The methods used in this research focus on the application layer of the network protocol stack. The reason is the focus on identifying the endpoint applications themselves. The lower-level information, such as the TCP/IP information, could be utilised as well to provide additional information about the operating system with OS fingerprinting. Combining OS fingerprinting with the application layer methods can give an even better context about the protected endpoint, and thus further research on this would be warranted. The TCP/IP level information may, however, be lost on the route, for example if the network security solution itself resides in the cloud and there are other devices in between it and the endpoint that have their own TCP/IP stack implementation.

List of References

[1] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Active Internet-Draft, Version 22, Internet Engineering Task Force (IETF), 2024.

[2] Google Inc. Feature: TLS ClientHello extension permutation. https://chromestatus.com/feature/5124606246518784, 2023. Accessed 11 September 2023.

[3] Roy Fielding and Julian Reschke. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content, RFC 7231, Section 5.5.3: User-Agent. Internet Engineering Task Force (IETF), 2014.

[4] Microsoft Corporation. Microsoft Edge Documentation. https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide/emulation. Accessed 1 January 2020.

[5] Google. Override The User Agent String From Chrome DevTools. https://developers.google.com/web/tools/chrome-devtools/device-mode/override-user-agent. Accessed 1 January 2020.

[6] Martin Husák, Milan Čermák, Tomáš Jirsík, and Pavel Čeleda. HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting. *EURASIP Journal on Information Security*, 2016(1):1–14, 2016.

[7] Jonathan Muehlstein, Yehonatan Zion, Maor Bahumi, Itay Kirshenboim, Ran Dubin, Amit Dvir, and Ofir Pele. Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application. In *Proceedings of the 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6, Las Vegas, NV, USA, 2017.

[8] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. AppScanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 439–454, Saarbrücken, Germany, 2016.

[9] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. Robust Smartphone App Identification via Encrypted Network Traffic Analysis. *IEEE Transactions on Information Forensics and Security*, 13(1):63–78, 2018.

[10] Lee Brotherston. Stealthier attacks and smarter defending with TLS fingerprinting. DerbyCon V, Louisville, Kentucky, USA, 2015.

[11] John Althouse, Jeff Atkinson, and Josh Atkins. Open Sourcing JA3. https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41, 2017. Accessed 2 April 2024.

[12] Ben Reardon. Open sourcing HASSH. https://engineering.salesforce.com/open-sourcing-hassh-abed3ae5044c, 2018. Accessed 3 April 2024.

[13] Caleb Yu. GQUIC Protocol Analysis and Fingerprinting in Zeek. https://engineering.salesforce.com/gquic-protocol-analysis-and-fingerprinting-in-zeek-a4178855d75f, 2019. Accessed 3 April 2024.

[14] Adel Karimishiraz. RDP fingerprinting. https://medium.com/@0x4d31/rdp-client-fingerprinting-9e7ac219f7f4, 2019. Accessed 3 April 2024.

[15] Michael R. Torres. SMBFP SMB Fingerprinting Zeek package. https://github.com/micrictor/smbfp, 2020. Accessed 3 April 2024.

[16] Sergey Frolov and Eric Wustrow. The use of TLS in Censorship Circumvention. In *Network and Distributed Systems Security (NDSS) Symposium, San Diego, CA, USA*, 2019.

[17] Blake Anderson and David McGrew. TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior. pages 379—392. Association for Computing Machinery, 2019.

[18] Cisco Systems, Inc. Encrypted Visibility Engine. https://secure.cisco.com/secure-firewall/docs/encrypted-visibility-engine, 2023. Accessed 11 September 2023.

[19] John Grady. Cloud-native firewalls are the next step in network security. https://www.techtarget.com/searchsecurity/opinion/Cloud-native-firewalls-the-next-step-in-network-security, 2023. Accessed 15 March 2024.

[20] Cynet Security, LTD. Endpoint Detection and Response (EDR) Tools: How They Work and Solutions to Know. https://www.cynet.com/endpoint-protection-and-edr/top-6-edr-tools-compared/, 2024. Accessed 15 March 2024.

[21] Musarubra US LLC. What Is Endpoint Detection and Response? https://www.trellix.com/security-awareness/endpoint/what-is-endpoint-detection-and-response/, 2024. Accessed 15 March 2024.

[22] Gavin Wright and Alexander S. Gillis. Definition: endpoint detection and response (EDR). https://www.techtarget.com/searchsecurity/definition/endpoint-detection-and-response-EDR, 2024. Accessed 15 March 2024.

[23] Alexander Harrison. Anyconnect swg agent and the external domains list (swg bypass). https://support.umbrella.com/hc/en-us/articles/360045222232-AnyConnect-SWG-Agent-and-the-External-Domains-List-SWG-Bypass, 2020. Accessed 15 March 2024.

[24] Forcepoint LLC. Deploying smartedge agent. https://help.forcepoint.com/fpone/deploy/rhtml/guid-9312b7e9-5a8d-4e27-9e25-dcac7707c30e.html, 2024. Accessed 15 March 2024.

[25] Kenneth Ingham and Stephanie Forrest. A history and survey of network firewalls. Technical report, University of New Mexico, 2002.

[26] Thimothy Lightoler. *The Gentleman and Farmer's Architect: A New Work; Containing a Great Variety of Useful and Genteel Designs; Being Correct Plans and Elevations Of Parsonage and Farm Houses...* Robert Sayer, 1774.

[27] Harry C. Forsdlck and David P. Reed. Patterns of security violations: Multiple references to arguments. https://people.csail.mit.edu/saltzer/Multics/MHP-Saltzer-060508/bookcases/RFCs/csr-rfc-059.pdf, 1974. Accessed 18 March 2024.

[28] Jacob Boren. 10 times '80s sci-fi movies predicted the future. https://screenrant.com/80s-sci-fi-movies-predicted-the-future/, 2019. Accessed 15 March 2024.

[29] Marcus J Ranum. A network firewall. 1992.

[30] D Brent Chapman. Network (in) security through ip packet filtering. In *USENIX Summer*, volume 21, 1992.

[31] Steve Crocker, Dr. David D. Clark, Robert T. Braden, and Christian Huitema. Report of IAB Workshop on Security in the Internet Architecture - February 8-10, 1994. RFC 1636, June 1994. URL https://www.rfc-editor.org/info/rfc1636.

[32] J. Mogul, R. Rashid, and M. Accetta. The packer filter: an efficient mechanism for user-level network code. *SIGOPS Oper. Syst. Rev.*, 21(5):39–51, nov 1987. ISSN 0163-5980. doi: 10.1145/37499.37505. URL https://doi.org/10.1145/37499.37505.

[33] Cisco Documentation. Evolution of the firewall industry. https://docstore.mik.ua/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ch3.pdf. Accessed 18 March 2024.

[34] Jody Brazil. A Practical History of the Firewall – part 4: The Next Generation. https://www.linkedin.com/pulse/practical-history-firewall-part-4-next-generation-jody-brazil. Accessed 18 March 2024.

[35] Olli-Pekka Niemi and Antti Levomäki. Evading deep inspection for fun and shell. *Black Hat USA*, 2013:1–5, 2013.

[36] Google. HTTPS encryption on the web - Google Transparency Report. https://transparencyreport.google.com/https/overview, 2024. Accessed 18 March 2024.

[37] Forcepoint. What is an Intrusion Prevention System (IPS)? Intrusion Prevention Systems (IPS) Defined. https://www.forcepoint.com/cyber-edu/intrusion-prevention-system-ips, 2024. Accessed 19 March 2024.

[38] Palo Alto Networks. What is an Intrusion Prevention System? https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-prevention-system-ips, 2024. Accessed 19 March 2024.
[39] Inc. Fortinet. What Is Intrusion Prevention System? Definition and Types. https://www.fortinet.com/resources/cyberglossary/what-is-an-ips, 2024. Accessed 19 March 2024.
[40] Forcepoint. What Are SASE and VPN? SASE and VPN Defined, Explained and Explored. https://www.forcepoint.com/cyber-edu/sase-and-vpn, 2024. Accessed 19 March 2024.
[41] Inc. Fortinet. What Is A VPN? https://www.fortinet.com/resources/cyberglossary/what-is-a-vpn, 2024. Accessed 19 March 2024.
[42] Nord Security. What is a VPN? https://nordvpn.com/what-is-a-vpn/, 2024. Accessed 19 March 2024.
[43] Forcepoint. What is Sandbox Security? Sandbox Security Defined, Explained, and Explored. https://www.forcepoint.com/cyber-edu/sandbox-security, 2024. Accessed 19 March 2024.
[44] Check Point Software Technologies Ltd. What Is Sandboxing? https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-sandboxing/, 2024. Accessed 19 March 2024.
[45] Inc. Fortinet. What Is Sandboxing? https://www.fortinet.com/resources/cyberglossary/what-is-sandboxing, 2024. Accessed 19 March 2024.
[46] Forcepoint. What is data loss prevention? data loss prevention defined, explained, and explored. https://www.forcepoint.com/cyber-edu/data-loss-prevention, 2024. Accessed 21 March 2024.
[47] Microsoft. What is data loss prevention (dlp)? https://www.microsoft.com/en-us/security/business/security-101/what-is-data-loss-prevention-dlp, 2024. Accessed 21 March 2024.
[48] Inc. Fortinet. What is data loss prevention (dlp)? https://www.fortinet.com/resources/cyberglossary/dlp, 2024. Accessed 21 March 2024.
[49] Forcepoint. What is SD-WAN? SD-WAN Defined, Explained, and Explored. https://www.forcepoint.com/cyber-edu/what-is-sd-wan-software-defined-wide-area-networking, 2024. Accessed 19 March 2024.
[50] Inc. Cisco Systems. What Is SD-WAN? https://www.cisco.com/c/en/us/solutions/enterprise-networks/sd-wan/what-is-sd-wan.html, 2024. Accessed 19 March 2024.
[51] Inc. Fortinet. What Is SD-WAN? https://www.fortinet.com/resources/cyberglossary/sd-wan-explained, 2024. Accessed 19 March 2024.
[52] Forcepoint. What is a Secure Web Gateway (SWG)? https://www.forcepoint.com/cyber-edu/what-is-secure-web-gateway, 2024. Accessed 19 March 2024.
[53] Check Point Software Technologies Ltd. What is a Secure Web Gateway (SWG)? https://www.checkpoint.com/cyber-hub/network-security/what-is-secure-web-gateway/, 2024. Accessed 19 March 2024.
[54] Palo Alto Networks. What Is a Secure Web Gateway (SWG)? https://www.paloaltonetworks.com/cyberpedia/what-is-secure-web-gateway, 2024. Accessed 19 March 2024.
[55] Forcepoint. What Is a CASB Service? CASB Services Defined, Explained and Explored. https://www.forcepoint.com/cyber-edu/casb-service, 2024. Accessed 19 March 2024.
[56] Check Point Software Technologies Ltd. What is a Cloud Access Security Broker (CASB)? https://www.checkpoint.com/cyber-hub/cloud-security/what-is-casb/, 2024. Accessed 19 March 2024.
[57] Inc. Gartner. Cloud Access Security Brokers (CASBs). https://www.gartner.com/en/information-technology/glossary/cloud-access-security-brokers-casbs, 2024. Accessed 19 March 2024.
[58] Palo Alto Networks. What is firewall as a service? https://www.paloaltonetworks.com/cyberpedia/what-is-firewall-as-a-service, 2024. Accessed 21 March 2024.
[59] Inc. Fortinet. Firewall as a service (fwaas). https://www.fortinet.com/resources/cyberglossary/firewall-as-a-service-fwaas, 2024. Accessed 21 March 2024.
[60] Check Point Software Technologies Ltd. What is firewall as a service (fwaas)? https://www.checkpoint.com/cyber-hub/network-security/firewall-as-a-service-fwaas/, 2024. Accessed 21 March 2024.

[61] John Kindervag. Build Security Into Your Network's DNA: The Zero Trust Network Architecture. https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf, 2010. Accessed 19 March 2024.

[62] Forcepoint. What is Zero Trust Network Access (ZTNA)? ZTNA Defined, Explained and Explored. https://www.forcepoint.com/cyber-edu/zero-trust-network-access-ztna, 2024. Accessed 20 March 2024.

[63] Inc. Fortinet. Zero Trust Network Access (ZTNA). https://www.fortinet.com/solutions/enterprise-midsize-business/network-access/application-access, 2024. Accessed 20 March 2024.

[64] Palo Alto Networks. What Is Zero Trust Network Access (ZTNA). https://www.paloaltonetworks.com/cyberpedia/what-is-zero-trust-network-access-ztna, 2024. Accessed 20 March 2024.

[65] Forcepoint. What Is a Zero Trust System? Zero Trust Systems Defined, Explained and Explored. https://www.forcepoint.com/cyber-edu/zero-trust-system, 2024. Accessed 20 March 2024.

[66] Microsoft. What is Zero Trust? https://learn.microsoft.com/en-us/security/zero-trust/zero-trust-overview, 2024. Accessed 20 March 2024.

[67] Palo Alto Networks. What is a Zero Trust Architecture. https://www.paloaltonetworks.com/cyberpedia/what-is-a-zero-trust-architecture, 2024. Accessed 20 March 2024.

[68] Neil MacDonald, Lawrence Orans, and Joe Skorupa. The future of network security is in the cloud. https://www.gartner.com/en/documents/3957375, 2019. Accessed 21 March 2024.

[69] Inc. Fortinet. Secure access service edge. https://www.fortinet.com/products/sase, 2024. Accessed 21 March 2024.

[70] Forcepoint. Data-first sase from forcepoint. https://www.forcepoint.com/use-case/sase-secure-access-service-edge, 2024. Accessed 21 March 2024.

[71] Palo Alto Networks. Prisma sase. https://www.paloaltonetworks.com/sase, 2024. Accessed 21 March 2024.

[72] Neil MacDonald, Lawrence Orans, and Joe Skorupa. 2021 strategic roadmap for sase convergence. https://www.gartner.com/en/documents/3999828, 2021. Accessed 21 March 2024.

[73] Inc. Gartner. Security service edge (sse). https://www.gartner.com/en/information-technology/glossary/security-service-edge-sse, 2024. Accessed 21 March 2024.

[74] Forcepoint. What is sse? a complete guide to security service edge. https://www.forcepoint.com/cyber-edu/security-service-edge-sse, 2024. Accessed 21 March 2024.

[75] Palo Alto Networks. What is sse? — security service edge (sse). https://www.paloaltonetworks.com/cyberpedia/what-is-security-service-edge-sse, 2024. Accessed 21 March 2024.

[76] Palo Alto Networks. App-id. https://www.paloaltonetworks.com/technologies/app-id, 2024. Accessed 15 August 2024.

[77] Forcepoint LLC. Getting started with network application elements. https://help.forcepoint.com/flexedge/sd-wan/en-us/7.2.0/onlinehelp/GUID-BE114C51-27A3-474E-A987-658F66D6258B.html, 2024. Accessed 15 August 2024.

[78] Check Point Software Technologies Ltd. Application control. https://www.checkpoint.com/quantum/application-control/, 2024. Accessed 15 August 2024.

[79] Inc Zoom Video Communications. Zoom: One platform to connect. https://otx.alienvault.com/browse/global/indicators, 2024. Accessed 8 August 2024.

[80] Roman Kunicher. Is it possible to block by user agent or client type? https://community.checkpoint.com/t5/Threat-Prevention/Is-it-possible-to-block-by-User-Agent-or-Client-Type/m-p/91039/highlight/true#M2763, 2024. Accessed 28 March 2024.

[81] Inc Fortinet. User agents. https://help.fortinet.com/fsiem/5-1-0/Online-Help/HTML5_Help/User_Agents.htm, 2024. Accessed 28 March 2024.

[82] Forcepoint. Detected vulnerabilities and situations in sgpkg-ips-1708-5242. https://autoupdate.ngfw.forcepoint.com/download/dynup/sgpkg-1708-SUMMARY.html, 2024. Accessed 28 March 2024.

[83] Forcepoint. Tls-ja3_chrome. https://autoupdate.ngfw.forcepoint.com/download/dynup/sgpkg-1708-SUMMARY.html#SID-TLS-Client-Hello-JA3-Hash-TLS-JA3-Chrome, 2024. Accessed 28 March 2024.

[84] Uli Heilmeier. Commit f18ee30a3dc8495a3913043aa64c506d3e92fe13: 'TLS: Adding JA3 and JA3S fingerprints'. https://github.com/wireshark/wireshark/commit/f18ee30a3dc8495a3913043aa64c506d3e92fe13, 2021. Accessed 13 September 2023.

[85] Mikhail Golovanov. Ja3.zone. https://ja3.zone/, 2024. Accessed 2 April 2024.

[86] The Alien Labs. Indicators search. https://otx.alienvault.com/browse/global/indicators, 2024. Accessed 2 April 2024.

[87] John Althouse, Jeff Atkinson, and Josh Atkins. TLS Fingerprinting with JA3 and JA3S. https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967, 2019. Accessed 3 April 2024.

[88] Jana Iyengar and Martin Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000, May 2021. URL https://www.rfc-editor.org/info/rfc9000.

[89] Fatema Bannatwala. KYD - Know your devices, a method for profiling devices using DHCP. https://github.com/fatemabw/kyd, 2019. Accessed 3 April 2024.

[90] McAfee Corp. Endpoint Intelligence Agent. https://docs.trellix.com/bundle/endpoint-intelligence-agent-v3-2-1-initNull-product/resource/prod-endpoint-intelligence-agent-v3-2-1-cat-product.pdf, 2019. Accessed 3 April 2024.

[91] Forcepoint LLC. Forcepoint One Endpoint and how it works. https://help.stonesoft.com/onlinehelp/StoneGate/SMC/6.10.0/GUID-A392A75D-7EBD-462D-A6FD-0E6F85E533B6.html, 2023. Accessed 3 April 2024.

[92] Gordon Lyon. Nmap Network Scanning. https://nmap.org/, 2023. Accessed 3 April 2024.

[93] Vahan Petrosyan. How to change user agents in chrome, edge, safari & firefox. https://www.searchenginejournal.com/change-user-agent/368448/, 2024. Accessed 28 March 2024.

[94] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Technical report, Internet Engineering Task Force, September 2024. Work in Progress.

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Painosalama, Turku, Finland 2025

ISBN 978-952-02-0084-8 (PRINT)
ISBN 978-952-02-0085-5 (PDF)
ISSN 2736-9390 (PRINT)
ISSN 2736-9684 (ONLINE)