EUROPEAN UNION AGENCY
FOR CYBERSECURITY

ENCRYPTED TRAFFIC ANALYSIS

Use Cases & Security Challenges

NOVEMBER 2019

ABOUT ENISA

The mission of the European Union Agency for Cybersecurity (ENISA) is to achieve a high common level of cybersecurity across the Union, by actively supporting Member States, Union institutions, bodies, offices and agencies in improving cybersecurity. We contribute to policy development and implementation, support capacity building and preparedness, facilitate operational cooperation at Union level, enhance the trustworthiness of ICT products, services and processes by rolling out cybersecurity certification schemes, enable knowledge sharing, research, innovation and awareness building, whilst developing cross-border communities. Our goal is to strengthen trust in the connected economy, boost resilience of the Union's infrastructure and services and keep our society cyber secure. More information about ENISA and its work can be found www.enisa.europa.eu.

CONTACT

For contacting the authors please use cod3@enisa.europa.eu.
For media enquiries about this paper, please use press@enisa.europa.eu.

AUTHORS

Paraskevi Dimou	ENISA
Jan Fajfer	Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University in Prague
Nicolas Müller	Fraunhofer Institute for Applied and Integrated Security
Eva Papadogiannaki	Foundation for Research and Technology, Greece
Evangelos Rekleitis	ENISA
František Střasák	Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University in Prague

ACKNOWLEDGEMENTS

Konstantin Böttinger	Fraunhofer Institute for Applied and Integrated Security
Sebastian Garcia	Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University in Prague
Ifigeneia Lella	ENISA
Veronica Valeros	Artificial Intelligence Center, Faculty of Electrical Engineering, Czech Technical University in Prague

LEGAL NOTICE

COPYRIGHT NOTICE

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Recent studies by browser vendors indicate that the usage of encrypted HTTPS traffic is increasing and it is currently around 70-90% of loaded HTTPS web pages. From an end user privacy perspective these are good news. Encryption protocols; such as Transport Layer Security, TLS 1.2 & TLS 1.3, provide security guarantees for data confidentiality and integrity and one would assume that security hygiene as a whole is raised. Unfortunately, the actual picture is more nuanced. Beyond the unavoidable risks posed by new cryptanalytic attacks and Quantum computing, there are many issues at hand that impact information security. For example, current studies show that many developers fail to properly implement the employed encryption schemes in their sites or applications, leading to weak encryption and providing users a false sense of security. In contrast, attackers have proved very capable in using encryption and cryptographic techniques in their attacks; instances ranging from ransomware to employing HTTPS to protect communication with infected devices and avoid detection.

In light of these developments, it becomes evident that the efficacy of rule-based monitoring and detection controls; e.g. Application firewalls, Intrusion Detection and Prevention Systems (IDPS), Data Loss Prevention\Protection (DLP) tools etc., which to a great extent relies on having access to the unencrypted traffic, is negatively affected. Organizations relying on such controls for their information security lose valuable insight and end up having blind spots in their managed infrastructure. To counter this vulnerability many organizations try to break end-to-end encryption by installing TLS inspection solutions (aka SSL/TLS proxies, middleboxes etc.) in key points of their network. This solution allows for the decryption of the encrypted traffic, to provide access to the plaintext payload for network monitoring and analysis tools, before re-encrypting it and forwarding it to its destination. While this method reclaims [partial] control for the organization, it also affects negatively the privacy of end users and is not guaranteed to detect all malicious traffic, which might employ custom or non-TLS encryption protocols to avoid detection. An alternative, and in our view complimentary, solution would be to employ Machine Learning (ML) and Artificial Intelligence (AI) techniques to perform encrypted traffic analysis.

This report explores the current state of affairs in Encrypted Traffic Analysis and in particular discusses research and methods in 6 key use cases; viz. application identification, network analytics, user information identification, detection of encrypted malware, file/device/website/location fingerprinting and DNS tunnelling detection. In the majority of use cases discussed, proposed techniques manage to undo many of the security assumptions users have when using encryption, lowering (or more correctly readjusting) privacy expectations. For example, application protocols such as DNS, FTP, HTTP, IRC, LIMEWIRE etc. may be distinguished using feature-based machine learning with reasonable accuracy, while observing certain properties of the encrypted data, it is possible to create data records which map these properties to the corresponding files or websites, a concept known as fingerprinting, which provides ways to infer which web pages, file, songs or videos are requested by a user, even if this traffic is encrypted. On the other hand, such techniques cannot offer the same level of insight as normal monitoring and analysis of unencrypted data a gap that additional research effort is trying to close as much as possible.

In addition, the report discusses recent research in TLS practices identifying common improper practices and proposing simple yet efficient countermeasures like certificates validation and pinning, minimize exposed data over HTTP redirects, using proper private keys and the latest versions of TLS (i.e. 1.2 and 1.3), deprecating older ones and employing certificate signing and by a trusted CA. While the proposals might seem trivial and only geared towards unexperienced developers or small companies, in fact studies have shown that such issues are quite prevalent.

1. INTRODUCTION

The advent of network traffic encryption, such as TLS, has not only significantly improved security and user privacy, but also reduced the ability of network administrators to monitor their infrastructure for malicious traffic and sensitive data exfiltration. In unencrypted network traffic, an eavesdropper; whether malicious (i.e. an attacker), or not (e.g. a network administrator monitoring his/her infrastructure) can read network packets and easily view their contents. Traffic or packet encryption methods, such as TLS, IPSec etc., on the other hand, ensure that while an eavesdropper can still record packets, he can no longer decipher their content or modify them without detection. For this reason, many TLS users assume that their connection to a web server is not interpretable by third parties. However, this is only partly true, because modern *encrypted traffic analysis* (sometimes called *ETA*) is able to soften these confidentiality gains. Using state-of-the-art methods, an eavesdropper can read the following information from encrypted network traffic, which should be private thanks to TLS:

One should not assume that traffic encryption can provide 100% privacy protection against an eavesdropper.

- They can gain information about the presence or absence of applications installed on a user's smartphone,
- find out which websites a user is surfing to, even though the user employs both traffic encryption and an anonymity network such as TOR,
- find out which files a user downloads and shares over an encrypted channel,
- identify user actions in mobile applications and build a user profile

The privacy of Internet users is therefore largely threatened by encrypted traffic analysis, and expected privacy guarantees are no longer fully met. This does not mean that encryption s broken or that eavesdroppers have full access to the exchanged data, but rather that our traffic and leaks more information than we assumed possible.

On the other hand, encrypted traffic analysis can also be a useful tool for network administrators. Common, rule-based, monitoring and detection security controls; e.g. Intrusion Detection and Prevention System (IDPS), Application Firewalls, Data Loss Prevention\Protection (DLP), rely in payload analysis to detect malicious traffic; such as connections to Command & Control (C&C) servers, virus spreading or sensitive data exfiltration. But attackers have become proficient in using state-of-the-art, of the self or custom encryption schemes to bypass such controls[1,2]. To ensure security and reliability in corporate networks and IT infrastructure in general, encrypted traffic analysis can be used [to a certain extend] as follows:

- Administrators can identify what type of network traffic is present, which is helpful in load balancing and predicting required network capacity.
- AI enhanced intrusion detection systems can use encrypted traffic analysis to identify questionable network traffic.
- Sophisticated viruses and botnets employ encryption in their communication with Command and Control servers. With the help of encrypted traffic analysis, this communication can be identified, and appropriate countermeasures can be employed.

[1] MITRE ATT&CK technique: Standard Cryptographic Protocol, https://attack.mitre.org/techniques/T1032/, accessed November 2019
[2] MITRE ATT&CK technique: Custom Cryptographic Protocol, https://attack.mitre.org/techniques/T1034/, accessed November 2019

1.1 SCOPE AND OBJECTIVES

The scope of this report is to examine the main use cases of encrypted traffic analysis and analyse how network administrators can use it to extract useful information without access to the transferred plaintext payload. It also discusses improper practices in securing network traffic using TLS, as well as raises the key issue of malicious encrypted traffic detection.

In particular, the objective of the study is to present how encrypted traffic analysis can be a useful tool for network administrators and security practitioners, but also to identify and describe the most dangerous encrypted traffic analysis-based attack vectors. As is often the case in cyber security, a tool may be used with and without malicious intent. In this study, we aim to present both aspects of encrypted traffic analysis.

1.2 OUTLINE

The structure of the report is as follows:

- Chapter 2 discusses how encryption may negatively impact the security posture of an organization by limiting the efficacy of existing security controls.
- Chapter 3 explores the taxonomy of the approaches to infer information from encrypted packet flows.
- Chapter 4 presents the approaches for application classification in encrypted traffic.
- Chapter 5 proposes methods to analyse encrypted network traffic to identify user actions.
- Chapter 6 details threats to user privacy over an encrypted network.
- Chapter 7 describes the detection of encrypted malware traffic.
- Chapter 8 analyses ways to infer information, using “fingerprinting”.
- Chapter 9 discusses DNS tunnelling use case.
- Chapter 10 provides a summary of available encrypted traffic analysis techniques.
- Chapter 11 describes improper TLS practices and their impact.
- Chapter 12 provides a summarization of key findings.
- Annex A provides a reminder of the Open Systems Interconnection model (OSI model) and a very short introduction to Machine Learning.

2. CHALLENGES OF ENCRYPTION IN SECURITY

Encryption protocols can provide privacy and security to network communication. By far the most popular network traffic encryption protocol is the Transport Layer Security (TLS)[3]. The goal of this protocol is to provide three main features for every connection: *confidentiality*, *authentication* and *message integrity*. Confidentiality of a connection prevents anyone from reading the contents of the messages, ensuring users' privacy to a degree. Authentication, verifies the identity of the parties communicating. And Message Integrity, provides a way to verify that a message has not been modified on the way from the sender to the receiver.

SSL 2.0 was designed in 1995. However, due to vulnerabilities, a more secure SSL 3.0 was released a year later. It was widely used until 2014 – when a major security vulnerability found by Google security team. TLS emerged out of SSL technology and has overshadowed it.

Nevertheless, encryption of network traffic has its challenges. For example it hinders detection of infected devices in a network. When malware uses traffic encryption to hide its activity[1,2] it is much more difficult to detect this encrypted malicious traffic, as opposed to the detection of unencrypted malicious traffic. Other challenges include implementation errors or configuration mistakes when deploying encrypted traffic protocols, performance cost of traffic encryption and the impact that new technologies [will] have on encrypted traffic.

2.1 TRAFFIC ANALYSIS

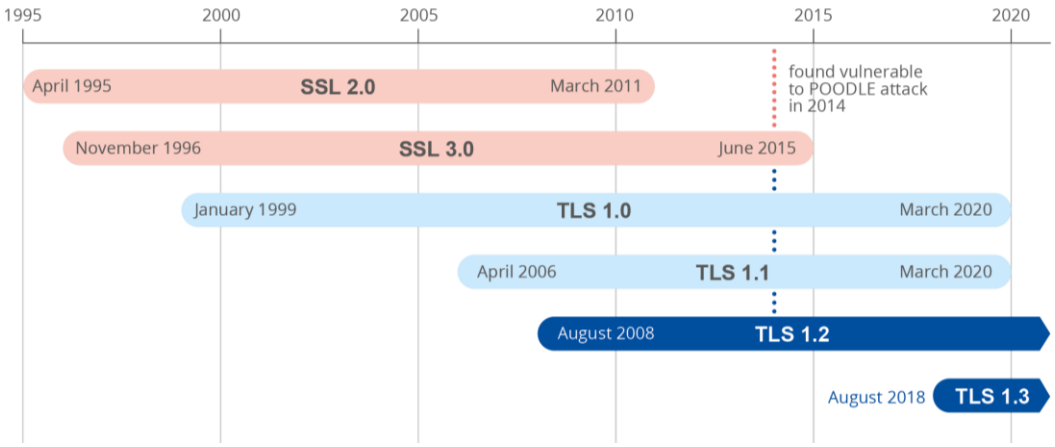
In general, encryption has a significant impact on detection and analysis of network traffic, because it hides all payload data. The fact that all data between a client and server can be encrypted practically and economically, is an important progress for our society; as it allows for secure [financial] transactions and private communications. On the other hand, encryption interferes with the efficacy of classical detection techniques. Finding suspicious patterns in the encrypted traffic is much more difficult. In addition, Data Loss Prevention\Protection solutions cannot monitor and protect against the unauthorized flow of sensitive data, since there is no visible pattern to identify them once encrypted. In fact, many malicious attacks take advantage of this, by encrypting data before exfiltrating them and/or encrypting communications between Command & Control and targeted systems. The obvious solution is to install TLS inspection opening the encrypted traffic with own TLS certificate and detect malicious behaviour in the decrypted traffic. However, this would break End-to-End encryption (or to be more accurate Client-to-Server encryption) placing trust to the TLS inspection box and all the inspection and analysis services accessing the plaintext traffic. Such a scenario might be undesirable, especially in instances were sensitive data are transmitted (e.g. financial transactions) and network monitoring had been outsourced to third parties (e.g. external Security Operations Center (SOC) teams, or subscription/cloud based security monitoring tools). Moreover, by unencrypting and examining payload the confidentiality of end users is not respected, especially in cases where users are unaware of this practice; either due to lack of technical knowledge or because their devices have been silently configured by administrators to trust intermediate certificates.

The main challenge today is to find a balance between end-to-end security, keeping the confidentiality of end users, and at the same time gathering valuable information from the traffic to detect possible threats and better allocate and protect resources. It is a difficult problem and it's important to identify alternative ways of detecting malicious behaviour that do not decrease the confidentiality of users. New technologies and algorithms of Artificial Intelligence and

[3] IETF News: TLS 1.3, https://ietf.org/blog/tls13/, accessed November 2019

Machine Learning could be a solution, but as we shall see this remains an active field of research.

Figure 1: Timeline of SSL/ TLS Protocols



2.2 DEPLOYING ENCRYPTION PROTOCOLS

Proper deployment of encryption technology in both client and server applications is another important challenge. Following good practices is essential in order to get all the expected security and privacy benefits that TLS provides. Mistakes, such as not verifying of the server's certificate or an insecure initiation of TLS communication, undermine the protocol's features and create easily exploited vulnerabilities. The fact that even with these omissions the protocol is still in place, creates a false sense of security. Since the network traffic is still encrypted, but easily observable or modifiable. This topic is addressed in a chapter 11.3 Proper TLS practices, which describes the impact of such faults.

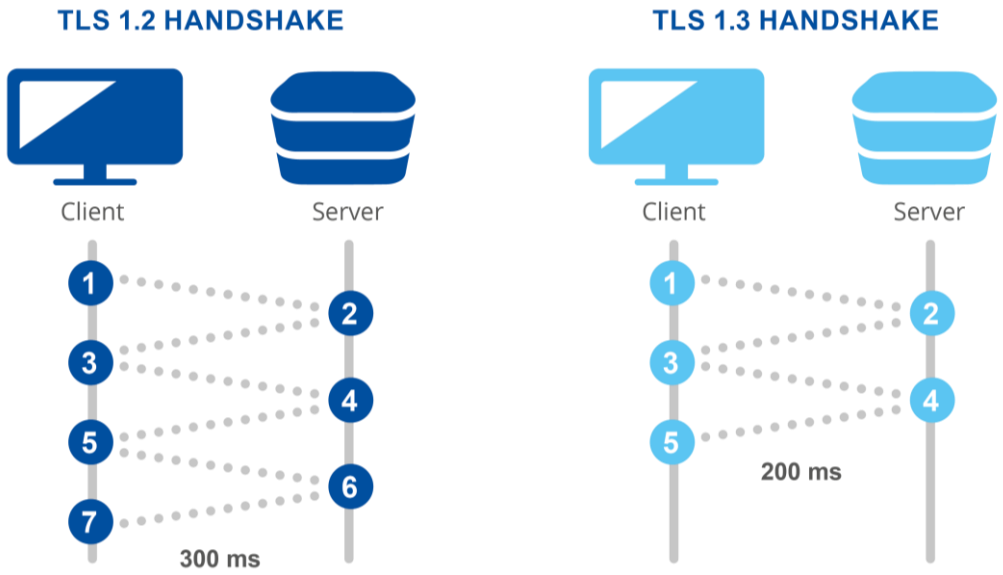
2.3 PERFORMANCE OF ENCRYPTION PROTOCOLS

Encrypting network traffic is an additional task, which increases time delay when communicating. The speed of such communication is therefore decreased. Using TLS and other security protocols also consumes computing power both for encryption and decryption during the handshake part. This motivates developers of applications to communicate data using HTTP and only check the integrity of the data transmitted; hopefully having first established that confidentiality is not of concern.

However, the use of HTTP is still a problem. Even when the integrity check is performed, there is usually important information about the given application revealed, which a potential attacker might use to attack the given device. To that extend, improvements have been made to the newest TLS version, TLS 1.3, which increase the speed of the TLS handshake. Not to mention that with new, faster and more powerful processing technologies, the performance impact should be negligible. In any case, the decrease in speed and the need for additional operations are well worth, considering the features that [a properly] TLS provides, and we could only envision conditions of resource restrained systems that might opt for speed-optimized alternatives.

In the figure dole there is comparison of handshakes between the most popular TLS 1.2 protocol and the latest TLS 1.3 protocol. As shown in Figure 2: Comparison of handshakes in TLS 1.2 and TLS 1.3, the handshake of TLS 1.3 is faster, and this slight improvement will have an essential impact for daily global internet traffic as it becomes more popular.

Figure 2: Comparison of handshakes in TLS 1.2 and TLS 1.3[4]



2.4 IMPACT OF NEW TECHNOLOGIES

The quick evolution of networking technologies has an impact on TLS and other encryption protocols. In the past, one of the reasons new encryption algorithms had to be introduced was the increase in computing power, which enabled for more effective attacks on short encryption keys. Similar scenarios can happen in the future, for example, a breakthrough invention in quantum computing would add unique possibilities in attacking TLS, which would force this protocol to adapt. It is dangerous not to pay attention to new trends and technologies as past solutions become vulnerable and easier to exploit. It is therefore essential to be able to stay up to date and adapt in this constantly evolving landscape.

To help mitigate these future threats, it is necessary for developers to be prepared to react quickly. This requires designing software that is adaptable to these changes and developing it in a way that enables addressing security incidents as fast as possible. Designing a software without considering future exploits and vulnerabilities, is dangerous and may compromise the security of the system. Security by design and Privacy by design should be augmented should incorporate preparedness and adaptability.

[4] Source: https://kinsta.com/blog/tls-1-3/, accessed November 2019

3. TAXONOMY OF ENCRYPTED TRAFFIC ANALYSIS

It is, often highly desirable to infer information such as application protocol, transferred files etc. from an encrypted packet flow. Using a variety of techniques, this is possible to some extent. In this section, we will give a short taxonomy of the approaches available. This taxonomy has been introduced by (Khalife, Hajjar, and Diaz-Verdejo 2014).

Encrypted Traffic Analysis may be characterized by the following properties.

1. Its *goals* and *purpose*.
2. The way it *extracts information* from the encrypted packet.
3. The way it *processes this information* to derive desired information.

In the following paragraph, we detail these properties.

1. ***Goals*.** There is no single goal of encrypted traffic analysis; rather, there are many different use cases, for example *Traffic Clustering*, *Application Type* and *Protocol Classification*, *Anomaly Detection* or *File Identification*.
2. ***Information extraction*.** There are several ways information is extracted from encrypted packets. Information may be extracted either by
    - observing behavioural properties (e.g. the round trip time, number of packets sent)
    - observing the encrypted payload itself
    - observing additional information such as protocol handshakes (e.g. TLS handshake)
3. ***Information processing*.** The information gained during information extraction has to be processed to derive the desired classification result. This processing can be very basic (by using heuristics, profiles or simple statistical means) or very complex (data-driven / machine-learning). Choosing the right processing depends a lot on the goal and the information available and is an active field of research in its own right. Considering the vast amount of existing algorithms, we can only present the general approach and defer the introduction of specific algorithms to the following sections, where we examine specific use cases of encrypted traffic analysis.

We will examine these methods in more detail in the following subsections. For now, we turn our attention to the process of feature extraction, a requirement for subsequent application of statistical methods.

Features are atomic observable properties of a given network flow or packet. In encrypted traffic, these features may be obtained from either the header of a packet, properties of the unencrypted handshake, by observing statistical properties such as the round-trip-time (RTT) or even from the encrypted data itself (via byte-patterns).

3.1 FEATURE EXTRACTION TAXONOMY

As listed above, there are three approaches to extract features from network flows: *Observing traffic properties*, *deriving information from the encrypted packet* itself and *extracting supplementary information* such as protocol handshake properties (Khalife, Hajjar, and Diaz-Verdejo 2014). We will now shortly introduce and explain these approaches.

***Behavioural feature extraction*.** As described above, this approach consists of observing how the traffic 'behaves' and modelling this information into features. (Moore et al. 2005) list over *240* features which can be extracted from any given flow using statistical means. They extract features from the TCP protocol, which makes their approach widely applicable. Some of the most interesting features include: *Inter-arrival-time*, *packet-length*, *number of ACK packets observed*, *number of retransmissions*, *round trip time* and corresponding *mean*, *variance*, *0.25* and 0.*75 quantile* for these features.

Figure 3: Promising features for modelling traffic behaviour

Feature extraction from data payload. Features may also be constructed directly of the packet payload. This approach has its origin in unencrypted traffic analysis, where regular expressions and signature matching are commonly used to infer information. However, some of the tools used for unencrypted traffic analysis may also be used for encrypted traffic analysis. (Velan et al. 2015) present surveys of tools for encrypted traffic analysis and find that while most tools for traffic analysis are designed to work on unencrypted data, a few of them manage to generalize to encrypted data as well. For example, they report studies which find the *libprotoident* tool[5] to be suitable for encrypted protocol analysis.

Another approach to feature extraction from payload data is neural networks, a machine learning method which has gained considerable traction during the past few years. Neural Networks may take in bytes as input and are able to infer information from this directly. There is no need for manual feature engineering. This method has been shown to work well for unencrypted data (Schneider and Böttinger 2018), but also for encrypted traffic (Wang et al. 2017).

Another interesting approach is presented by (Sherry et al. 2015), who introduce a new protocol and encryption scheme called 'BlindBox' which aims to strike a balance between enabling deep packet inspection, while at the same time maintaining user privacy by encrypting the traffic. While this is an interesting concept, BlindBox requires both client and user to use their proposed protocol suite and will not work on standard SSL/TLS.

[5] libprotoident, The University of Waikato, Hamilton, New Zealand, https://github.com/wanduow/libprotoident, accessed November 2019

***Feature extraction from supplementary information sources*.** Information may also be extracted from information not present in the data packets themselves, but in the auxiliary packets present in a data flow, for instance by observing the TLS handshake or the X.509 certificate exchange. For example, it is possible to perform client fingerprinting based on the initial SSL/TLS handshake[6]. SSLBL[7] is a project dedicated to "detecting malicious SSL connections, by identifying and blacklisting SSL certificates used by botnet C&C servers".

Having seen how features may be extracted from encrypted traffic, we now turn our attention to specific use cases, where we will show how feature extraction and information processing work in conjunction.

[6] Qualys SSL Labs - Projects. "HTTP Client Fingerprinting Using SSL Handshake Analysis, https://www.ssllabs.com/projects/client-fingerprinting/, accessed November 2019
[7] SSLBL | Detecting Malicious SSL Connections, https://sslbl.abuse.ch/, accessed November 2019

4. ENCRYPTED TRAFFIC ANALYSIS USE CASE: APPLICATION IDENTIFICATION

4.1 PROBLEM DESCRIPTION

Identifying and classifying traffic according to the application layer offers substantial benefits in areas such as IT security, network maintenance, quality of service and business intelligence. Traffic Encryption poses limitations on this kind of analysis.

Identifying and classifying traffic according to ISO/OSI layer-7 application offers substantial benefits in areas such as IT security, network maintenance, quality of service and business intelligence. For example, this allows a company to use dynamic access control to restrict unauthorized applications, such as using peer-to-peer file-sharing. Another use case is trend analysis, where forecasting network demands is facilitated by matching network traffic to application.

In unencrypted traffic, several techniques can be employed to perform application identification. (Zander, Nguyen, and Armitage 2005) identify the following different approaches: Identification via port number, stateful reconstruction of session and application information from the packet content and finally data driven / statistical methods. In the following paragraph, we will briefly discuss these approaches. We shall see their limitations with respect to encrypted traffic and motivate the use of data driven and statistical methods for application classification in encrypted traffic.

Application protocol identification is most commonly done *by port number*. There are several lists of port numbers used by some of the most popular applications; e.g. “Common Application Ports - Bandwidth Controller”[8]. For example, port 80 is used by browsers and the HTTP protocol, 443 for secure web browsing (HTTPS) and port 4000 for ICQ. Classifying traffic by port number is, however, infeasible in today’s World Wide Web. Many applications have no strongly defined ports or use dynamically allocated ports. When several servers share a common IP address, the same port cannot be used by two servers at a time, so a different port must used. Moreover, a user may deliberately change the port of their application to bypass port-based detection (Zander, Nguyen, and Armitage 2005) or even malware designers may employ “a commonly used port to bypass firewalls or network detection systems and to blend with normal network activity to avoid more detailed inspection”[9].

Stateful reconstruction methods work by listening in on the network traffic, inspecting its contents, reconstructing them and matching them against a data base of application traffic schematics. This can obviously not be done when the traffic is encrypted.

Data driven methods work by applying statistical or machine-learning based methods to classify traffic based on features extracted from the network flow. The accuracy of such methods is extremely dependent on a) the quality of the data set and b) the features. Features may be extracted from both unencrypted and encrypted traffic (from either the unencrypted handshake, header statistics, packet entropy, etc.) This makes statistical methods a suitable candidate for application identification in encrypted traffic.

[8] Bandwidth Controller: Common Application Ports, http://bandwidthcontroller.com/applicationPorts.html, accessed November 2019
[9] MITRE ATT&CK technique: Commonly Used Port, https://attack.mitre.org/techniques/T1043/, accessed November 2019

In the following section, we give an overview over such methods and detail their applicability to application and application protocol identification.

4.2 DATA DRIVEN METHODS FOR APPLICATION PROTOCOL CLASSIFICATION

In this section we present work which applies statistical methods for application protocol classification, using behavioural features as presented in (Moore et al. 2005). All of the following work use data driven methods to perform protocol classification: These methods first acquire some data set of flows, extract behavioural features and train a machine-learning algorithm on the data. When surveying literature for application protocol identification, this is the prevalent in state-of-the-art approach.

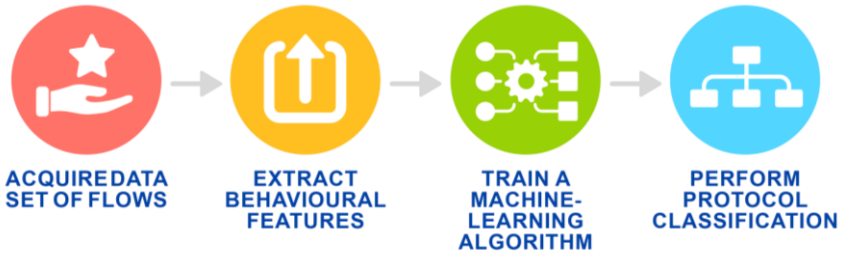
(Sun et al. 2010) employ statistical traffic analysis for application protocol classification. They use behavioural features and employ Naïve Bayes to classify flows of encrypted traffic as either HTTPS or TOR. While they do report high precision and recall (> 0.92), their evaluation is limited in that it considers only TOR vs. HTTPS traffic.

(Erman, Arlitt, and Mahanti 2006) perform application protocol classification by clustering algorithms, which is a specific sort of machine-learning algorithms which try to find group instances based on some notion of 'similarity'. The authors reference (Zander, Nguyen, and Armitage 2005) when describing their feature extraction mechanism and use these features to train both a K-Means Clustering, DBSCAN Clustering and AutoClass on the publicly available Auckland IV and Calgary dataset. While the results for K-Means and DBSCAN are mediocre (.84 accuracy after hyper parameter turning), AutoClass achieves better accuracy of .92. In summary, the authors show that application protocols such as DNS, FTP, HTTP, IRC, some P2P file sharing etc. may be distinguished using feature-based machine learning with reasonable accuracy.

(Bar - Yanai et al. 2010) present a clustering based classifier, where they collect and (semi-automatically) label a data set consisting out of 12 Million flows. They extract behavioural features from this data and train a hybrid k-means / knn model on their data. They are able to identify HTTP, SMTP, POP3, SKYPE, Edonkey and encrypted BitTorrent with high accuracy (> .94) and note that their algorithm is resistant to encryption.

Finally, (Wang et al. 2017) present an end-to-end approach to encrypted traffic classification using convolutional neural networks. This approach differs from the approaches above in that it does not perform manual feature extraction by first computing behavioural statistics such as round trip rime etc., but uses one-dimensional convolution neural networks to directly extract features from the raw traffic. These networks are most prominently used in computer vision tasks, but also gain traction in other fields such as natural language processing and information security. Convolutional networks work by sliding a 'kernel' over the input (which in this case is the raw, encrypted traffic), thus extracting contextual information which standard feed-forward networks fail to capture. The authors evaluate their approach on the ISCX VPN traffic data set and report precision and recall ranging between 0.7 and 0.95 when identifying Streaming, VoIP, Chat and Email within encrypted traffic.

Figure 4: Data driven methods for application protocol classification



4.3 DATA DRIVEN METHODS FOR APPLICATION TYPE CLASSIFICATION

Closely related to application protocol classification is application type classification. The former identifies protocols such as HTTP or SSH, while the latter identifies individual applications (such as Skype or Google Talk). The methodology for application type classification remains largely the same as for application protocol classification: State-of-the-art extracts behavioural features from a data set of flows and employs data-driven methods to perform classification. In the following paragraph, we present such work.

(Alshammari and Zincir-Heywood 2009) provide a comparison between five machine-learning based classifiers (Adaboost[10], SVM[11], Naïve Bayes[12], RIPPER[13] and C4.5[14]) for identifying SKYPE and SSH in encrypted network traffic. They choose features such as number of packets, inter arrival time and mean packet length and evaluate their approach on four data sets. They achieve high detection rate (.98) when detecting skype traffic, with an acceptable false positive rate (.07).

(Alshammari and Zincir-Heywood 2010) extend work from (Alshammari and Zincir-Heywood 2009) and map application protocol classification techniques to application type identification. Using similar classifiers (in both bases, the C4.5 algorithm is found to have superior performance), they show that Skype and Gtalk traffic can be reliably identified (true positive rate > .99, false positive rate < .02) within a data set containing FTP, SSH, MAIL, HTTP, HTTPS and MSN traffic.

A refreshing take on application identification is presented by (Taylor et al. 2018). The authors shift their attention to the identification of smartphone applications. The presence or absence of applications can reveal much information about the user (sexual orientation, health, religious believes). Even though these applications encrypt their communication using TLS/SSL, passive fingerprinting of behavioural traffic properties is enough to adequately detect application traffic in encrypted flows. The authors fingerprint 110 of the most popular Google Play Store Apps and classify them, six months later, with accuracy ranging between .66 and .73. They use behavioural features as presented in (Moore et al. 2005), plus packet timing and co-occurrence information and interact with the application using UI fuzzing (e.g. they use scripts to interact with the application in an automated way in order to generate the traffic). IP addresses were used only for flow separation and not for identifying the application itself (since IP addresses are likely to change during the lifetime of an application). In the following section we will discuss more efforts to determine active applications in a mobile device.

[10] Adaptive Boosting, https://en.wikipedia.org/wiki/AdaBoost, accessed November 2019
[11] Support-vector machine, https://en.wikipedia.org/wiki/Support-vector_machine, accessed November 2019
[12] Naive Bayes classifier, https://en.wikipedia.org/wiki/Naive_Bayes_classifier, accessed November 2019
[13] Repeated incremental pruning to produce error reduction, an optimized version of IREP proposed by William W. Cohen, http://www.csee.usf.edu/~lohall/dm/ripper.pdf, accessed November 2019
[14] C4.5 algorithm, https://en.wikipedia.org/wiki/C4.5_algorithm, accessed November 2019

4.4 APPLICATION IDENTIFICATION RECOMMENDATIONS

- Motivate the use of data driven and statistical methods for application classification in encrypted traffic
- Statistical methods are a suitable candidate for application identification in encrypted traffic

5. ENCRYPTED TRAFFIC ANALYSIS USE CASE: NETWORK ANALYTICS

5.1 APPLICATION CLASSIFICATION

BLINC was one of the pioneer papers that aimed to classify the network traffic in the dark, having no access to packet payload, no knowledge of port numbers and no additional information other than what current flow collectors provide, solely based on the host behavioural patterns (Karagiannis et al, 2005). (Wang et al., 2015) taking advantage of the fact that mobile applications produce more identifiable traffic patterns, perform a packet-level analysis to determine what applications a single individual is using, exploiting numerous side-channel information (e.g. traffic flow bursts) that are leaked through network traffic from mobile devices. AppScanner enables automatic fingerprinting of Android applications through encrypted traffic. To generate app fingerprints, the authors run apps automatically on a physical device to collect their network traces. The application classification is conducted using a supervised learning algorithm that is fed with features that are exported through the collection of network traces (Taylor et al., 2016). (Bernaille et al., 2007) propose a method to detect applications in SSL encrypted connections, taking advantage of the size of the first few packets of an SSL connection to recognise the application. Winter et al. present how the Great Firewall of China is able to identify and block Tor traffic, by dropping connections to certain ip:port tuples for a period of time or by examining the protocol (TOR traffic can be distinguished by inspecting the "TLS client/server hello'" messages) (Winter et al. 2012).

DNS tunnelling can be employed by attackers to protect communication between the malware and external C&C servers and keep the infection undetected for as long as possible in order to maximally exploit the target

5.2 APPLICATION USAGE CLASSIFICATION

The works in this section present fine-grained application event identification over encrypted traffic and clearly motivate the feasibility of traffic analysis, often with the use of machine learning techniques. (Coull et al.,2014) proposed a method for traffic analysis of encrypted messaging services. Specifically, they show that an eavesdropper can learn information about user actions inside an application, the language and the size of the messages exchanged. (Conti et al., 2015) proposed a system to analyse encrypted network traffic to identify user actions on Android devices, such as email exchange, interactions over social network, etc. Their framework leverages information that is available in TCP/IP packets, like IP addresses and ports, along with other features, like packet size, direction and timing. NetScope is a work that performs robust inference of users' activities, for both Android and iOS devices, based on inspecting IP headers. Its main purpose is to demonstrate that a passive eavesdropper is capable of identifying fine grained user activities within the wireless network traffic (even encrypted) generated by applications (Saltaformaggio et al., 2016). OTTer is a scalable engine that identifies fine-grained user actions, like voice/video calls or messaging, in Over-The-Top (OTT) mobile applications, such as WhatsApp and Skype, even in encrypted network traffic connections (Papadogiannaki et al., 2018).

5.2.1 VoIP Conversation Decoding

Others used traffic analysis to extract voice information from encrypted VoIP conversations. For example, (Wright et al., 2008) showed that when the audio is encoded using variable bit rate codecs, the lengths of encrypted VoIP packets can be used to identify phrases spoken within a call with high accuracy.

6. ENCRYPTED TRAFFIC ANALYSIS USE CASE: USER INFORMATION IDENTIFICATION

In the preceding chapters, several threats to user privacy by means of encrypted traffic analysis have been presented. In this chapter, we identify and detail further risks to user privacy when surfing the Internet over an encrypted channel.

6.1 IDENTIFY USERS' OPERATING SYSTEM, BROWSER AND APPLICATION

(Muehlstein et al. 2017) present an approach to identify a user's operating system, browser and their applications observing only the encrypted HTTPS traffic. Again, the authors exploit traffic characteristics similar to the ones presented in (Moore et al. 2005). In addition, they present new features which observe bursty behaviour of the browser, for example the silence before and after a transmission. The authors perform extensive experiments and present a data set of 20,000 incenses, which consists of traffic collected over 4 operating systems and 5 applications. Using this extended feature set, they can identify a user's operating system, browser and application with .96 accuracy.

Even though there are much more versions and flavours of operating systems, browsers and applications than they capture in their data set (and thus the real-world implications may be limited) the authors show that it is possible to extract private and sensitive information about the user's browser set up from encrypted traffic only.

7. ENCRYPTED TRAFFIC ANALYSIS USE CASE: DETECTION OF ENCRYPTED MALWARE TRAFFIC

Malware behaviour is as old as the internet. From this historical perspective, the competition between malware creators and defenders of companies, governments and ordinary users keeps malware constantly changing and evolving. With the growing usage of the Internet and its increasing importance for our daily lives, security plays a more profound and essential role than ever before. While security companies have developed methods to protect users against attackers, many challenges keep making this task difficult to achieve in real life. One of the most important challenges in security is encryption of the internet traffic, as it requires a fine balance between respecting the privacy of users as much as possible and at the same time gathering data from the traffic for detecting malicious behaviour with the best accuracy. This problem opens the doors for new approaches and methods of Artificial Intelligence.

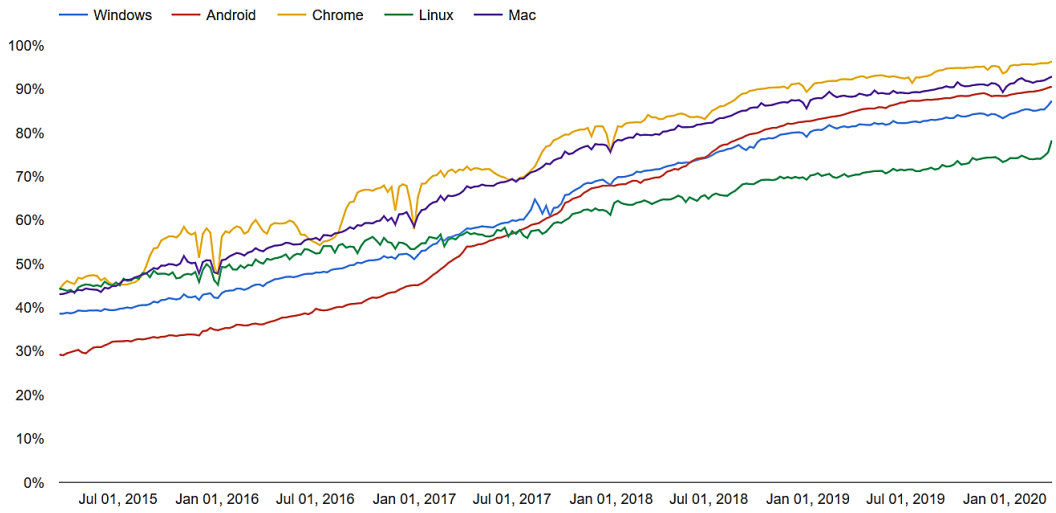
7.1 THE INCREASE OF HTTPS TRAFFIC

The most known and used protocol for the encryption of internet traffic is TLS (Transport Layer Security). TLS is most commonly used to encrypt the content of HTTP protocol. HTTPS protocol or HTTP Secure or Hypertext Transfer Protocol over TLS, is the standard for secure communications on the Internet and is dominantly used in any computer networks.

According to a Google report from April 2020[15] the usage of HTTPS is increasing and it is currently around 78-96% of loaded HTTPS web pages by the Chrome browser depending on the operating system (Windows, Android, Chrome, Linux, Mac). The report shows that Windows users using Chrome browser load almost 87% of visited websites over HTTPS, while Mac users load almost 93% and Linux users 78%.

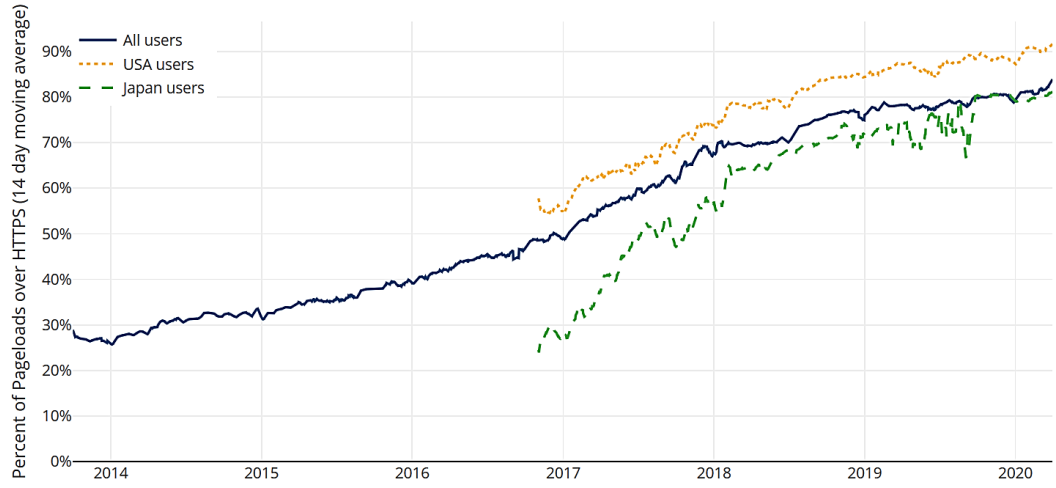
[15] Google Transparency Report, HTTPS encryption on the web, https://transparencyreport.google.com/https/overview, accessed April 2020

Figure 5: Google report from April 2020[15]



In March 2020, the Let's Encrypt certificate authority reported[16] that from all observed countries, an average of 83% of websites loaded by Firefox browser were encrypted.

Figure 6: Percentage of Web Pages Loaded by Firefox Using HTTPS[16]



There is an ongoing effort from global market actors, security organizations and authorities to increase the usage of HTTPS protocol to keep internet browsing safe by securely connecting browsers or applications with websites. Thereby the evolution and future perspective of HTTPS protocol as a way of encrypting traffic is expected to continue.

With this increasing amount of encrypted network traffic on the internet, malware has also started to use encryption to secure its own communication. In 2019 a report from Cisco[17] stated that more than 70% of malware campaigns in 2020 will use some type of encryption to conceal malware behaviour. It is important to realize

[16] Let's Encrypt Stats, Percentage of Web Pages Loaded by Firefox Using HTTPS, https://letsencrypt.org/stats/, accessed March 2020
[17] Cisco Encrypted Traffic Analytics, https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/nb-09-encrytd-traf-anlytcs-wp-cte-en.pdf, accessed November 2019

that “some type of encryption” doesn’t have to be only HTTPS (TLS), it could also refer to other protocols (IPSec) or even custom encryption schemes.

As far as malware using HTTPS for encryption are concerned, it is hard to get a true estimate. In 2016[18], Cisco reported around 10-12% of malicious communication using HTTPS and in 2017, Cyren[19] claimed that 37% of malware was using HTTPS. Although these estimations vary, it is safe to assume that there is an increase of encryption and HTTPS usage for the concealment of malware communication.

7.2 PROBLEM STATEMENT

The detection of malware traffic, which is not encrypted, is not a simple task, because even though all data are visible in the unencrypted traffic, it does not mean that all malicious behaviour can be detected there. The reason is simple: the attackers still come up with new types of attacks and malicious behaviour is still dynamically changing in contrast with other Machine Learning tasks, where targeted objects (e.g. images of humans, English sentences) do not change so much during our lives.

In general, the problem of malware traffic detection is a very difficult one from Machine Learning point of view. However, the detection of encrypted malware traffic compared to unencrypted traffic detection is a much harder problem, because all payload data of the traffic is hidden, due to encryption. In such cases, detection with high accuracy, low false positive and false negative rates is a challenge for the whole community.

7.3 INSPECTION SOLUTION

A possible solution for dealing with malware HTTPS traffic in companies and organizations is to install HTTPS interceptor proxies (aka middleboxes, TLS interception etc.) These hardware servers can open and inspect the HTTPS traffic of the employees by installing a special certificate in their computers. The HTTPS interceptor is placed between the client and the server, where the encrypted traffic is decrypted, scanned for malicious software, encrypted again and sent to the destination IP. This approach allows to use classic detection methods for detecting unencrypted malware traffic and it is a significant simplification of the problem.

However, there are few critical disadvantages. The first one is that it requires powerful hardware, because decryption and re-encryption of traffic without introducing noticeable delays is computationally demanding. The second disadvantage is that opening the traffic does not respect the original idea of HTTPS which is to have a private and secure communication throughout the channel.

On December 2019, National Security Agency (NSA) published a report[20] describing a potential risk from the improper usage of TLS inspection (TLSI). Claiming that “network owners should be aware that TLSI is not a cure-all" and there should be a big effort to set TLS Inspection properly. Otherwise the network can become more vulnerable and dangerous with TLS Inspection than without it.

In addition, we should bear in mind that HTTPS protocol is not the only way of encrypting malware traffic. According to a 2019 report from Cisco[17], in 2020 up to 60% of all companies

[18] Cisco, Hiding in Plain Sight: Malware’s Use of TLS and Encryption, https://blogs.cisco.com/security/malwares-use-of-tls-and-encryption, accessed November 2019
[19] Cyren Security Blog, Malware is Moving Heavily to HTTPS, https://www.cyren.com/blog/articles/over-one-third-of-malware-uses-https, accessed November 2019
[20] National Security Agency - Managing risk from transport layer security inspection, https://media.defense.gov/2019/Dec/16/2002225460/-1/-1/0/INFO SHEET MANAGING RISK FROM TRANSPORT LAYER SECURITY INSPECTION.PDF, accessed November 2019

using the inspection of traffic will fail to decrypt and identify malicious encrypted traffic due to adversaries employing custom encryption techniques. (see also ch. 10.4 dole)

7.4 A SOLUTION NOT REQUIRING DECRYPTION

As already mentioned, detection of encrypted malware behaviour is much more challenging compared to detection of malware in unencrypted traffic despite the fact there is relevant research with interesting results claiming the opposite:

- Learning communication patterns for malware discovery in HTTPs data (Kohout et al., 2018)
- Deciphering Malware's use of TLS (without Decryption) (Anderson et al., 2016)
- Detection of HTTPS Malware Traffic (Střasák , 2017)

One of the main tools used to detect encrypted malware traffic is Machine Learning. Employing ML algorithms requires us to choose appropriate features to analyse and a suitable data representation.

7.4.1 Features to detect malware with ML

Regarding features to detect malware using the HTTPS (TLS) protocol, there are a few important aspects to consider. The most valuable features are coming from the server's certificate. The purpose of the TLS certificate is to encrypt the traffic and prove the authenticity of the server, so the traffic is not only encrypted but also includes information proving the ownership of a public key in the certificate.

The first feature telling us something valuable about the HTTPS traffic is the validity length of the certificate. Each certificate is valid for a specified period. If the certificate has expired or if it is used before the commencement of its validity period can be an indication that something is potentially malicious. However, a short validity period of a certificate is not suspicious in itself.

The next interesting piece of information from the TLS certificate is the Subject Alternate Name (SAN) domains. This field contains at least one hostname however it can be valid for multiple ones. The number of hostnames and which hostnames are there can be useful for a later evaluation stage.

Each end-user certificate (the certificate the server sends to the client) should be issued by trusted Certificate Authorities (CA) that enable the end-user to verify that the server and all CA's are trustful. This process is called certificate chain and it is a critical part of HTTPS handshake. The number of certificate authorities in the chain and its domains are interesting information. However, some certificates can be self-signed which means that there is no certification authority and the certificate is signed by the same individual whose identity it certifies. This information can contribute to the final decision of malicious behaviour.

There are a lot of other features from the certificate and TLS handshake such as a version of TLS, signature algorithm, key type, etc. However, from August 2018 version 1.3 of the TLS protocol was defined in RFC 8446. TLS 1.3 has started to be used and deployed massively. The main difference between TLS 1.3 and previous versions of TLS (TLS 1.2, 1.1, 1.0, SSL 3.0, 2.0, and 1.0) is that TLS 1.3 is faster and safer and that is very good news from the end user point of view. The drawback is that the certificate is now encrypted inside the HTTPS traffic. Hence feature extraction from certificates, which was a very useful way of identifying malicious traffic, can no longer be used. So, while TLS 1.3 is a step in the right direction for the Internet and it is very good to use from a user perspective, it invalidates existing techniques making detection of malicious behaviour harder.

At this moment the most used version of TLS is still version 1.2, which does not encrypt the certificate, but in the following years the usage of TLS 1.3 is expected to eclipse past versions.

Losing access to TLS certificate features due to the introduction of TLS 1.3 can be mitigated by downloading the certificate for the observed traffic separately. Another solution is to rely only on features from the TCP layer such as the transferred number of packets and bytes between the client and server, the state of connection and the periodicity of packets or network flows. Whether or not these low-level features are enough for the detection of malicious encrypted traffic depends on many factors; e.g. what we want to detect, how much data we have, how well we understand the problem, which types of malware we want to detect, etc. However, another potential huge advantage of this approach is that we do not care about the type of encryption that the given malware uses.

7.4.2 Representation of data for ML

To detect any type of malware behaviour by Machine Learning, the first step and one of the biggest challenges is data. Of course, there are a lot of public and prepared datasets for a wide variety of tasks such as computer vision, natural language processing or speech recognition. However as far as public datasets for Malware traffic detection tasks, there are only a few of them. One of the public and available datasets for Malware traffic analysis is Malware Capture Facility Project[21] that contains a wide choice of Malware captures for last years.

In the case of generating a new dataset, the first step is to realize that the entire process can be expensive and time consuming, because it is necessary to generate enough traffic for the following research. It is also important to define at the beginning if the research will be focused on binary classification (Malware and Normal traffic) or classification of malware families (RAT, Trojan, etc.) for clear labelling of samples.

When the dataset is ready to use it is important to select suitable representation of samples for ML algorithms. There are plenty of possibilities and there is no best option in general, because it depends on the size of dataset and selected Machine Learning algorithm.

Examples of a sample for ML:

- Gathered all flows with the same source IP, destination IP, destination port, protocol
- Gathered all flows going in and out from one IP

For each sample from the dataset the features must be defined and computed. For example, in case of grouped flows with the same source IP, destination IP, destination port and protocol as a sample for ML, the features can be for example: mean of duration of flows, mean of transferred bytes between client and server, standard deviation of transferred packets between a client and server and so on. This is an area that is very important for future research. There is also a need to understand the traffic to identify which information from it is significant and which can be represented as the features for the new machine learning task.

Having defined the representation of samples and since the size of dataset is known and we also defined what we want to detect, it is time to select a Machine Learning method and algorithm. It is also good to analyse the data with some statistical tests to get an idea which features from your data are beneficial and which not.

Nowadays there are 2 main ways to use ML algorithms for data analysis. There are Classical Machine Learning methods and Deep Learning methods. Both have disadvantages and advantages for detection of malware behaviour.

[21] Malware Capture Facility Project https://www.stratosphereips.org/datasets-malware, accessed November 2019

For the classical ML approaches, the main concentration is directed towards high level features and expected patterns in internet traffic. The key to choosing essential features is to understand the traffic and compute high level features such as averages, standard deviations, etc. The advantages of this approach are that dataset does not have to be so large for training the model and the time for training is much shorter than in Deep learning scenario. There is also another interesting advantage about choosing this kind of features, because if the model does not work as excepted, the features can be changed very easily.

As far as Deep Learning approaches a large dataset and enough computational power is required. In this case it is better to choose low-level features instead of computing high level features such as means and standard deviations of samples. With the right representation of samples and with big enough dataset, the Deep Learning techniques can achieve better results than Classical ML approaches.

Every task is very specific and there is no defined procedure to select the right algorithm and correct representation of samples with features to achieve the best results. It is a difficult part of research and it requires time and knowledge to have a valuable detection system using Machine Learning systems.

To conclude, detection of encrypted malware traffic is a difficult area from Machine Learning point of view and the encryption exacerbates this problem. Furthermore, this challenge is escalated given the recent release of TLS 1.3 standard in 2018 that encrypts most of available information that were available in the older TLS protocols. The promising answer to the cyber security challenges of the beginning of the 21st century is Artificial Intelligence. These modern approaches help us automate the detection of malicious behaviour and at the same time respect the privacy of users as much as possible.

8. ENCRYPTED TRAFFIC ANALYSIS USE CASE: FINGERPRINTING

8.1 PROBLEM DESCRIPTION

When users browse the internet, many websites default to HTTPS, where the HTTP traffic is encrypted using TLS/SSL. While it is possible to infer the website they're surfing to by eavesdropping DNS requests or simply by observing the destination IP, a more granular analysis of a user's behaviour on the website is obscured by the encryption. For example, which sub-page, files or video clips are requested by the user cannot generally be inferred by observing DNS requests or IP headers. All an eavesdropper can observe are encrypted packets, destined to a given IP address. This ensures the user's privacy. A user's website traffic may reveal sensitive information such as sexual orientation, religious believes or medical information and must thus remain confidential.

A user's website traffic may reveal sensitive information such as sexual orientation, religious believes or medical information and must thus remain confidential. But there are, however, ways to infer which web pages, file, songs or videos are requested by a user, even if this traffic is encrypted

There are, however, ways to infer which web pages, file, songs or videos were requested by a user, even if the traffic is encrypted. Observing certain properties of the encrypted data, it is possible to create data records which map these properties to the corresponding files or websites. This is termed 'fingerprinting'. Given a corresponding data base of fingerprints, it is possible to find a match in the database of fingerprints and thus perform file or website identification even on encrypted traffic. In the following sections, we present work which uses variants of the above approach to perform file classification on encrypted traffic.

8.2 FILE FINGERPRINTING

In file fingerprinting, the goal is to identify which files (images, audio, video) are currently being transmitted over an encrypted channel. A popular approach to this problem is to observe the entropy of data packets. Entropy is the average amount of information given some stochastic source of data. For example, when tossing a fair coin, the Shannon entropy is 1 bit, since each coin toss yields one bit of information (either 0 or 1). Now suppose the coin is rigged and always yields heads. The corresponding Shannon entropy is 0 bit, since no information is gained by observing a toss. Coins with non-equiprobable outcomes thus have a Shannon entropy between 0 and 1 bit. Thus, the associated entropy can be understood as a 'fingerprint' of a coin. When observing enough tosses, we may estimate the entropy from the resulting distribution and thus find a mapping between entropy and coin. The same goes for network and file identification: When observing a given file, we may calculate its entropy by means of some stochastic property observable from the encrypted packets.

(Böttinger, Schuster, and Eckert 2015) use such an entropy based approach to file identification. The authors are able to estimate the entropy of the file being transmitted by exploiting file compression in TLS. Their approach is as follows: When a file is being transmitted via TLS, the file is split up into fragments of 214 bytes. These fragments are then compressed using some compression algorithm negotiated during the TLS handshake, and then the compressed fragments are encrypted and sent over the medium. Thus, a given file is split into n parts, where each part (except the last) is of length 214 bytes and where each part has size Cn after compression and encryption. This allows the authors to obtain a distribution of the file's entropy. This is because segments with high entropy contain more information, where compression cannot work as effectively, thus resulting in a larger file size. Thus, by using the compressed and encrypted packets' size as a proxy for the file's entropy, they are able to build

a database of files and corresponding entropy distributions. They apply their method to 5434 MP3 files, of which they create corresponding entropy fingerprints and are able to identify these files in TLS-encrypted traffic with an accuracy of .93 using a Random Forest. This vastly exceeds the random baseline (.00018) and shows that users' privacy is under threat even when using TLS.

8.3 WEBSITE FINGERPRINTING

A related subject is website fingerprinting (WFP), where the goal is to identify which website, or part of a website, a user is browsing to, given only the user's encrypted traffic. Extensive work has been done in this area, and we will highlight some important milestones in this section.

Early work on website fingerprinting has been presented in 2003 by (Hintz, 2003). The authors examine a common real world setting, where both encryption and a proxy server are employed in order to hide which websites are being requested. In their work, the authors use the now deprecated Internet Explorer 5 plus SSL encryption. However, this setup hides neither the file size nor the number of files downloaded, since for each download a separate TCP connection is instantiated. Thus, by simply building a database of 'fingerprints' (the number of files and their corresponding sizes), the authors implement an attack which can infer the requested site.

This attack becomes ineffective with the advent of more recent versions of the HTTP protocol, which make use of 'persistent HTTP' and 'HTTP pipelining', where a single TCP connection is used to transfer multiple files, without waiting for the corresponding responses. Thus, it is no longer trivially possible to distinguish individual files and infer their sizes.

However, the threat to users' privacy posed by WFP is far from banished. A more recent work has been presented by (Panchenko et al., 2016) in 2016, which analyses website fingerprinting over the TOR anonymity network. The authors make use of a similar concept as (Hintz, 2003) and observe that size, direction and timing of the transmitted packets still leaks considerable information, which is often enough to identify the requested website. More specifically, they identify websites from encrypted and TOR-anonymized traffic using the following feature set: Number of incoming and outgoing packets, sum of incoming and outgoing packet sizes and n=100 additional features which are derived from the cumulative sum of packet sizes over a trace of packages. Put differently, they fingerprint a website based on the timing and size of the encrypted packets. For evaluation, the authors propose two scenarios: A 'closed-world' scenario, where the number of websites a user may visit is limited and an 'open-world' scenario, where the task is to classify whether or not the current stream of traffic originates from an instance of a set of monitored websites. In the closed-world scenario, the author's approach achieves .91 accuracy when classifying an encrypted stream of traffic using an SVM classifier. Simulating an open-world scenario, the authors use a dataset which comprises 100 monitored websites' traffic fingerprint (90 instances each) and 9000 other pages, which serve as background noise. Their system is tasked to decide whether a given encrypted stream of traffic belongs to the monitored set or the background set. They achieve a true positive rate of more than .96 and a false positive rate of less than .02.

This is a significant result with real-world consequences. Even when using TOR and TLS, an observer (such as the Internet Service Provider, an Institution or a State) can accurately detect if a user browses a website from a monitored set. Dissidents in totalitarian regimes can no longer use TOR and be sure to anonymously access blocked internet sites and may thus be subjected to repercussions for browsing websites not conforming to the regime's ideology.

There have been efforts to mitigate the privacy threat posed by WFP. A straight-forward solution is to employ padding, such that the size of a file can no longer be inferred accurately (Juarez et al., 2016). These methods, however, come with considerable latency or bandwidth overhead ((Juarez et al., 2016) report about 60%), since superfluous data has to be transmitted in order to throw off an adversary. Additionally, research has shown that even when these defences are in place, deep learning methods still allow for successful WFP attacks (Sirinam et al., 2018).

8.4 DEVICE IDENTIFICATION

Other works focus on extracting TCP or IP packet metadata, in order to investigate if the behaviour of specific packet contents can be correlated with OSes, device types and other characteristics. (Chen et al., 2014) utilize multiple features in TCP/IP headers for OS identification, NAT and tethering detection. The authors use real network traffic traces to evaluate the accuracy of fingerprinting. Their study shows that several techniques that successfully fingerprint desktop OSes are not effective for fingerprinting mobile devices, accordingly. For OS fingerprinting they use the IP TTL value, the IP ID monotonicity, the TCP timestamp option, the TCP window size scale option, and the clock frequency. For tethering detection they use the TCP timestamp monotonicity, the clock frequency, and the boot time. (Ruffing et al., 2016) investigate OS identification against smartphones that use encrypted traffic. A traffic content agnostic identification algorithm is proposed that is based on the spectral analysis of the encrypted traffic. Their evaluation shows that the identification accuracy can reach 100% with an input of 30 seconds of network traffic.

8.5 LOCATION ESTIMATION

(Husted et al., 2010) present the ability to use wireless radios for positioning and tracking individuals' movements. The authors provide evidence that a small, but not insignificant number of mobile devices can be used to track a majority of users during a significant fraction of their travel. (Musa et al., 2012) focus on passively tracking unmodified smartphones, based on such Wi-Fi detections. The authors propose a trajectory estimation method which takes second-by-second detections of a moving device as input, and produces the most likely spatio-temporal path taken. The results are evaluated using ground-truth GPS data. A cellphone's position can be located by monitoring the traffic of certain applications that provide location-based services, even over encrypted network traffic. For example, (Ateniese et al., 2015) show that an adversary could be able to extrapolate the position of a target user by just analysing the size and the timing of the encrypted traffic exchanged between that user and a location-based service provider.

9. ENCRYPTED TRAFFIC ANALYSIS USE CASE: DNS TUNNELLING DETECTION

If a computer is running malicious software, the attacker aims to keep this infection undetected for as long as possible in order to maximally exploit the target. For this reason, it is important that all communication between the malware and external Command and Control (C&C) servers is not detected and subsequently prevented by firewalls or intrusion detection systems. For example, a direct TCP connection to the outside could be too easily detected. For this reason, it is necessary for the attacker to hide the malware's network traffic in legitimate network traffic. One way to do this is DNS tunnelling.

DNS tunnelling can be employed by attackers to protect communication between the malware and external C&C servers and keep the infection undetected for as long as possible in order to maximally exploit the target

9.1 TECHNICAL BACKGROUND AND PROBLEM DESCRIPTION

DNS tunnelling is a technique that allows a bidirectional exchange of information via the DNS protocol. This is an abuse of DNS, which has originally been designed to convert human-readable domain names into IP addresses (A records for IP4, and AAAA records for IP6 addresses). For example, DNS is used to find the IP address 172.217.16.133 for the mail.google.com domain.

Additionally, DNS supports queries of the mail server (MX record) or the canonical name (CNAME record, representing the real or original name). For the example domain used above, the following CNAME record is returned: googlemail.l.google.com.

In this way, after the client has initiated a connection, information can be exchanged in both directions. Malware such as viruses in a company network use a malicious DNS server to receive commands or load additional malicious code via the DNS protocol alone. The attacker can easily register a domain and then has full authority to answer corresponding DNS queries.

If, for example, the attacker wants to load further malicious code onto the victim system, a so-called stager could send the following DNS request: getPayload001.evildomain.com. This request is then sent to the local DNS server in the company network, which does not know the domain and therefore has to rely on the answer from evildomain.com, which sends the following CNAME record as reply: *dg59knca2rlpmnh98jdwyasdfer34.evildomain.com*.

"dg59knca2rlpmnh98jdwyasdfer34" is then interpreted by stager as the first part of the malware code queried. After several similar requests, the entire malware code is transferred to the victim's system. Firewall and IDS systems only see DNS traffic, which tends to be considered harmless. Furthermore, there is no direct communication between the malicious software and the malicious domain. This makes DNS tunnelling very difficult to detect: The malicious communication is hidden “in plain sight”.

9.2 DNS TUNNELING DETECTION

The detection of DNS tunnelling is therefore of great importance for network administrators. However, although the communication is not encrypted, the identification of DNS tunnelling is problematic. The following is an overview of the various ways to approaching the identification DNS tunnelling.

Simple methods such as black- or whitelisting cannot be used in the context of DNS tunnelling detection. Blacklists are not feasible because a malicious attacker can always register new domains. Whitelists are not practicable because such lists cannot be maintained, and Internet access will be too restricted.

Thus, payload based analysis seems more promising, where individual DNS queries are analysed and classified as malicious or benign. The following properties can serve as a basis for classification.

- The size of incoming and outgoing DNS requests. When loading malicious code, attackers try to use the maximum character length in order to minimize the number of DNS requests that need to be sent.
- The entropy of the requested domains and their character distribution (Born and Gustafson 2010). Malicious domains often include cryptic subdomains, which encrypt malicious code or the instructions to be executed. These differ significantly from colloquial English. However, this need not necessarily be the case, and even benign domains often contain non-human-readable words, e.g. in the case of content-delivery-networks.
- The absence of accompanying, benign network traffic. Since DNS requests are often the first step to further requests, e.g. before an HTTP request through the browser, isolated DNS request indicate irregularities.

In addition, statistical analysis can be used, for example counting the number of requests in a certain time window. An example of this is the frequency distribution of DNS records. (Raman et al. 2013) found that on average 38-48% of DNS traffic is generated by A records, 25% by AAAA and 20-30% by CNAME records. If a frequency distribution of requests is observed which deviates significantly from this, it can indicate malicious activity.

(Buczak et al. 2016) use features like these to train a decision tree that distinguishes between benign and malicious DNS queries. They test their approach against DNS tunnelling software like Iodine and DNSCat2, and evaluate the accuracy of the system using True Positive / False Positive Detection Probability (TP/FP-DP).

Their approach achieves over 99% TP-DP, and 0% FP-DP. In addition, several DNS tunnels are detected which originate from tunnelling software not represented in the training set.

10. ENCRYPTED TRAFFIC ANALYSIS TECHNIQUES

10.1 OFFLINE ANALYSIS (ML & DL TECHNIQUES)

The majority of works that employ machine learning techniques to investigate the feasibility of analysis and classification of encrypted network traffic focus on supervised learning algorithms. This means that these techniques train their models using ground-truth datasets and then test the performance of these models. Furthermore, there is a fraction of works that focus on unsupervised learning techniques, using algorithms that test similarities presented among the data.

The majority of works that employ ML techniques for encrypted network traffic analysis focus on supervised learning algorithms. Although there is a fraction that focus on unsupervised learning techniques, using algorithms that test similarities presented among the data

For instance, in the category of tunnelled website classification and fingerprinting (e.g. network traffic over OpenVPN, OpenSSH), the most popular algorithms are Multinomial Naive Bayes (MNB) (Herrmann et al., 2019), Support Vector Machine (SVM) (Panchenko et al., 2011) and Hidden Markov Models (HMM) (Cai et al., 2012). In addition, Levenshtein distance and the Jaccard classifier are used in a number of works to examine similarities between website fingerprints and properly classify them into categories (Lu et al., 2010). In the domain of network traffic analysis for mobile application classification, authors use again the same algorithms between other classifiers, such as Random Forest (RF), Decision Trees, Gaussian Naive Bayes (Alan et al. 2016) and the k-Nearest Neighbours (k-NN) algorithm for pattern recognition (Draper-Gil et al., 2016). For more fine-grained classification, such as identifying usage actions and events inside mobile applications, authors use hierarchical clustering techniques (Fu et al., 2016). Authors seem to conclude that feature selection is the key point to successfully classify and characterize traffic. Anderson et al. examine and address the pitfalls in traffic analysis, such as inadequate and inaccurate ground-truth datasets and non-stationary data distribution (Anderson et al., 2017). Specifically, the authors show that combining diverse views of the data, such as features pertaining to how the application is transmitting data with features that are representative of the application, is an approach that can significantly improve the performance of typical machine learning algorithms. Their work focus on malware traffic detection.

Besides machine learning, researchers make use of neural networks and deep learning techniques. More specifically, in the domain of intrusion detection, (Shone et al., 2018) propose a deep learning classification model constructed using stacked non-symmetric deep autoencoders (NDAEs). (Tang et al., 2016) present a deep learning approach for flow-based anomaly detection in SDN environments. Authors build a Deep Neural Network (DNN) model for an intrusion detection system and train it with the NSL-KDD dataset, using six basic features of the NSL-KDD dataset. Kitsune (Mirsky et al., 2018) is a NIDS, based on neural networks, and designed for the detection of abnormal patterns in network traffic. It monitors the statistical patterns of recent network traffic and detects anomalous patterns.

Furthermore, in the domain of application characterization with traffic analysis, (Lotfollahi et al., 2017) present a system that is able to handle both traffic characterization and application identification by analysing encrypted traffic with deep learning, embedding stacked autoencoder and convolution neural network (CNN) to classify network traffic. (Cruz et al., 2017) identify tunnelled BitTorrent traffic with a deep learning implementation that takes a feature-set based on the statistical behaviour of TCP tunnels proxying BitTorrent traffic, transforms it to multiple timestep sequences, and uses it to train a recurrent neural network.

10.2 COMPUTATIONS (DIRECTLY) ON ENCRYPTED TRAFFIC

Besides typical encryption schemes, homomorphic encryption has been proposed. Homomorphic encryption differs from typical encryption methods in that it allows computation to be performed directly on encrypted data without requiring access to a secret key. The result of such a computation remains in encrypted form, and can at a later point be revealed by the owner of the secret key[22]. However, the resulting performance efficiency appears to be questionable when it comes to current network speed and capacity.

10.3 INSPECTION USING SIDE-CHANNEL INFORMATION (METADATA)

Cisco's Encrypted Traffic Analytics (ETA) is a proprietary solution for businesses that offers traffic security and analytics by utilising various features of the network traffic, extracted from other Cisco's technologies[23]. encrypted traffic analysis extracts four main data elements; (i)~the initial data packet, (ii)~the sequence of packet lengths and times, (iii)~the byte distribution, and (iv)~TLS-specific features. The initial data packet IDP is used to obtain packet data from the first packet of a flow. It allows extraction of interesting data such as an HTTP URL, DNS hostname/address, and other data elements. The TLS handshake is composed of several messages that contain interesting, unencrypted metadata used to extract data elements such as cipher suites, TLS versions, and the client's public key length. The sequence of packet lengths and times conveys the length of each packet's application payload for the first several packets of a flow, along with the inter-arrival times of those packets.

The byte distribution represents the probability that a specific byte value appears in the payload of a packet within a flow. The byte distribution of a flow can be calculated using an array of counters. The major data types associated with byte distribution are full byte distribution, byte entropy, and the mean/standard deviation of the bytes[17, 23].

OTTer (Papadogiannaki et al., 2018) is a scalable engine that identifies fine-grained user actions in OTT mobile applications even in encrypted network traffic. OTTer uses signatures of TCP packet payload length sequences and it is able to detect user actions, such as voice/video calls and messaging, in four popular Over-The-Top applications, i.e. WhatsApp, Skype, Facebook Messenger and Viber. The performance overhead of OTTer when appended in a proprietary DPI engine was very low even in real traffic conditions.

10.4 MIDDLEBOXES AND INTERCEPTION OF ENCRYPTED TRAFFIC

Client-side software and network middleboxes that inspect HTTPS traffic operate by acting as transparent proxies. They terminate and decrypt the client-initiated TLS session, analyse the inner HTTP plaintext, and then initiate a new TLS connection to the destination website.

By design, TLS makes such interception difficult by encrypting data and defending against man-in-the-middle attacks through certificate validation, in which the client authenticates the identity of the destination server and rejects impostors. To circumvent this validation, local software injects a self-signed CA certificate into the client browser's root store at install time.

For network middleboxes, administrators will similarly deploy the middlebox's CA certificate to devices within their organization. Subsequently, when the proxy intercepts a connection to a particular site, it will dynamically generate a certificate for that site's domain name signed with its CA certificate and deliver this certificate chain to the browser (Durumeric et al., 2017).

[22] Basics of Homomorphic Encryption. http://homomorphicencryption.org/introduction, accessed November 2019.
[23] Encrypted Traffic Analytics (ETA). https://www.cisco.com/c/en/us/solutions/enterprise-networks/enterprise-network-security/eta.html, accessed November 2019.

(Goh et al., 2010A & Goh et al., 2010B) propose mirroring the traffic to a central IDS, able to decrypt the traffic and perform deep packet inspection, yet, without any privacy preserving guarantees. As Symantec states, most cyber threats hide in SSL/TLS encryption -- up to 70% of all network traffic. Symantec Proxies and SSL Visibility Appliance decrypt traffic to support infrastructure security and protect data privacy[24] . More specifically, Symantec offers the Encrypted Traffic Management (ETM) tool[25] that provides visibility into encrypted traffic by decrypting part of it; however this is a technique that could eventually cause privacy violations. Haystack enables fully device-local, context-aware traffic inspection on Android mobile devices using a standard app distributable via the usual app stores and offers device local, context-aware traffic inspection on commodity mobile devices. To offer full functionality --even for encrypted network traffic-- Haystack intercepts encrypted traffic via a local TLS proxy. At install time, Haystack prompts the user to install a self-signed Haystack CA certificate in the user CA certificate store (which they may accept or decline) (Razaghpanah, 2015).

10.4.1 Security Impact of HTTPS Interception

(Durumeric et al., 2017) show that web servers can detect interception by identifying a mismatch between the HTTP User-Agent header and TLS client behaviour. The authors build a set of heuristics to detect interception and identify the responsible product. Deploying these heuristics at the Mozilla Firefox update servers, a set of e-commerce sites and the Cloudflare content distribution network, they found out that there is an order of magnitude more interception than previously estimated. They investigate popular middleboxes and client-side security software, finding that nearly all reduce connection security and many introduce severe vulnerabilities.

[24] Symantec's SSL Visibility Appliance. https://www.symantec.com/products/ssl-visibility-appliance, accessed November 2019.
[25] Symantec's Encrypted Traffic Management Family. https://www.symantec.com/products/encrypted-traffic-management, accessed November 2019.

11. IMPROPER TLS PRACTICES

The Transport Layer Security (TLS) protocol provides security to network communications between a server and a client. It encrypts and secures a connection in a way that prevents eavesdroppers from reading and modifying the data that are being transferred. It also provides authentication for the participants of the connection.[26]

TLS is commonly used to secure the HTTP protocol. This combination is commonly referred to as the HTTPS protocol.[27] This widely popular extension of the HTTP protocol provides application level security to websites or desktop and mobile applications. This chapter will be focused on this protocol, but the principles mentioned apply to other uses of TLS as well.

The security of a connection does not come by default simply by enabling TLS. There are several versions of this protocols with different options. Therefore, the security features and advantages of this protocol are available only when it is used properly. An error during the development process of an application that uses TLS can cause its connections to be insecure; even though the connection might be encrypted, an attacker can easily evade the security measures and decrypt or modify it.

Improper TLS practices undermine the features that it offers and create a false sense of security. The additional danger comes from the fact that the parties communicating over such connection trust its security and are willing to transfer data they would not transfer over an unencrypted connection.

The core part of this chapter describes several common improper TLS practices and their impact. It then further proceeds with a study of the security of mobile Android applications. This study confirmed that the errors when using TLS are common among popular applications and dangerous for their users. It also shows the reaction of the developers to the issues found. The following section discusses the new version of TLS and the changes it brings. The chapter closes with a section summarizing the main good practices when deploying TLS in an application.

11.1 IMPROPER PRACTICES AND THEIR IMPACT

TLS offers security features for client/server connections. However, these features rely on proper practices when using this protocol. TLS connection with vulnerabilities created by bad implementation practices are often easily exploitable making the presence of TLS close to useless. This seemingly encrypted but in fact a vulnerable connection might be even more dangerous than if the network traffic was not encrypted at all.

The additional danger comes from the false sense of security that TLS connection with a vulnerability creates. A client using an application might trust it more just because it uses TLS. This leads to the client sharing more private information and possibly a subsequent attack which leads to a compromise of information that the client would otherwise not share had he known the connection was insecure.

The following sections describe improper practices when using TLS and their impact. For each subsection it is discussed how these issues occur and how are they exploited.

[26] https://tools.ietf.org/html/rfc5246, accessed November 2019.
[27] https://tools.ietf.org/html/rfc2818, accessed November 2019.

11.1.1 Lack of Certificate Validation

Certificate validation is a vital part of implementing an encrypted communication with TLS. The lack of this important practice can expose the communication to be vulnerable to man-in-the-middle (MITM) attacks. These attacks allow an adversary to easily decrypt all the network traffic as well as modify it. This might lead to the adversary getting any information from the traffic or exploiting the target device itself. This improper practice hurts TLS implementation in every way - eavesdropper can decrypt, observe and modify the network traffic and therefore overcome all the security measures of TLS.

For example, a transport application that fails to perform certificate validation is easily exploitable. With a MITM attack, an adversary can steal account credentials, observe the user's current location and the planned route to a destination. By modifying the traffic, the adversary can also inject a malicious link to exploit the client's phone or redirect the user to a different destination than the desired one.[28]

11.1.2 Man-in-the-middle attack on a TLS connection

The goal of a MITM attack is to observe or modify the victim's network traffic. It happens when an adversary gets into a specific position to be able to interact with the stream of data going to and from the victim's device. An example of that would be an attacker using a public Wi-Fi at a cafe and attacking a victim connected to the same Wi-Fi. With an ARP-poisoning attack, the adversary can redirect all the victim's traffic to another device. This attack makes the router and the victim send all the communication between them to the attacker instead of one another.

At this point, the adversary can observe the traffic and modify it. When the traffic is unencrypted, it is generally easy for the attacker to modify it without the client noticing the change in the data transferred. However, when the attacker modifies a TLS traffic, because of the integrity that TLS provides, the one receiving a modified packet will be able to notice that. For the receiver to not notice this change, the attacker must decrypt the traffic, modify it and then encrypt it using the standard TLS method. This however is possible only when there is a vulnerability such as the client not performing certificate validation.

During a connection's initiation phase, a server sends its certificate to a client. The client can then verify if this certificate is valid and if it belongs to that server. By validating the server's certificate, the client verifies the identity of the server. The certificate is a part of a key negotiation process. The keys generated from that process then enable to encrypt the upcoming communication. However, when this step during the connection initiation is not performed, it is no longer fully secure. The client cannot be sure of the server's identity. In such a case, an eavesdropper can swap a certificate coming from the server to the client for a different one. The certificate received by a client is not validated, and therefore the client doesn't notice anything and continues with the connection. This enables the adversary to get a hold of the key used to encrypt and decrypt the rest of the connection. The adversary eavesdropping on the communication can then observe and modify it.

[28] https://www.civilsphereproject.org/blog/2019/6/6/mobile-insecurity-series-application-czech-public-transport-idos-leaks-your-location-password-and-email-1, accessed November 2019.

Figure 7: The client connects to a Wi-Fi router which then forwards the connection to a server. The attacker does not interfere.

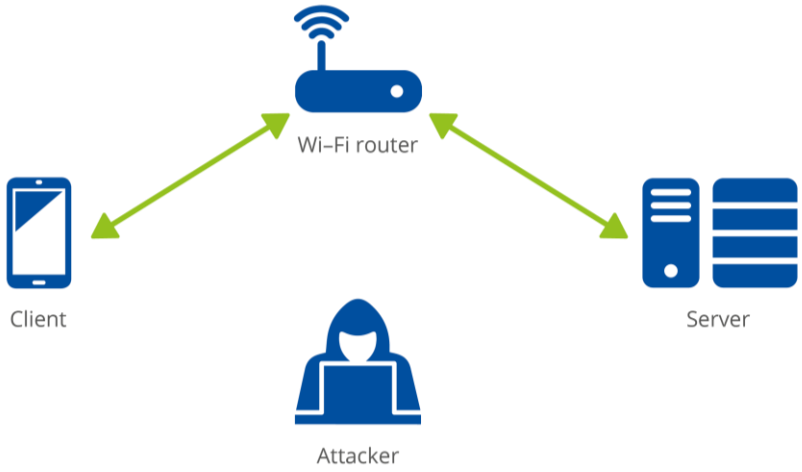
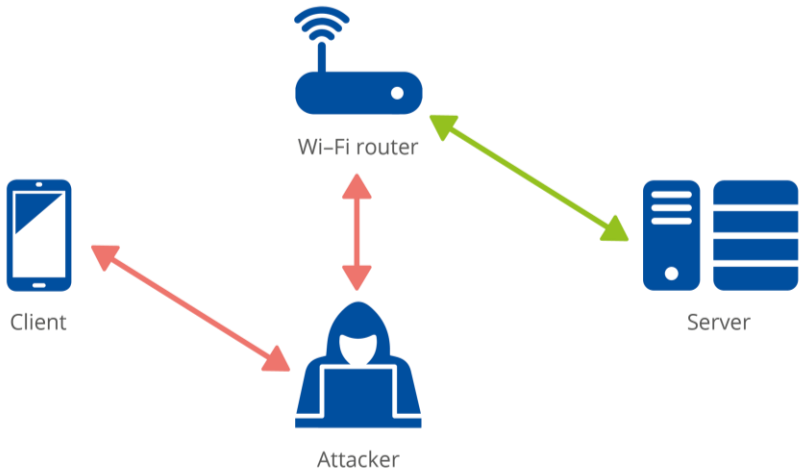


Figure 8: The ideal position for an attacker to perform a MITM attack is to be on the route of a connection from a client to a server. In this scenario, an attack forced the router and the client to send all the traffic to the attacker instead of one another.



Certificate validation is a necessary part of using TLS. Without its presence, the features and advantages of TLS are completely undermined. Moreover, the fact that the traffic is encrypted can be misleading as it is not fully secure. Behaviour of the user would rapidly change if the user knew that the data could be observed and modified.

11.1.3 Improper Use of HTTP Redirects

Redirection messages are a standard practice in HTTP and serve to redirect a client to a different website. They are initiated by an HTTP request sent to a host. The host then answers with a redirection message to redirect the client to a different location. This is a practice which enables an eavesdropper to modify the redirection packets and attack the client.

When a client tries to connect to a server using the insecure HTTP protocol, a server can try to redirect this connection to a secure one. A common practice is to use HTTP redirects which instruct the client to connect to a different network port which supports HTTPS. Redirecting a user to use HTTPS instead of HTTP seems like a good practice. It prevents the communication

to continue unencrypted. However, even one unencrypted packet can be exploited by an eavesdropper. This practice leads to leaking information as well as making the user visit malicious servers.

11.1.3.1 How redirects leak data

The first issue occurs when the initial HTTP request already contains information. It usually happens when an application initiates a connection with an unencrypted HTTP request packet. Such packet might already contain the user's credentials, a tracking number of a package or other sensitive data. This can be easily observed by an eavesdropper.

11.1.3.2 How they can be modified to redirect a victim to a malicious site

The second issue occurs when an eavesdropper modifies the redirection packet coming from the server, in particular the location field. This will make the client redirect to any site the eavesdropper wants to. Redirecting to a malicious website can cause the victim's phone to be exploited.

An example of that would be a vulnerability we discovered in an antivirus application which we tested as a part of our research that is described in a later section of this chapter called "Perspective of the Industry." The vulnerability enables for the following scenario. The user wants to log in, the application sends a single HTTP request that is then redirected to a secure connection. An adversary can however easily change the response from the server which was still unencrypted and redirect the user to a site that looked identical. The server that the user is redirected to is mimicking the login site but is managed by the attacker. There is no indication for the user that the login site is malicious which leads to the adversary getting credentials and possibly planting some malicious exploit.

The two packets in this type of unencrypted communication can lead to information leak and baiting a client to visit the website of an attacker's choosing. However, the responsibility is at the client's side. The client should not initiate unencrypted connection in the first place. The server's best options are redirecting to a secure connection. It is a standard practice that is convenient and widely used. Not responding to an HTTP request at all might seem like a more secure option but in the case of a MITM attack, the eavesdropper can just craft a custom redirection packet to respond the client's request without the servers reply.

HTTP redirects are a common practice that can be easily used for an attack. Nevertheless, it is often used by the developers of client applications and not considered as a vulnerability even though it poses a significant risk to the users.

11.1.4 Weak Ciphers and Deprecated Protocols

In the world of information technology security, there is a constant cycle of development, new exploits and further development to address the vulnerabilities found. TLS is no different. Throughout its existence there were several successful attacks that exploited different features of TLS. New practices and versions of TLS address these issues. However, when a developer of an application does not pay attention to this evolution, it is easy for an attacker to exploit it using known vulnerabilities in the old technology that an application uses.

11.1.4.1 TLS cipher suites and negotiating ciphers for a connection.

In the beginning of a TLS connection, both communicating parties agree to use a cipher suite to negotiate a symmetric key to encrypt and send application data. However, with the ever-evolving landscape of information technology, there are new vulnerabilities discovered in ciphers as well. And, with the increase of computing power, the time for an attacker to perform attacks which require heavy computation decreases. This makes it possible to perform such attacks in a reasonable time. The time it takes to guess a key or decrypt the network traffic usually depends on the length of the keys used and the strength of the encryption algorithms. These known threats to the security of TLS can be easily mitigated by not using old and

deprecated protocols and using the latest TLS versions with cipher suites recommended by the Internet Engineering Task Force (IETF).[29]

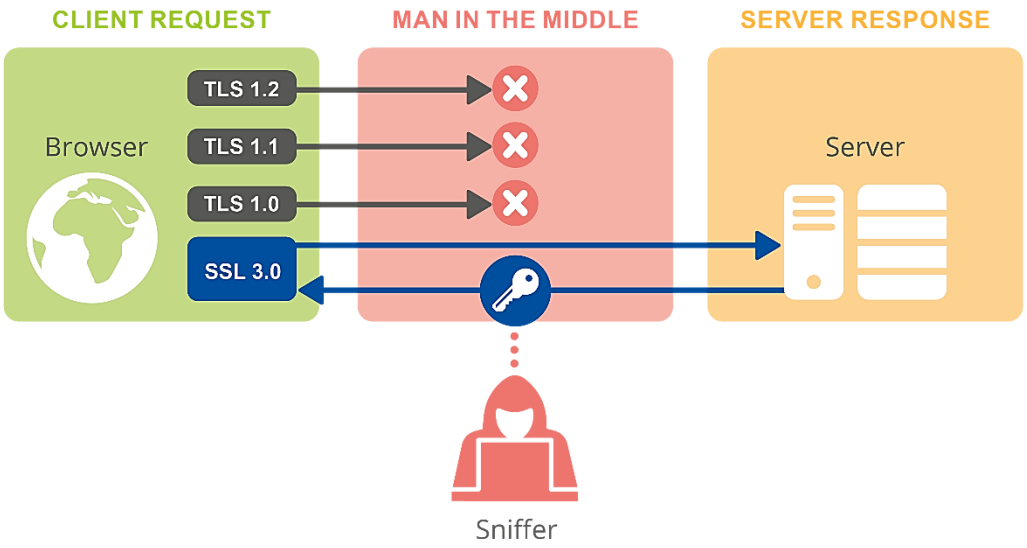
11.1.4.2 Notorious attacks that exploit vulnerabilities in the old and the new protocols

There are several known attacks against specific versions of TLS. These attacks can be usually easily mitigated using the latest TLS version. However, an attacker can find servers that use deprecated protocols and exploit their clients by using attacks that were used in the past before the vulnerabilities have been patched.

11.1.4.3 POODLE

An attack called the Padding Oracle On Downgraded Legacy Encryption affects the predecessor of TLS, SSL 3.0. This attack exploited the option in TLS which allowed the communication to be downgraded to SSL 3.0. After that, the attack consists of targeting the cipher-block-chaining (CBC) mode which eventually enables the attacker to decrypt the traffic. The impact is high as the attacker can decrypt information like cookies, passwords and other text which is sent encrypted.[30]

Figure 9: Attackers may force the communication between a client and server to downgrade from TLS to SSL 3.0 to be able to decrypt the network communication[31]



11.1.4.4 HEARTBLEED

Heartbleed is a name for a serious bug in the OpenSSL library which allows an attacker to decrypt the content that is encrypted using TLS. This was possible because of a vulnerability that enabled attackers to read the memory of systems that used the library. This was not a flaw in the TLS protocol itself but an implementation mistake in the OpenSSL library which provides cryptographic services such as TLS to applications.[32]

[29] https://tools.ietf.org/html/rfc7525, accessed November 2019.
[30] https://www-01.ibm.com/support/docview.wss?uid=swg21693271, accessed November 2019.
[31] https://blog.trendmicro.com/trendlabs-security-intelligence/poodle-vulnerability-puts-online-transactions-at-risk/, accessed November 2019.
[32] https://heartbleed.com, accessed November 2019

11.1.4.5 ROBOT

Return Of Bleichenbacher's Oracle Threat is an old attack with new updates that can exploit even the latest version of TLS. This attack targets the use of an insecure use of the RSA encryption mode. Attacker can record this network.[33]

11.1.4.6 Using key negotiation methods with no forward secrecy

Forward secrecy is a feature of a TLS connection which assures that in case of a future compromise of the private key of the given server the past connections cannot be decrypted. This is ensured by selecting specific key generation methods during the TLS handshake.[34] However, some protocols do not support this feature and using them creates a vulnerability that can have great consequences. When a server's private key gets compromised, an adversary can not only decrypt all the victim's traffic from that point on but also decrypt the connections for which the private key was used in the past.

It is necessary to use the latest TLS protocols with the recommended cipher suites and not use the protocols that are widely considered insecure.

11.1.4.7 Lack of adaptability

Another important practice for a development of secure applications is developing them to be adaptable when new exploits or issues occur. Whenever there is a new vulnerability discovered that affects the application, it is important to fix it as soon as possible. Because if the application stays vulnerable and is exposed to the Internet, there are most probably going to be attackers which will try the new attack on it. Time is usually of essence in these cases, so it is important for the developers to create an application in a way that enables them to make changes quickly. For example, deprecate a cipher suite, update a library, or a protocol used.

11.1.5 Other improper practices

There are other improper practices that undermine the features that TLS offers. A lot of them have to do with the way private keys are generated and stored. A private key that is used in the encryption process of a server's communication can be used by an attacker to decrypt and optionally modify the traffic. Other improper practices have to do with server certificates. Using a certificate that is not signed by a Certificate Authority or was generated by some untrusted source. The use of this certificate for a website can cause attackers to exploit the connection to this server with MITM attacks as browsers are not going to be able to validate the certificate.

The most common improper practices when deploying TLS are covered in this chapter. The practices mentioned are mostly compromising the security of a connection. Practices regarding the performance of the connection and other attributes of a TLS connection are not covered here.

11.2 THE NEW PROTOCOL AND ITS IMPACT

11.2.1 Introduction to TLS 1.3

Ten years after the TLS 1.2, a new version of the popular Transport Layer Security protocol was released.[35] This new protocol, called TLS 1.3, brings several improvements to performance and security. The changes help mitigate threats from a wide range of attacks, such as the previously mentioned ROBOT attack. Not only is the new version of TLS more secure, it also decreases the average time it takes to perform a TLS handshake which speeds up each connection.

[33] https://robotattack.org, accessed November 2019.
[34] https://www.ietf.org/rfc/rfc2409.txt, accessed November 2019.
[35] https://www.ietf.org/blog/tls13, accessed November 2019.

11.2.1.1 The biggest differences from the last version

The main differences between TLS 1.3 and TLS 1.2 are the following.

- The support for some symmetric encryption algorithms ended. The fact that insecure and old cipher suites were discarded is a positive update. However, it is important for servers to support this version and for application developers to use it as well.
- All public key based key exchange mechanisms now provide forward secrecy. This fact helps eliminate the use of deprecated and vulnerable key exchange mechanisms.
- All messages after server hello are encrypted. This means that the domain and the certificate will also be encrypted. This doesn't eliminate bad practices but helps prevent several attacks that tried to decrypt previously observed traffic.[36 & 37]

11.3 PROPER TLS PRACTICES

There are several good practices when deploying TLS which should be taken into consideration in order to take advantage of all the security features that the protocol offers.

11.3.1 Certificates validation and pinning

Certificate validation is an essential practice when using HTTPS anywhere. It helps the client authenticate the server. This practice prevents MITM attacks that intercept the server's certificate and allow an attacker to decrypt the traffic.

Certificate pinning is tying the given connection with a specific certificate which belongs to a specific host or with a certificate authority then signs certificates. This practice prevents MITM attacks as well as other attacks where the adversary installs a malicious certificate on the victim's device.

11.3.2 HTTP redirects

HTTP redirects are not a responsibility of servers. It is the client who initiates the connection and becomes vulnerable. Therefore, the client applications are responsible to initiate each connection using the TLS protocol. From the server's perspective, it is not considered a bad practice but rather a standard one. However, by not sending any unencrypted traffic, the servers can force the client applications to use strictly HTTPS.

11.3.3 Private Keys

The length of private keys also affects the security of the connection. A long enough key can prevent attacks by adversaries that try to guess it. It is also essential to store private keys properly so they are not accessible to anyone but the authorized individuals.

11.3.4 Using latest versions of TLS and deprecating older ones

Not allowing connections using old and deprecated protocols is a necessary practice which is also a responsibility of the server.

11.3.5 Deploying TLS

When implementing an application that uses TLS connections, it is important to be prepared to change and adapt according to new vulnerabilities and exploits. When a new vulnerability occurs, it is then essential for the application to quickly migrate to a safer option.

[36] https://tools.ietf.org/html/rfc8446, accessed November 2019.
[37] https://www.ietf.org/rfc/rfc5246.txt, accessed November 2019.

11.3.6 Certificate signing and trusted CAs

When deploying TLS on a server, it is important to make sure that the server's certificate is valid and signed by a trusted certificate authority. With an invalid certificate, it is easier to perform a MITM attack and mimic that certificate.

11.4 PERSPECTIVE OF THE INDUSTRY

Improper TLS practices are to be expected from individual developers who are still learning about TLS and do not have enough experience. This however should not be a common problem with larger companies and teams of developers that have all the resources to ensure that proper TLS practices are in place.

Nevertheless, a research by the CivilSphere project[38] points to the opposite. The researchers evaluated several mobile applications to see whether they had some of the issues mentioned in this chapter. Issues were found in over 81% of the applications that were tested. These issues ranged from single HTTP redirects to the lack of certificate validation. Moreover, many of the developers, to which the issues were reported to, never addressed them. These results point to the fact that poor practices are both widely spread and often not considered necessary to be fixed.

11.4.1 Widely spread malpractice

The researchers of CivilSphere tested several popular Android mobile applications. They focused mainly on two groups of applications. The first group consisted of the most popular Android Antivirus and security applications, the second one consisted of popular Android applications that are the responsibility of Latin American governments. All these applications were heavily used and together have well over 1.5 billion installations. Because of the nature of the applications, their wide use and the fact that the entities responsible for them are government institutions or large corporations, they are usually expected to be secure. The goal of this research by the CivilSphere project was to analyse the state of widely used and trusted Android applications and report on their security. The testing process was the following. The researchers chose an application to test. They connected an Android device to their own VPN which stores a copy of all the network traffic going through it. After that they installed the application and used it for some time as a normal user would. They tried the main functionalities of the application. After that they retrieved the network capture from the VPN and manually analysed the traffic.

In the network traffic, the team of researchers looked for any vulnerabilities and leaking information they could find. Finally, they reported these issues to the developers of the given application and continued to be in contact with them if they needed further assistance to fix the issues.

The most common issue was the use of the HTTP protocol for communication with vendor and advertisement servers. This communication consisted of HTTP redirects and other HTTP communication with no encryption. Second most occurring vulnerability was the lack of certificate validation. Third most occurring issue were occurrences of the use of deprecated protocol TLS 1.0.

These vulnerabilities result in leaking personal data and information about the user and the device. It also creates a way for attackers to inject exploits, malicious scripts and redirect users to malicious sites.

[38] https://www.civilsphereproject.org, accessed November 2019.

47 application were tested in total. 29 Security applications, 15 Latin American applications and three others.

- Over 96% applications had some of the above-mentioned security issues.
- 94% of them had some HTTP traffic that could be used in a malicious way by an adversary.
- 24% of applications tested lacked certificate validation. The total of applications tested for certificate validation was 30.
- 64% of the developers of applications that had some vulnerabilities did not respond to our reports in any way.

These common issues in applications make users vulnerable to all kinds of attacks. Moreover, a lot of the time the developers of these applications do not address the issues which leaves their users open for exploits. These issues affect over one billion Android users.

From the above one can conclude that the TLS protocol offers great features, but only when handled properly. It also provides freedom for developers to use it in different ways and not restricting them to specific features that are considered secure. As TLS is widely used it is important that it supports different cipher suites and other modifiable options. However, these options also allow for poor practices that undermine the features of TLS. It is therefore important for developers using TLS in their applications to know the protocol well and all the recommended practices that help provide a secure connection. Finally, a consequence of improper TLS practices is not only losing all the features that TLS offers but also creating a false sense of security that can cause more harm than if it was known that the traffic was unencrypted.

12. CONCLUSIONS

Increasing processing capabilities (with a decreasing cost), years of promoting cyber hygiene & security by design and conscious end users have led to widespread adoption of communication encryption. According to recent studies 70%-90% of internet traffic is protected by HTTPS and many applications employ encryption by default for their communication. Unfortunately, a similar picture is drawn in the adversarial part. Beyond ransomware, more and more sophisticated malicious software also employing encryption – whether standard protocols, like TLS, or custom cryptoalgorithms – to avoid detection and protect their communication. Hence it is important to consider the alternatives organizations have to analyse their [encrypted] network traffic in order to detect malicious activities and react appropriately.

The main conclusion of this report is that there is no one solution to rule them all, each has its drawbacks and many of the researched ML and AI based proposals have not reached an appropriate maturity level. Organizations will need to use a combination of tools and methods to protect their infrastructure and should carefully assess the risks, both negative and opportunities, inherent to them. Relying on only TLS inspection, for example, will leave the organization blind to threats using non-standard encryption.

Regarding the privacy implications, a big majority of the research proposals discussed are focused on end users' privacy, as opposed to security controls. In many instances the same results can be used both as a tool to protect cyber infrastructure and to invade a user's privacy; tracking, or leaking information about him or her. For example, fingerprinting techniques so far have been researched with respect to their privacy implications, but could potentially also be employed in data loss prevention.

In all cases, users should be aware that encryption cannot offer perfect privacy and especially pay attention to cases were identification and tracking are possible. Fingerprinting techniques, for example, pose a serious threat, especially in controlled environments. Furthermore, device characteristics, such OS, browser set up etc. can be revealed by observing only encrypted traffic and even privately owned and managed devices might leak enough information to allow adversaries to enumerate installed applications or even identify fine grained user activity, like voice/video calls or messaging.

The main challenge is then to find a balance between end-to-end security, protecting the privacy of end users and at the same time gathering valuable information from the traffic to detect possible threats and better allocate and protect resources. It is a difficult problem and it's important to discover new ways of detecting malicious behaviour, instead of simply opting for decreased user security and privacy. New technologies and algorithms of Artificial Intelligence and Machine Learning might be part of the solution, but as we saw this remains an active field of research were more effort should be put; especially on data driven and statistical methods for application classification in encrypted traffic, which seem quite promising.

Since the accuracy of data driven methods is extremely dependent on a) the quality of the data set and b) the features, it comes as no surprise that another area of research that needs attention is the generation of adequate datasets, of good quality, that will allow for training and testing new tools. Furthermore, one should remain aware of the constraint scope in which several proposed solutions operate, which somewhat limits their applicability in dynamic environments. Here again more research is required and encrypted traffic analysis using Neural Networks might be a possible direction; given that it does not require manual feature extraction, since they operate directly on the raw traffic, however yet more research is needed in the area.

In general, the problem of malware traffic detection is a very difficult one from Machine Learning point of view. However, the detection of encrypted malware traffic compared to unencrypted traffic detection is much more difficult problem, because all payload data of the traffic is hidden due to encryption and the detection with high accuracy, low false positive and false negative rates is a challenge for the whole community.

HTTPS interceptor proxies (aka middleboxes, TLS interception etc.) allow the use of classic detection methods (for detecting unencrypted malware traffic) on encrypted traffic, significantly simplifying the problem at hand. However, there are critical disadvantages; viz. resource intensive operations, does not respect end-to-end security, creates a potential point of exposure, it cannot protect against adversaries using non-standard encryption.

Detection of encrypted malware behaviour remains more challenging though and the introduction of TLS 1.3 makes classification of malicious behaviour even harder. Again more research will be needed to find alternative classification solutions for TLS 1.3, which might rely, for instance, on low-level features from the TCP layer. If successful, this methods would have the added benefit of being oblivious to the type of encryption used by the malware user.

Finally, to help mitigate future threats, like quantum cryptanalysis, it is necessary for developers to react quickly, making software adaptable to changes and developing it in a way that enables addressing security incidents as fast as possible. Designing a software without considering exploits and vulnerabilities that can occur is dangerous and will lead to security compromises.

13. REFERENCES

Alan, Hasan Faik, and Jasleen Kaur. *Can Android applications be identified using only TCP/IP headers of their launch time traffic?*. Proceedings of the 9th ACM conference on security & privacy in wireless and mobile networks. ACM, 2016. https://doi.org/10.1145/2939918.2939929

Alshammari, Riyad, and A. Nur Zincir-Heywood. 2009. *Machine Learning Based Encrypted Traffic Classification: Identifying SSH and Skype*. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009. https://doi.org/10.1109/CISDA.2009.5356534

Alshammari, Riyad, and A. Nur Zincir-Heywood. *An Investigation on the Identification of VoIP Traffic: Case Study on Gtalk and Skype*. In 2010 International Conference on Network and Service Management, IEEE, 2010. https://doi.org/10.1109/CNSM.2010.5691210

Anderson, Blake, S. Paul, and David McGrew. *Deciphering malware's use of tls (without decryption).* Journal of Computer Virology and Hacking Techniques, 2016. https://www.researchgate.net/publication/304965227_Deciphering_Malware's_use_of_TLS_without_Decryption

Anderson, Blake, and David McGrew. *Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity*. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2017.

Ateniese, Giuseppe, et al. *No place to hide that bytes won't reveal: Sniffing location-based encrypted traffic to track a user's position*. International Conference on Network and System Security. Springer, Cham, 2015.

Bar - Yanai, Roni, Michael Langberg, David Peleg, and Liam Roditty. *Realtime Classification for Encrypted Traffic*. Springer, Berlin, Heidelberg, 2010. https://doi.org/10.1007/978-3-642-13193-6_32

Bernaille, Laurent, and Renata Teixeira. *Early recognition of encrypted applications*. International Conference on Passive and Active Network Measurement. Springer, Berlin, Heidelberg, 2007.

Bishop, Christopher M., *Machine Learning and Pattern Recognition*. In Information Science and Statistics, 2006.

Born, Kenton, and David Gustafson. *Detecting DNS Tunnels Using Character Frequency Analysis*, 2010. https://arxiv.org/pdf/1004.4358.pdf

Böttinger, Konstantin, Dieter Schuster, and Claudia Eckert. *Detecting Fingerprinted Data in TLS Traffic*. In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15, New York, USA. ACM Press, 2015. https://doi.org/10.1145/2714576.2714595

Buczak, Anna L, Paul A Hanke, George J Cancro, Michael K Toma, Lanier A Watkins, and Jeffrey S Chavis. *Detection of Tunnels in PCAP Data by Random Forests*. In Proceedings of the 11th Annual Cyber and Information Security Research Conference*, CISRC 2016, 2016*. https://doi.org/10.1145/2897795.2897804

Cai, Xiang, et al. *Touching from a distance: Website fingerprinting attacks and defenses* [sic]. Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012. https://doi.org/10.1145/2382196.2382260

Chen, Yi-Chao, et al. *OS fingerprinting and tethering detection in mobile networks.* Proceedings of the 2014 Conference on Internet Measurement Conference. ACM, 2014.

Conti, Mauro, Luigi V. Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. *Can't You Hear Me Knocking: Identification of user actions on android apps via traffic analysis.* In Proceedings of the 5th ACM Conference on Data and Application Security and Privacy - CODASPY '15, New York, USA. ACM Press, 2015. https://doi.org/10.1145/2699026.2699119

Coull, Scott E., and Kevin P. Dyer. *Traffic analysis of encrypted messaging services: Apple imessage and beyond.* ACM SIGCOMM Computer Communication Review 44.5, 2014.

Cruz, Michelangelo, et al. *Fingerprinting BitTorrent Traffic in Encrypted Tunnels Using Recurrent Deep Learning.* 2017 Fifth International Symposium on Computing and Networking (CANDAR). IEEE, 2017.

Draper-Gil, Gerard, et al. *Characterization of encrypted and vpn traffic using time-related*. Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), 2016.

Durumeric, Zakir, et al. *The Security Impact of HTTPS Interception*. NDSS, 2017.

Erman, Jeffrey, Martin Arlitt, and Anirban Mahanti. *Traffic Classification Using Clustering Algorithms*. In Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data - MineNet '06, New York, USA: ACM Press, 2006. https://doi.org/10.1145/1162678.1162679

Fu, Yanjie, et al. *Service usage classification with encrypted internet traffic in mobile messaging apps*. IEEE Transactions on Mobile Computing 15.11, 2016.

Goh, Vik Tor, Jacob Zimmermann, and Mark Looi. *Experimenting with an intrusion detection system for encrypted networks*. International Journal of Business Intelligence and Data Mining 5.2, 2010.

Goh, Vik Tor, Jacob Zimmermann, and Mark Looi. *Intrusion detection system for encrypted networks using secret-sharing schemes*. International Journal of Cryptology Research, 2010.

Herrmann, Dominik, Rolf Wendolsky, and Hannes Federrath. *Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier*. Proceedings of the 2009 ACM workshop on Cloud computing security. ACM, 2009.

Hintz, Andrew. *Fingerprinting Websites Using Traffic Analysis*. Springer, Berlin, Heidelberg, 2003. https://doi.org/10.1007/3-540-36467-6_13.

Husted, Nathaniel, and Steven Myers. *Mobile location tracking in metro areas: malnets and others*. Proceedings of the 17th ACM conference on Computer and communications security. ACM, 2010.

Juarez, Marc, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. *Toward an Efficient Website Fingerprinting Defense* (sic). In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9878 LNCS, 2016. https://doi.org/10.1007/978-3-319-45744-4_2.

Karagiannis, Thomas, Konstantina Papagiannaki, and Michalis Faloutsos. *BLINC: multilevel traffic classification in the dark*. ACM SIGCOMM computer communication review. Vol. 35. No. 4. ACM, 2005.

Khalife, Jawad, Amjad Hajjar, and Jesus Diaz-Verdejo. *A Multilevel Taxonomy and Requirements for an Optimal Traffic-Classification Model*. International Journal of Network Management 24 (2), 2014. https://doi.org/10.1002/nem.1855

Kohout, Jan, Tomáš Komárek, Přemysl Čech, Jan Bodnár, and Jakub Lokoč. *Learning communication patterns for malware discovery in HTTPs data*, 2018. https://doi.org/10.1016/j.eswa.2018.02.010

Lotfollahi, Mohammad, et al. *Deep packet: A novel approach for encrypted traffic classification using deep learning*. Soft Computing (2017): 1-14, 2017.

Lu, Liming, Ee-Chien Chang, and Mun Choon Chan. *Website fingerprinting and identification using ordered feature sequences*. European Symposium on Research in Computer Security. Springer, Berlin, Heidelberg, 2010.

Mirsky, Yisroel, et al. *Kitsune: an ensemble of autoencoders for online network intrusion detection*. arXiv preprint arXiv:1802.09089, 2018.

Moore, Andrew, Denis Zuev, Michael Crogan, Andrew W Moore, and Michael L Crogan. *Discriminators for Use in Flow-Based Classification Discriminators for Use in Flow-Based Classification* *, 2005. https://www.cl.cam.ac.uk/~awm22/publication/moore2005discriminators.pdf

Muehlstein, Jonathan, 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 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 1–6. IEEE, 2017. https://doi.org/10.1109/CCNC.2017.8013420

Musa, A. B. M., and Jakob Eriksson. *Tracking unmodified smartphones using wi-fi monitors*. Proceedings of the 10th ACM conference on embedded network sensor systems. ACM, 2012.

Panchenko, Andriy, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. 2016. *Website Fingerprinting at Internet Scale*. https://doi.org/10.14722/ndss.2016.23477

Panchenko, Andriy, et al. *Website fingerprinting in onion routing based anonymization networks*. Proceedings of the 10th annual ACM workshop on Privacy in the electronic society. ACM, 2011.

Papadogiannaki, Eva, et al. *OTTer: A Scalable High-Resolution Encrypted Traffic Identification Engine*. International Symposium on Research in Attacks, Intrusions, and Defenses. Springer, Cham, 2018.

Raman, Daan, Bjorn De Sutter, Bart Coppens, Stijn Volckaert, Koen De Bosschere, Pieter Danhieux, and Erik Van Buggenhout. *DNS Tunneling for Network Penetration*. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 7839 LNCS, 2013. https://doi.org/10.1007/978-3-642-37682-5_6

Razaghpanah, Abbas, et al. *Haystack: A multi-purpose mobile vantage point in user space*. arXiv preprint arXiv:1510.01419. 2015.

Ruffing, Nicholas, et al. *Smartphone reconnaissance: Operating system identification*. 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2016.

Saltaformaggio, Brendan, et al. *Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic*. 10th USENIX Workshop on Offensive Technologies (WOOT 16). 2016.

Schneider, Peter, and Konstantin Böttinger. *High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks*. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy - CPS-SPC '18, 1–12. New York, New York, USA: ACM Press, 2018. https://doi.org/10.1145/3264888.3264890

Sherry, Justine, Chang Lan, Raluca Ada Popa, Sylvia Ratnasamy, Justine Sherry, Chang Lan, Raluca Ada Popa, and Sylvia Ratnasamy. *BlindBox: Deep Packet Inspection over Encrypted Traffic*. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15, 45, New York, USA: ACM Press, 2015. https://doi.org/10.1145/2785956.2787502

Shone, Nathan, et al. *A deep learning approach to network intrusion detection*. IEEE Transactions on Emerging Topics in Computational Intelligence 2.1, 2018.

Sirinam, Payap, Mohsen Imani, Marc Juarez, and Matthew Wright. *Deep Fingerprinting*. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18, 1928–43. New York, New York, USA: ACM Press, 2018. https://doi.org/10.1145/3243734.3243768

Sun, Guang-Lu, Yibo Xue, Yingfei Dong, Dongsheng Wang, and Chenglong Li. *An Novel Hybrid Method for Effectively Classifying Encrypted Traffic*. In 2010 IEEE Global Telecommunications Conference GLOBECOM 2010. IEEE, 2010. https://doi.org/10.1109/GLOCOM.2010.5683649.

Střasák, František. *Detection of HTTPS Malware Traffic*. Thesis, Czech Technical University in Prague, 2017. https://dspace.cvut.cz/bitstream/handle/10467/68528/F3-BP-2017-Strasak-Frantisek-strasak_thesis_2017.pdf

Tang, Tuan A., et al. *Deep learning approach for network intrusion detection in software defined networking*. 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM). IEEE, 2016.

Taylor, Vincent F., et al. *Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic*. 2016 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE, 2016.

Taylor, Vincent F., 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), 2018. https://doi.org/10.1109/TIFS.2017.2737970

Velan, Petr, Milan Čermák, Pavel Čeleda, and Martin Drašar. A *Survey of Methods for Encrypted Traffic Classification and Analysis*. International Journal of Network Management 25 (5), 2015. https://doi.org/10.1002/nem.1901

Wang, Qinglong, et al. *I know what you did on your smartphone: Inferring app usage over encrypted data traffic*. 2015 IEEE Conference on Communications and Network Security (CNS). IEEE, 2015.

Wang, Wei, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. *End-to-End Encrypted Traffic Classification with One-Dimensional Convolution Neural Networks*. In 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), 43–48. IEEE, 2017. https://doi.org/10.1109/ISI.2017.8004872

Winter, Philipp, and Stefan Lindskog. *How the great firewall of china is blocking TOR*. USENIX-The Advanced Computing Systems Association, 2012.

Wright, Charles V., et al. *Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations*. 2008 IEEE Symposium on Security and Privacy. IEEE, 2008.

Zander, S., T. Nguyen, and G. Armitage. *Automated Traffic Classification and Application Identification Using Machine Learning*. In The IEEE Conference on Local Computer Networks 30th Anniversary. IEEE, 2005. https://doi.org/10.1109/LCN.2005.35

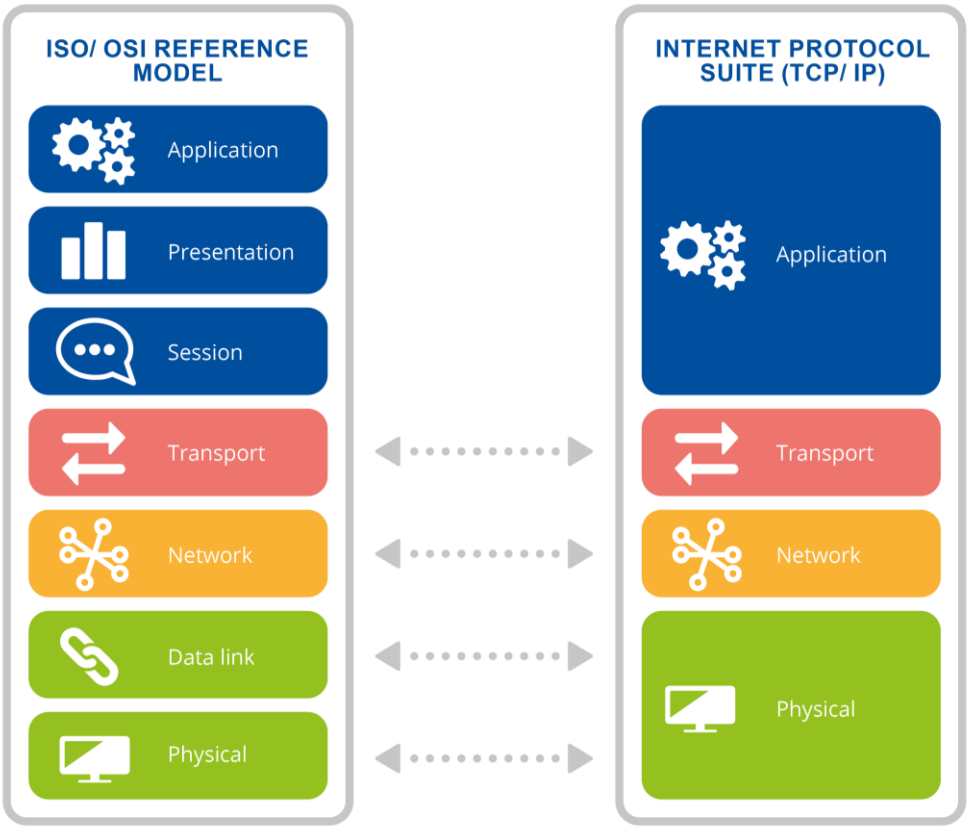
Zimmermann, H. *OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection*. IEEE Transactions on Communications 28 (4), 1980. https://doi.org/10.1109/TCOM.1980.1094702

A ANNEX: TECHNICAL BACKGROUND

A.1 COMMUNICATION IN COMPUTING SYSTEMS

IT network communication is standardized by the Open Systems Interconnection model (OSI model) (Zimmermann 1980). In this model, communication is abstracted into seven layers, where each layer performs a specific task. For example, the first layer (the physical layer) converts digital bits into an analog signal that can then be sent over a wire or over the air. Upper layers are responsible for segmentation and error correction (layer 2, Ethernet), routing (layer 3, IP), segmentation and error handling of the routing layer (layer 4, TCP) and the application layer (layer 7), which usually comprises layers 5 and 6.

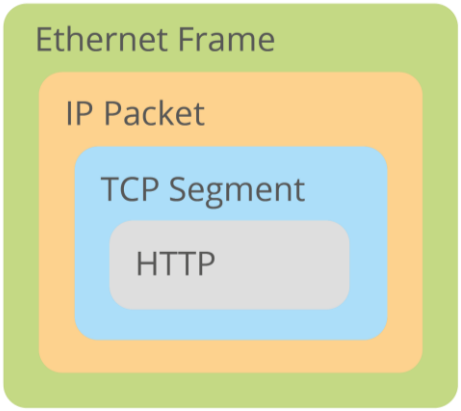
Figure 10: ISO/OSI TCP/IP comparison



As an example, consider a server sending a website via HTTP to a client. The application layer constructs the HTTP packet and hands it down to the next layer, the transport layer. There, the TCP protocol is used to perform flow control, segmentation and error control. The transport layer splits the packet into smaller protocol data units (PDU), each of which is handed down to the IP layer, which performs routing services. During this process, each layer adds a small section at the beginning of the packet, which contains some layer-specific information (for example, the IP layer adds a header which contains the destination IP address). Thus, the packet is encapsulated or 'wrapped' by each processing layer, with the original packet

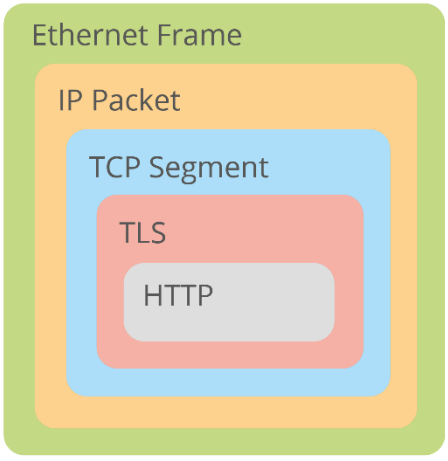
(constructed by the application layer) as the innermost packet. The following picture gives a graphical illustration. The HTTP packet is encapsulated by the TCP segment, which is encapsulated by the IP packet, which again is encapsulated by the Ethernet frame.

Figure 11: Encapsulated HTTP packet



Note that this packet is not encrypted at all, so at any given time, any of the corresponding packets can be inspected. Interesting properties such as the application protocol as well as the application content can be directly observed. Encryption is added by introducing Transport Layer Security (TLS), a cryptographic protocol which runs on top of TCP. Thus, the HTTP packet is encrypted using TLS, and the encrypted bytes are passed down to TCP, which processes them as usual. This results in the following packet structure.

Figure 12: TLS encrypted encapsulated HTTP packet



This obscures the application protocol. When inspecting such a packet, anything within the TLS frame is encrypted, and the original application protocol may not be inferred directly, let alone its contents.

A.2 MACHINE LEARNING BACKGROUND

Machine learning methods are data driven methods. This means that for a machine learning algorithm to learn, it requires data. In supervised learning, data consists of a pair (X, y), where X contains the data to learn from, and y constitutes the target class to learn. For example, X may be an (n, d) matrix comprising n observations, where each observation comprises d features. The appropriate Y would be an array of size n, containing the ground truth (e.g. the result the classifier has to learn).

Features are atomic observable properties of a given network flow or packet. In encrypted traffic, these features may be obtained from either the header of a packet, properties of the unencrypted handshake, by observing statistical properties such as the round-trip-time (RTT) or even from the encrypted data itself (via byte-patterns). While there are many ways to extract features, all of the approaches share the same premise: Different applications exhibit different behaviour, which will transpire to the features. With a large enough data set, a mapping f may be created which maps a set of features to their corresponding application. This principle underlies all of ML and suffices to understand the ML used in the following chapters. A detailed study on the subject may be found in (Bishop 2006).

ABOUT ENISA

The mission of the European Union Agency for Cybersecurity (ENISA) is to achieve a high common level of cybersecurity across the Union, by actively supporting Member States, Union institutions, bodies, offices and agencies in improving cybersecurity. We contribute to policy development and implementation, support capacity building and preparedness, facilitate operational cooperation at Union level, enhance the trustworthiness of ICT products, services and processes by rolling out cybersecurity certification schemes, enable knowledge sharing, research, innovation and awareness building, whilst developing cross-border communities. Our goal is to strengthen trust in the connected economy, boost resilience of the Union's infrastructure and services and keep our society cyber secure. More information about ENISA and its work can be found www.enisa.europa.eu.

ENISA
European Union Agency for Cybersecurity

Athens Office
1 Vasilissis Sofias Str
151 24 Marousi, Attiki, Greece

Heraklion office
95 Nikolaou Plastira
700 13 Vassilika Vouton, Heraklion, Greece

enisa.europa.eu