Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting

Martin Husák, Milan Čermák, Tomáš Jirsík, Pavel Čeleda
Institute of Computer Science, Masaryk University, Brno, Czech Republic
{husakm, cermak, jirsik, celeda}@ics.muni.cz

***Abstract*—The growing share of encrypted network traffic complicates network traffic analysis and network forensics. In this paper, we present real-time lightweight identification of HTTPS clients based on network monitoring and SSL/TLS fingerprinting. Our experiment shows that it is possible to estimate the User-Agent of a client in HTTPS communication via the analysis of the SSL/TLS handshake. The fingerprints of SSL/TLS handshakes, including a list of supported cipher suites, differ among clients and correlate to User-Agent values from a HTTP header. We built up a dictionary of SSL/TLS cipher suite lists and HTTP User-Agents and assigned the User-Agents to the observed SSL/TLS connections to identify communicating clients. We discuss host-based and network-based methods of dictionary retrieval and estimate the quality of the data. The usability of the proposed method is demonstrated on two case studies of network forensics.**

I. Introduction

The rising popularity of encrypted network traffic is a double-edged sword. On the one hand, it provides secure data transmission, protects against eavesdropping, and improves the trustworthiness of communicating hosts. On the other hand, it complicates the legitimate monitoring of network traffic, including traffic classification and host identification. Nowadays, we are able to monitor, identify, and classify plain-text network traffic, such as HTTP, but it is hard to analyse encrypted communication. The more secure the connection is, from the point of view of communicating partners, the harder it is to understand the network traffic and identify anomalous and malicious activity. Furthermore, the attackers can evade disclosure by hiding malicious network behaviour in encrypted connections, where it is invisible to detection mechanisms.

In this paper, we discuss the most common encrypted network traffic, HTTPS - HTTP over SSL/TLS protocols. In communication encrypted by SSL/TLS, the hosts have to first agree on encryption methods and their parameters. Therefore, the initial packets contain unencrypted messages containing information about the client and server. The fingerprint of a SSL/TLS handshake varies among different clients and their versions. One similar client identifier is a User-Agent value in a HTTP header, which is commonly used for identifying the client and classifying traffic. However, only the SSL/TLS handshake can be observed in a HTTPS connection without decrypting the payload. Therefore, we approach the problem of identifying the SSL/TLS client and classifying HTTPS traffic by building up a dictionary of SSL/TLS handshake fingerprints and their corresponding User-Agents.

A. Motivation

The motivation for our work came from two case studies of HTTPS client identification in the field of network forensics. The first example addresses client identification in network traffic destined to a HTTPS server. The second case study addresses a HTTPS-based client identification in a Network Address Translation (NAT) environment.

In the first case study, a HTTPS server is facing malicious network traffic. The goal is to analyse the incoming traffic and identify malicious connections. We do not have access to the server, therefore we have to analyse incoming network traffic. If the server was running HTTP, we could analyse the User-Agents of the clients and filter out legitimate ones, e. g., common web browsers, to get a set of potentially malicious clients. However, as more and more services move from HTTP to HTTPS, it is common to see web servers running HTTPS only. In this case, we have to rely on HTTPS client fingerprinting as the only source of information which leads to the identification of clients.

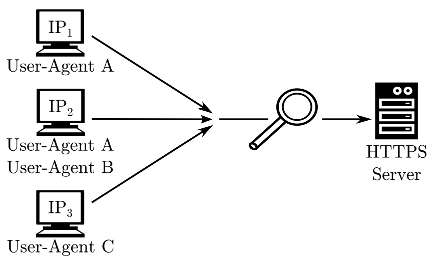

Fig. 1. Network-based HTTPS client identification.

In the second case study, a network of clients located behind a NAT is monitored. Therefore, we have to evaluate other identifiers than IP addresses. The first task is to identify and enumerate distinct clients. The second task is to detect presence of particular clients. Basic information, such as the operating system, can be obtained via packet fingerprinting [1]. Other valuable information for client identification is provided by the User-Agent, which can be obtained from any HTTP connection. However, we are missing a significant portion of HTTPS traffic, which would not be a problem if we didn't have a limited time for monitoring or we had a more specific query to process. For example, if we are tracing a mobile device which is connecting to a particular HTTPS server,

978-1-4673-6590-1/15 $31.00 © 2015 IEEE
DOI 10.1109/ARES.2015.35

such as Gmail, we are again left with only HTTPS client fingerprinting.

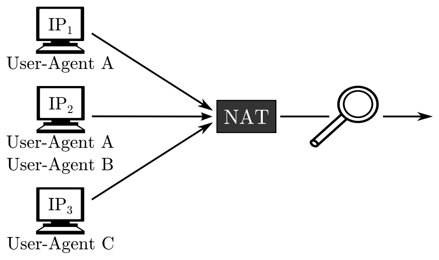

Fig. 2. HTTPS-based identification of clients behind NAT.

B. Research Questions

We set up an experiment, which shall answer three research questions. To sum up our goals, the research questions are:

(i) Which parameters of a SSL/TLS handshake can be used for client identification?

(ii) How can we build a dictionary of SSL/TLS handshakes and HTTP User-Agents?

(iii) How large does the dictionary need to be to cover a significant portion of network traffic?

First, we aim to observe of real network traffic to gain insight into contemporary SSL/TLS handshakes. We deployed network traffic monitoring, filtered HTTPS connections, and made a list of the SSL/TLS handshakes and their fingerprints. We focused on analysing information provided during the handshake by the client, i. e., the *ClientHello* message containing the protocol version, list of supported cipher suites, and other data. Apart from the usable information for identifying the client, we are particularly interested in the share of old and vulnerable protocol versions. Recent discoveries of severe vulnerabilities, such as POODLE [2], might have significantly changed the proportion of protocol versions in use.

Second, we correlated selected parts of SSL/TLS handshakes and HTTP headers. We suppose that the list of supported cipher suites (declared by the client in the *ClientHello* message) can be used as an identifier similarly to a User-Agent in a HTTP header. However, it is not possible to get the User-Agent from the HTTPS request without decryption. We use two approaches to obtain pairs of cipher suites and evaluate User-Agents. The host-based approach is based on advanced logging on the server side. The novel network-based method is based on simultaneously monitoring HTTP and HTTPS connections. We assume that clients mostly communicate on both protocols. Therefore, we look for HTTP and HTTPS connections from the same client over a short time period and pair cipher suites and User-Agents from such connections.

Third, we used the pairs of SSL/TLS fingerprints and User-Agents as a dictionary to assign User-Agents to the HTTPS connections observed during the measurement. We shall discuss the quality of the obtained pairs with respect to the dictionary size and accuracy of the User-Agent estimation. The goal of this part is to estimate the size and accuracy of a dictionary which could be used for identifying clients on a large-scale and classifying network traffic.

C. Paper Organization

This paper is divided into seven sections. Section II presents related work and a brief introduction into SSL/TLS protocols. The experiment design, measurement tools, and measurement environment are described in Section III. The results are presented in Section IV and evaluated in Section V. The applicability of the results, with respect to the proposed examples, is discussed in Section VI. Finally, Section VII concludes the paper.

II. State-of-the-art

In this section we shall first present an introduction to SSL/TLS protocols and a survey of network-based SSL/TLS analysis. Then we will present a short survey of case studies in network forensics and related fields.

Transport Layer Security (TLS) [3] is a new version of the Secure Sockets Layer version 3 (SSLv3) protocol [4], which is no longer recommended for use due to its security vulnerabilities. It provides confidentiality, data integrity, non-repudiation, replay protection, and authentication through digital certificates directly on top of the TCP protocol. The TLS protocol is currently used for securing the most common network protocols, such as HTTP, FTP, and SMTP, and is part of Voice over Internet Protocol (VoIP) and Virtual Private Network (VPN) protocols. In this paper, we shall focus on SSL/TLS's use within the HTTP protocol, known as HTTPS [5], which is the most common use of the TLS.

The TLS connection can be divided into two phases: an initial handshake and application data transfer, both depicted in detail in Fig. 3. The initial handshake begins with a *ClientHello* message identifying which protocol version is used, the cipher suite list, and extensions. The full list of these identifiers is available on the IANA web page [6]. The following messages of the initial handshake are used for peer authentication using X.509 certificates [7] and shared secret establishment based on agreed parameters. All TLS messages exchanged during the initial handshake are not encrypted until shared keys are established and confirmed by *Finished* messages. The TLS protocol consists of configurable cryptographic algorithms and different sub-protocols which form a layered design [8]. The main part of this design is the Record Protocol [3], which is used as an envelope for all TLS messages and encrypted application data.

The long-term monitoring and measurement of SSL/TLS traffic in the Internet was presented by Levillain et al. [9] for the period 2010–2011. They observed a lot of servers which were intolerant to some cipher suite lists, and detected certificate chains which did not comply with the standard. Another study of SSL traffic was conducted by Holz et al. [10], who focused on SSL/TLS certificate properties. They revealed a great number of invalid certificates and certificates shared among a large number of hosts. The work of Holz et al. was followed by the work of Durumeric et al. [11] which focused on an assessment of certification authorities.

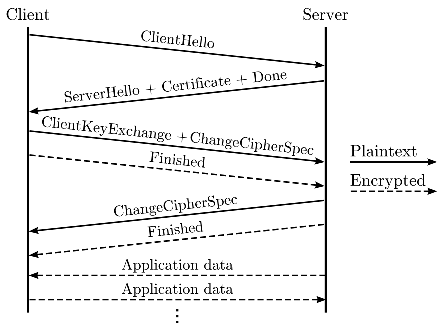

Fig. 3. A SSL/TLS handshake.

The SSL/TLS protocol and its applications are continuously analysed by Qualys SSL Lab [12]. In addition to SSL/TLS applications testing, they present the idea of HTTP client fingerprinting using an analysis of the SSL/TLS handshake. The idea is represented by the SSLhaf [13] proof-of-concept tool for a simultaneous host-based analysis of HTTP and SSL/TLS connections. The idea was also implemented by Majkovski [1] as the *p0f* tool module used for fingerprinting operating systems. Another idea was presented by Bernaille and Teixeira [14], who identify underlying applications in a SSL encrypted connection based on the first SSL/TLS packet size.

The case studies of network forensic analysis, including analysis based on User-Agent identification, were presented by Raftopoulos and Dimitropoulos [15]. They cite Win32/Hotbar as an example of malware, whose activity can be detected by searching for HTTP requests with a specific User-Agent. One related problem is the identification of network address translation (NAT) in the network. A traffic flow-based method was proposed by Gokcen et al. [16] and Krmíček et al. [17].

III. Experiment Design

We designed a three-phase experiment to answer our research questions and verify the idea of using HTTPS client identification with SSL/TLS fingerprinting. In the first phase, we set up measurement of live network traffic in the campus network of Masaryk University. The monitoring was primarily focused on SSL/TLS connections. In the second phase, we created a dictionary of SSL/TLS fingerprints and HTTP User-Agents, based on an analysis of the captured network traffic. In the third phase, we applied this dictionary to assign User-Agents to the measured traffic and verify the capabilities of HTTPS client identification.

A. SSL/TLS Traffic Measurement

We measured live network traffic in the campus network of Masaryk University. The network has more than 40,000 users and 15,000 active IP addresses on average per day. We used a flow-based network probe deployed in a 10 Gbps link that connects the university and the network of CESNET, Czech National Research and Education Network (NREN).

The measured network flow represents one-way network communications defined as a series of packets with a shared set of L3/L4 parameters: protocol, IP addresses, and port numbers [18]:

$$F = \{\, proto,\ srcIP,\ dstIP,\ srcPort,\ dstPort \,\}$$

Since the standard flow does not contain detailed information about HTTP and HTTPS traffic, we used two extensions for flow measurement, which add new elements to the flow record. The first extension adds a User-Agent (ua) element to the flow, based on the analysis of the HTTP traffic [19]:

$$F_{HTTP} = \{\, F \cup \{ua\} \mid dstPort = 80 \ \wedge\ ua \neq \emptyset \,\}$$

The second flow measurement extension adds elements from the *ClientHello* message exchanged during the initial SSL/TLS handshake of the HTTPS connection. We measured only those elements which do not change with each client connection, namely the SSL/TLS protocol version (vr), cipher suite list (cs), compression (cm) and TLS extensions (ex):

$$Hello = \{\, vr,\ cs,\ cm,\ ex \,\}$$
$$F_{HTTPS} = \{\, F \cup Hello \mid dstPort = 443 \ \wedge\ cs \neq \emptyset \,\}$$

The aim of the measurement is to get the base data for the subsequent phases of the experiment. In the next phase, we created a dictionary wchich allows us to transform elements of the SSL/TLS fingerprint into HTTP User-Agents. This dictionary was then applied to all the measured data to verify its usability and gain more information about HTTPS clients. The secondary aim of the measurement is to get a closer look at SSL/TLS connections and to obtain basic statistics about network traffic, primarily focused on SSL/TLS traffic.

B. Pairing Cipher Suite Lists and User-Agents

To identify HTTPS clients, it is necessary to create a dictionary containing pairs of SSL/TLS handshake elements and User-Agents. This represents the second phase of our experiment. We decided to use only a cipher suite list from the *ClientHello* message to build up a dictionary. Cipher suite lists are the most varied elements of the SSL/TLS handshake and we suppose that they should be sufficient for identifying clients. Other elements of the handshake only have a few different values, therefore we do not plan to include them in the dictionary. However, we assume they could clarify ambiguous results.

We take two approaches, host-based and flow-based, to pair a cipher suite list to a User-Agent. The host-based method uses the information from a single HTTPS connection on the server side, where the unencrypted data including the HTTP header, are available. This method is very accurate, but it requires clients to visit the server where the monitoring is deployed. We set up a HTTPS server running Apache web server and SSLhaf plugin [13]. SSLhaf enables us to log the SSL/TLS parameters of a HTTPS connection. We logged

SSL/TLS connection parameters, including the cipher suite list in a *ClientHello* message, and the User-Agent from the HTTP header for each incoming connection. The dictionary is created with a simple combination of the cipher suite list and the User-Agent from a single connection.

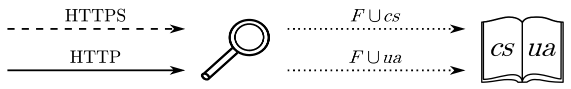

Fig. 4. A flow-based cipher suite list and User-Agent pairing.

The flow-based method is based on network monitoring, the extraction of cipher suite lists and User-Agents, and the correlation of HTTP and HTTPS connections from a single client, see Fig. 4. We assume that web clients commonly communicate via both HTTP and HTTPS protocols. SSL/TLS connections monitoring, as well as HTTP monitoring, is utilized in this phase of the experiment. The method of pairing cipher suite lists and User-Agents is described as follows:

$$\begin{aligned} Dict \;=\; \{\; (cs,\; ua) \mid \exists\; F_{HTTPS},\; \exists\; F_{HTTP} :\\ F_{HTTPS}.cs = cs \wedge F_{HTTP}.ua = ua \;\wedge\\ F_{HTTPS}.srcIP = F_{HTTP}.srcIP \;\} \end{aligned}$$

We searched for HTTP and HTTPS connections with the same source IP address. We selected a cipher suite list from the HTTPS connections and paired it to the User-Agent from the HTTP connection which was the closest in time. We assume that the flow-based approach would better reflect the structure of live network traffic and allow us to cover more cipher suite lists observed in the network.

We didn't expect that the dictionary would provide an unambiguous translation of one cipher suite list to one User-Agent, but there would be the one cipher suite list with more User-Agents and vice versa. However, we assume that User-Agents assigned to the one cipher suite list will only have slight differences, such as the software version. Therefore, it does not affect the identification of general properties, e. g., the operating system or web browser. We also expected there to be some significant deviations caused by ambiguity in the flow-based approach or, for example, by forged connections by a malicious crawler pretending to be a legitimate search engine. In this case, we took only the most similar User-Agents substrings.

C. Assigning User-Agents to Measured HTTPS Flows

The third phase of our experiment is a conjunction of the results from the previous phases, as depicted in Fig. 5. We combined both types of pairs of cipher suite lists and User-Agents which were generated in the second phase. Then we applied them to the measured data from the first phase. The result was HTTPS flows extended by information about the corresponding User-Agent or list of User-Agents:

$$F'_{HTTPS} = \{\; F_{HTTPS} \cup \{ua\} \mid (F_{HTTPS}.cs, ua) \in Dict\}$$

We plan to validate the results of the assignment and verify our idea of HTTPS client identification using SSL/TLS fingerprinting. We are interested in the share of SSL/TLS traffic for which we are able to assign a correlating User-Agent. The connections containing cipher suite lists for which we failed to assign a User-Agent are potentially relevant from a security perspective. We expected most of the traffic to be initiated by common web browsers, for which we are able to easily get the pair of a cipher suite list and User-Agent using the host-based method. However, we expected that a combination of host-based and flow-based dictionaries are needed to cover the majority of the traffic.

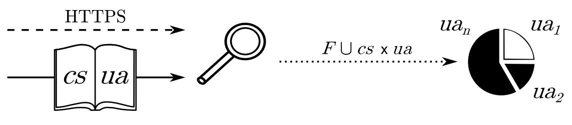

Fig. 5. Assigning User-Agents to cipher suite lists.

If multiple User-Agents which correspond to a single cipher suite list are found, we plan to evaluate what the minimal set of information provided by the set of User-Agents is. For example, when multiple User-Agents, which correspond to the same cipher suite list, point to different versions of a client application. Similarities in grouped User-Agents can indicate at least if the client is web browser, what its operating system is, or if it is a mobile device. Major differences would otherwise indicate an error in the pairing method.

IV. Experiment Results

In this section, we shall present the results of the experiment. First, we will sum up the results of measuring SSL/TLS connections in the campus network. Second, we will describe the set of User-Agents and cipher suite list pairs obtained via the host-based and flow-based methods. The section closes with the results of assigning User-Agents to the SSL/TLS connection obtained in the first phase of the experiment.

A. Measurement

We conducted measurements over a 7 day period in January 2015. We filtered the HTTPS connections, processed the SSL/TLS handshakes, and saved the content of *ClientHello* messages. The SSL/TLS version, cipher suite list, compression, and extensions were recorded for each connection. In total, we processed 85,250,090 HTTPS connections.

TABLE I
Distribution of SSL/TLS versions

Version	Number of Connections
TLS 1.2	49,140,929
TLS 1.0	33,827,182
SSL 3.0	1,365,409
TLS 1.1	913,014
other	3,556

The observed versions are listed in Table I. Over 57 % of connections used the TLS 1.2 protocol followed by almost 40 % for TLS 1.0. Only 1.6 % of connections used the older and more vulnerable SSL 3.0 protocol. TLS 1.1 represented around 1 % of connections. The remaining connections were unrecognized. However, the number of such connections is insignificant.

We can confirm that only a small number of cipher suites and cipher suite lists cover the majority of live network traffic. As we can see in Fig. 6, the top 10 cipher suite lists represent 68.5 % of live network traffic and top 31 (out of 1598) cipher suite lists are enough to cover 90 % of the traffic.

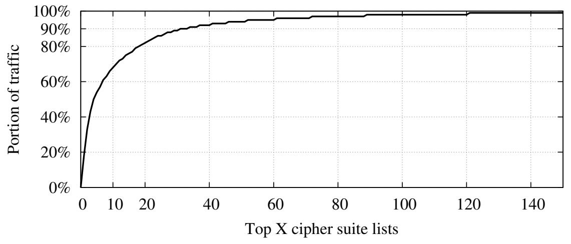

Fig. 6. Network traffic represented by Top X cipher suite lists.

B. Pairing Cipher Suite Lists and User-Agents

First, we created a base set of pairs using the host-based method and SSLhaf. We manually contacted the monitoring server with common available clients, such as web browsers and tools such as *curl* [20], to create an initial dataset. We then made the server publicly accessible and spread the links to lure more clients, such as web crawlers. In total, we obtained 72 unique cipher suite lists and 293 unique User-Agents, forming 307 pairs. Multiple User-Agents, with the same cipher suite list, were similar in most cases. The differences were usually in the version of the client in the User-Agent.

Then, we moved on to the flow-based method, i. e., combined monitoring of cipher suite lists from SSL/TLS handshakes in HTTPS connections and the User-Agents from HTTP connections. We analysed a 1-hour sample of peak network traffic from our campus network and selected the hosts which initiated both HTTP and HTTPS connections. User-Agents from HTTP connections, and cipher suite lists (from HTTPS connections), from the same client created a new pair. We observed 10,890 clients communicating on both protocols in a short period of time, 305 unique cipher suite lists, and 5,043 unique User-Agents. In total, we derived 12,832 unique pairs during the measurement.

Following this, we investigated the relationship between cipher suite lists and User-Agents by determining the cardinality of the relationship. Both methods provided more User-Agents which correspond to one cipher suite list, i. e., a 1:n relation. After a manual inspection, we discovered, that these User-Agents differ mostly in the system versions while the information about the client, e. g. browser type, stays more or less constant. Therefore, it is possible to identify a client with high accuracy. The flow-based method also generated a single User-Agent which corresponds to more cipher suite lists. However, this is most likely caused by inaccuracy in the method which cannot distinguish more clients communicating at the same time.

C. Assigning User-Agents to Measured HTTPS Flows

We first used the dictionary provided by the host-based method and then filled in the supplemented results with a dictionary provided by the flow-based method. The host-based dictionary contains only 72 unique cipher suite lists, which represents 4.5 % of all cipher suite lists measured during the first phase. However, we observed that those unique cipher suite lists cover 78.0 % of all the measured HTTPS flows. When we combined the host-based and flow-based dictionaries, we obtained 316 unique cipher suite lists (19.8 % of all) covering 99.6 % of the measured HTTPS connections. Therefore, we assigned a User-Agent to almost all observed HTTPS connections using a combined dictionary based on data from a single server, and the correlations from a 1-hour sample of network traffic.

As we already mentioned, multiple User-Agents were assigned to one cipher suite list, which causes ambiguity in the translation. Even hundreds of different User-Agents were found to correspond to a single cipher suite list. However, we discovered that multiple User-Agents assigned to a single cipher suite list differ only in details, such as the version of the software used. Fig. 7 shows the results of the User-Agent assignment to the measured HTTPS connections according to the level of certainty. Certainty represents the number of User-Agents per one cipher suite list.

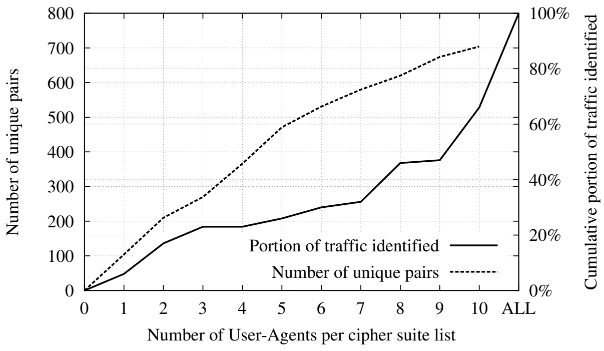

Fig. 7. Relation of dictionary size and covered portion of network traffic.

The results show that in the event of unambiguous pairing, i. e., one User-Agent per one cipher suite list, the dictionary contains 104 unique pairs and covers only 6.3 % of all the HTTPS flows measured. If we gradually decrease the level of certainty, we are able to cover more cipher suite lists and a greater proportion of HTTPS flows. If we use up to 10 User-Agents per one cipher suite list, we are able to cover 66.0 % of all HTTP flows, using 704 unique pairs with 253 unique cipher suite lists. In this case, User-Agents are relatively different, nevertheless, we are able to derive a general identification from the client, e. g., if it is a web browser, mobile device, or web crawler.

V. Experiment Evaluation

In this section, we shall discuss the experiment's circumstances and results. First, we evaluate the measurement phase and shares of protocol versions and cipher suite lists in the observed network traffic. Second, the quality of dictionaries used for client identification is discussed. Methods are proposed which can be used to increase the accuracy of the dictionary. Finally, we shall evaluate the structure of the network traffic according to the estimated User-Agents and compare our results to the related work to estimate the credibility of our results.

A. Measurement

The plain measurement results were similar to our expectations. We analysed the shares of SSL/TLS versions and cipher suite lists in the monitored connections. It is not surprising that the majority of the HTTPS connections use the latest TLS 1.2 protocol. However, such a high share of TLS 1.0 should not remain unnoticed. We further confirmed that the majority of SSL/TLS connections are represented only by a small number of cipher suites and cipher suite lists.

The interesting figure is the 1.6 % share of SSL 3.0 in the observed HTTPS connections. The SSL 3.0 protocol is no longer considered safe due to serious vulnerabilities discovered in 2014, such as the POODLE attack [2]. We naturally wonder if the discovery of serious vulnerability in a protocol leads to a decrease in its usage. In comparison to earlier results, the share of connections initiated over SSL is decreasing. For example, Levillain et al. [9] reported 5 % share of SSL 3.0 in 2011.

B. Pairing Cipher Suite Lists and User-Agents

The host-based and flow-based method of pairing cipher suite lists and User-Agents provided diverse, but complementary, results. The host-based method was more precise and feasible in a controlled environment. However, the quantity of data depends on the popularity of the server where the monitoring is deployed. It could be interesting to perform host-based monitoring on a popular web server with a variety of clients. However, even the high attractiveness of a server does not guarantee capturing traffic from all the common clients in the network. Client applications like Spotify and Instagram, which communicate only to specific servers, are typical examples.

The flow-based method, focusing directly on clients, provided more pairs than the host-based method but at the cost of uncertainty. However, clients like web crawlers, uncommon clients and even suspicious ones, are hard to capture without access to a live network. The 1-hour sample of network traffic was sufficient to capture connections from almost all the different clients observed over a longer period of time.

The combination of both methods provided a usable dictionary which was sufficient for the needs of our experiment. The pairs obtained via the host-based method were also obtained via the flow-based method, which suggests that the flow-based method can provide acceptable results.

C. Assigning User-Agents to the Measurement Results

The assignment of User-Agents to the observed HTTPS connections was successful. The size of the dictionary was sufficient to describe almost every cipher suite list observed during the measurement. The number of connections with an unknown cipher suite list was negligible. The accuracy of the assignment can be disputed because more User-Agents corresponded to a single cipher suite list. However, multiple User-Agents with the same cipher suite list are typically similar.

To improve the quality and accuracy of the assignment, we have to improve the dictionary. We do not expect to gain more results from the host-based method without distributing the measurement among more attractive HTTPS servers. However, we can improve the quality of results in the flow-based method. First, we can manually process the multiple User-Agents assigned to a single cipher suite list and select the minimal common identifier in the event of their similarity, e. g., only the name of the web browser instead of its name, version, and operating system. Second, we can repeat the measurement to get a larger set of pairs and enrich the dictionary by the most statistically significant ones, e. g., the pairs that appear repeatedly.

VI. Discussion

In this section we shall discuss the data which can be derived from a cipher suite list (and its corresponding User-Agent) and their application. We will also present a breakdown of the dictionary according to identifiers found in User-Agents, e. g., types of a client application or a device. An application of the results is demonstrated on the case studies outlined in Section I.

A. Grouping of Client Types

As our experiment shows, we can assign a User-Agent to a known cipher suite list. However, the assignment is not exact as there are typically multiple User-Agents which correspond to a single cipher suite list. The multiple User-Agents are typically similar and we can extract common information, e. g., a type of client, from them.

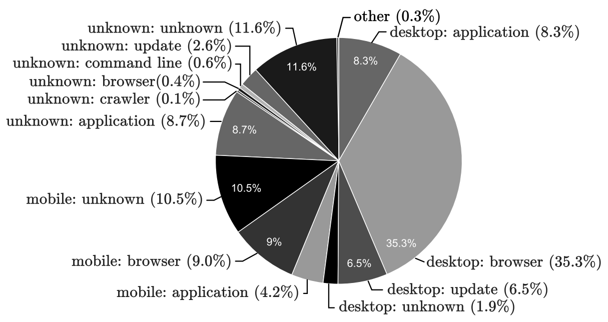

Fig. 8. Shares of HTTPS client types in the dictionary.

We extracted two pieces of information from the User-Agents, device type and application type. First, we distinguished three device types: desktop, mobile, and unknown device. Second, we distinguished five application types: web browser, GUI application, command line application, automatic update, and unknown application. In both cases, the unknown value means that we are not able to determine the device or application type from the given User-Agent. The final client type is a concatenation of device and application type. Major client types, such as desktop browsers, could have been further split into subtypes for particular browsers, however, the results of such fine-grained classification are not presented.

The share of client types in the dictionary is presented in Fig. 8. This figure represents only the structure of a dictionary, not the relevance of particular client types. However, we can see significant shares of client types which are hard to detect using a host-based pairing method. Desktop and mobile applications typically communicate only to specific servers with a specific service. This demonstrates the contribution of the flow-based pairing method.

The shares of client types in the live network traffic are presented in Fig. 9. As we can see, more than half of the connections were initiated by desktop browsers. Desktop, mobile, and unknown device browsers together stand for the vast majority of the network traffic, which is as could be expected. One interesting figure is the relatively high amount of traffic initiated by mobile applications.

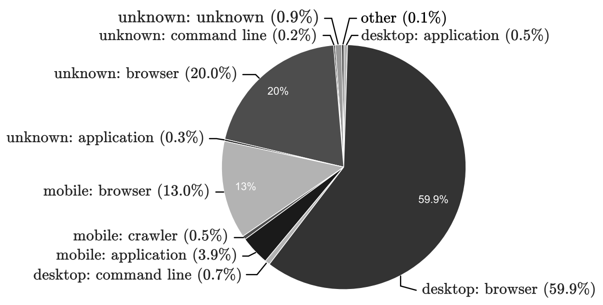

Fig. 9. Shares of HTTPS client types in a live network traffic.

B. Application of SSL/TLS Fingerprinting on Case Studies

In the first example, we have to identify clients which access a server. We do not have access to the server and the server provides only HTTPS. Therefore, the only option is to fingerprint the HTTPS connections. Although the correlation between a cipher suite list and a User-Agent is not exact, we can identify common legitimate traffic and highlight potentially malicious connections. For example, if a server is hosting a common web page, legitimate clients are web browsers and other GUI clients. The presence of connections initiated by command line tools indicates unusual traffic, e. g., connections initiated by malware and an automated communication (click fraud). Conversely, a GUI client that connects to dedicated devices, e. g., SIP devices and hosts in a technological network, indicate potentially malicious traffic.

In the second example we have to enumerate and identify clients behind a NAT. Enumeration and identification of operating systems behind a NAT is possible via TCP/IP fingerprinting [21]. However, an analysis of User-Agents provides better results, which may even lead to the identification of individual users. As in the previous example, a significant share of HTTPS traffic makes the User-Agent analysis unsuitable. The clients do not have to use HTTP at all, or the monitoring time window can be so small that there is no HTTP traffic. Using our proposed approach, we can assign the User-Agent to a HTTPS cipher suite list or at least estimate which type of a client it is. We are thus able to detect the activity of users with a specific web browser or detect rogue devices, e. g., mobile devices in a network where only desktops should appear.

VII. Conclusion

In this paper, we have shown that it is possible to estimate the User-Agent of a client in HTTPS communication. This is done for further identifying the client using network monitoring and fingerprinting the SSL/TLS handshake, which is the main contribution of this paper. We designed an experiment in which we measured HTTPS traffic in a campus network. We processed only the initial SSL/TLS handshake in which the client and server negotiate the parameters of the encryption. Therefore, our approach is lightweight and avoids decrypting traffic.

First, we investigated the parameters of the SSL/TLS handshake, which can be used to identify the client. The client identifies itself in a *ClientHello* message during the handshake. The most varied part of the *ClientHello* was the list of cipher suite lists supported by the client. The cipher suite list differs among various client applications and their versions, which makes them suitable for further identification. In total, we observed 305 unique cipher suite lists during our measurement. The other parts of the *ClientHello* message, such as the SSL/TLS version, compression, and supported extensions, are interesting for analysis, but unusable for client identification due to the limited number of distinct values.

Second, we studied the relationship between SSL/TLS cipher suite lists and HTTP User-Agents. The User-Agent is a common client identifier in HTTP. However, in HTTPS, it is not accessible without decrypting the transferred data. We deployed two methods for monitoring SSL/TLS handshakes and HTTP headers simultaneously in order to pair cipher suite lists and User-Agents. The host-based method, i. e., measurement on the server side, provided accurate results. However, this method is limited by the set of clients accessing the monitoring server and we obtained a smaller number of pairs. The flow-based method uses network monitoring and is not limited to a single server. We were looking for clients communicating on HTTP and HTTPS protocols over a short period of time, and paired the observed cipher suite lists and User-Agents from both connections. We gained a large dictionary of more than 12,000 pairs. However, this method is less accurate compared to the host-based method.

Third, we assigned the corresponding User-Agents from the dictionary to the results from monitoring the SSL/TLS connections and discussed the required size and accuracy of the dictionary. We found that we need a dictionary of about 300 cipher suite lists with assigned User-Agents. Therefore, the dictionary which was created using the host-based method was not sufficient to cover all the distinct cipher suite lists which appear in network traffic. On the other hand, only a 1-hour sample of the HTTPS traffic contained almost all the cipher suite lists which were observed over the week-long measurement. This led us to use the dictionary created via the flow-based method. However, many cipher suite lists were paired with more than one User-Agent. We were able to assign a User-Agent to almost every observed cipher suite list with a certain level of probability. Fortunately, in many cases a lot of User-Agents which corresponded to a single cipher suite list share the same client identifier, and differ only in their version or a similarly attainable value.

In conclusion, our work enhances the capabilities of network forensics by introducing the network-based identification of HTTPS clients. Our network-based approach is lightweight, not limited to a single server, and does not approach the encrypted data. Therefore, we can identify clients while preserving the communication's privacy. Our results are applicable for identifying clients in the network, detecting the activity of a specific client, and breaking down the structure of HTTPS traffic in a whole network. This was demonstrated in the experiment and two case studies of network forensics.

Acknowledgement

This material is based on work supported by the Security Research for the Needs of the State 2010–2015 programme funded by the Ministry of the Interior of the Czech Republic.

References

[1] M. Majkowski, "SSL fingerprinting for p0f," Web page, June 2012, accessed 2015-01-28. [Online]. Available: https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f/

[2] B. Möller, T. Duong, and K. Kotowicz, "This POODLE Bites: Exploiting The SSL 3.0 Fallback," PDF online, 2014, accessed 2015-01-12. [Online]. Available: https://poodlebleed.com/ssl-poodle.pdf

[3] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176.

[4] A. Freier, P. Karlton, and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," RFC 6101 (Historic), Internet Engineering Task Force, Aug. 2011.

[5] E. Rescorla, "HTTP Over TLS," RFC 2818 (Informational), Internet Engineering Task Force, May 2000, updated by RFCs 5785, 7230.

[6] IANA – Internet Assigned Numbers Authority, "Protocol Registries," Web page, 2014, accessed 2015-01-28. [Online]. Available: http://www.iana.org/protocols

[7] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280 (Proposed Standard), Internet Engineering Task Force, May 2008, updated by RFC 6818.

[8] C. Meyer, "20 Years of SSL/TLS Research: An Analysis of the Internet's Security Foundation," Ph.D. dissertation, Ruhr-University Bochum, February 2014, accessed 2015-01-15. [Online]. Available: http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/MeyerChristopher/diss.pdf

[9] O. Levillain, A. Ébalard, B. Morin, and H. Debar, "One Year of SSL Internet Measurement," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 11–20.

[10] R. Holz, L. Braun, N. Kammenhuber, and G. Carle, "The SSL Landscape: A Thorough Analysis of the x.509 PKI Using Active and Passive Measurements," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 427–444.

[11] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS Certificate Ecosystem," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 291–304.

[12] Qualys SSL Lab, "HTTP Client Fingerprinting Using SSL Handshake Analysis," Web page, 2014, accessed 2015-01-23. [Online]. Available: https://www.ssllabs.com/projects/client-fingerprinting/

[13] I. Ristić, "Passive SSL client fingerprinting using handshake analysis," GitHub repository, 2014, accessed 2015-01-30. [Online]. Available: https://github.com/ssllabs/sslhaf

[14] L. Bernaille and R. Teixeira, "Early Recognition of Encrypted Applications," in *Passive and Active Network Measurement*, ser. Lecture Notes in Computer Science, S. Uhlig, K. Papagiannaki, and O. Bonaventure, Eds. Springer Berlin Heidelberg, 2007, vol. 4427, pp. 165–175.

[15] E. Raftopoulos and X. Dimitropoulos, "Understanding network forensics analysis in an operational environment," in *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE, 2013, pp. 111–118.

[16] Y. Gokcen, V. A. Foroushani, and A. Heywood, "Can we identify NAT behavior by analyzing Traffic Flows?" in *Security and Privacy Workshops (SPW), 2014 IEEE*. IEEE, 2014, pp. 132–139.

[17] V. Krmíček, J. Vykopal, and R. Krejčí, "Netflow Based System for NAT Detection," in *Proceedings of the 5th International Student Workshop on Emerging Networking Experiments and Technologies*, ser. Co-Next Student Workshop '09. New York, NY, USA: ACM, 2009, pp. 23–24.

[18] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.

[19] P. Velan, T. Jirsík, and P. Čeleda, "Design and Evaluation of HTTP Protocol Parsers for IPFIX Measurement," in *Advances in Communication Networking*, T. Bauschert, Ed., vol. 8115. Heidelberg: Springer Berlin Heidelberg, 2013, pp. 136–147.

[20] cURL Contributors, "cURL - command line tool and library for transferring data with URL syntax," 2015, accessed 2015-01-25. [Online]. Available: http://curl.haxx.se/

[21] R. Beverly, "A Robust Classifier for Passive TCP/IP Fingerprinting," in *Passive and Active Network Measurement*, ser. Lecture Notes in Computer Science, C. Barakat and I. Pratt, Eds. Springer Berlin Heidelberg, 2004, vol. 3015, pp. 158–167.