Husák *et al. EURASIP Journal on Information Security* (2016) 2016:6
DOI 10.1186/s13635-016-0030-7

EURASIP Journal on Information Security
a SpringerOpen Journal

RESEARCH Open Access

# HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting

Martin Husák, Milan Čermák, Tomá Jirsík* and Pavel Čeleda

**Abstract**

The encryption of network traffic complicates legitimate network monitoring, 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. The dictionary was used to classify live HTTPS network traffic. We were able to retrieve client types from 95.4 % of HTTPS network traffic. Further, we discussed host-based and network-based methods of dictionary retrieval and estimated the quality of the data.

**Keywords:** Network monitoring, HTTPS, User-Agent, SSL, TLS, Fingerprinting

## 1 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 analyze 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, malicious network behavior can be hidden in encrypted connections, where it is invisible to detection mechanisms.

In this paper, we shall discuss HTTPS-HTTP over SSL/TLS, the most common encrypted network traffic protocols. In a communication encrypted by SSL/TLS, the hosts have to first agree on encryption methods and their parameters. Therefore, the initial packets contain unencrypted messages with information about the client and server. This information varies among different clients and their versions. The 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.

### 1.1 Research questions

We set up an experiment, which will answer three research questions. The fourth question addresses applicability of results. To sum up our goals, the research questions are:

(i) *Which parameters of a SSL/TLS handshake can be used for client identification?*
(ii) *Can we pair selected SSL/TLS handshake parameters and HTTP header fields?*
(iii) *How much information, i.e., number of known SSL/TLS parameters and pairings to HTTP headers, do we need to analyze a significant portion of network traffic?*
(iv) *Can we utilize the SSL/TLS fingerprinting in network security monitoring and intrusion detection?*

*Correspondence: jirsik@ics.muni.cz
Institute of Computer Science, Masaryk University, Brno, Czech Republic

First, we aim to observe real network traffic to gain insight into contemporary SSL/TLS handshakes. We shall deploy network traffic monitoring, filter HTTPS connections, and create a list of the SSL/TLS handshakes and their fingerprints. We focus on analyzing 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 [1], might have significantly changed the proportion of protocol versions in use.

Second, we shall correlate 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 suite lists and corresponding User-Agents. The host-based approach is based on advanced logging on the server side. The novel network-based method is based on simultaneous monitoring of 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 shall use 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.

Fourth, we are interested in application of the results in the area of network security monitoring and intrusion detection. Network hosts exhibiting suspicious or even malicious intentions appear on a daily basis. Assuming that the malicious applications are designed for a specific purpose, their fingerprints may be different from legitimate clients. Utilizing fingerprinting as a detection method of malicious clients may prove itself suitable in a situation when even the botnets are switching their communication to HTTPS.

### 1.2 Paper organization

This paper is divided into eight sections. Motivation for this work is stated in Section 2. Section 3 presents related work and a brief introduction into SSL/TLS protocols. The experiment design, measurement tools, and measurement environment are described in Section 4. The results are presented in Section 5 and evaluated in Section 6. The applicability of the results, with respect to the proposed examples, is discussed in Section 7. Finally, Section 8 concludes the paper.

## 2 Motivation

The motivation for our work came from three main areas of interest. Firstly, it is the analysis of encrypted network traffic in general, including identification and characterization of encrypted network traffic and classification of communicating clients. From this point of view, we are not interested in individual clients but rather overall characteristics of network traffic. The second area of interest is the client identification, which aims to obtaining information on an individual client. This applies mostly on a web browser fingerprinting. The third area is network security and forensics, where we typically want to detect activity of a specific network host, detect malicious clients, and evaluate the activity of unknown or unusual clients.

### 2.1 Analysis of encrypted network traffic

We have to understand the network traffic before we can proceed to client identification and detection of suspicious or even malicious activity. Therefore, we have to observe network traffic to get insight into typical patterns. Specifically, in this case, we have to retrieve record of encrypted network traffic containing as much different patterns as possible. To motivate our work, we decided to analyze real network traffic in a campus network instead of generating the traffic patterns in laboratory environment. Therefore, we can get more interesting results which are not necessarily related to the proposed experiment. These results can later be useful for network administrators, security practitioners, and scientific community.

For the purpose of this paper, we have to identify what are the options of establishing the SSL/TLS communication and which options are used in real traffic. We have to use methods from a survey by Velan et al. [2] as the basis and a real network data to identify these options. Then, we have to find which of the options are varying the most and if the variability of these options indicates different traffic patterns, e.g., different communicating partners or type of traffic.

### 2.2 Client identification and browser fingerprinting

Having insight into the network traffic, we can proceed to the client identification and browser fingerprinting. The client identification and browser fingerprinting contribute largely to network security and detection of malicious activities, e.g., by outdated system identification or unusual behavior detection. Identification and fingerprinting are moreover useful for commercial purposes

(targeting ads, price discrimination, assessing financial credibility), network accounting, and client behavior monitoring [3].

Assuming the clients have unique fingerprints, the fingerprinting can be used for advanced traffic analysis. For example, we can enumerate unique client fingerprints that share the same IP address as depicted in Fig. 1. This approach can help in enumerating the number of users of a specific machine, presence of a NAT mechanism and number of clients behind a NAT, etc.

A problem of client identification based on HTTP User-Agent is that the User-Agent string can be easily manipulated. For example, illegitimate web crawlers and bots typically spoof the User-Agent string as to be mistaken for legitimate ones such as Googlebot [4]. Manipulation with User-Agent string does not apply only to malicious clients but also to legitimate clients. Historically, the web browsers were adding identifiers of each other to their User-Agent to resolve compatibility issues with certain web pages. Therefore, the Internet Explorer includes "Mozilla" in its User-Agent string and Android browsers claim themselves to be Safari web browser. Many browsers also offer user-friendly option to completely replace a User-Agent with arbitrary string or identifier of a different browser. To sum it up, there is a constant risk that the User-Agent string is forged and the results of any work based on User-Agent string analysis cannot be trusted.

### 2.3 Network security and forensics

Network security and forensics have their unique preferences and points of view in comparison to network traffic analysis and client fingerprinting. Common network traffic is not a primary interest of network security monitoring. Instead, the abnormal and previously unknown traffic patterns are in question. Of course, we have to first understand the network traffic to recognize common traffic patterns. Then, it is possible to focus on unusual events and, in our case, fingerprints to detect suspicious or straightly malicious clients and their activity.

The case studies in network security and forensics are dealing with malware and vulnerability exploitation. For example, Win32/Hotbar is a malware, whose activity can be detected by searching for HTTP requests with a specific User-Agent [5]. Another example is related to Shellshock, Bash vulnerability disclosed in 2014. Shellshock can be exploited via HTTP requests containing strings starting with "`() { ;; };`" in values of various headers, which could be processed by some script at a server side. The attack can be detected by checking the characteristic sequence in HTTP request headers. In both cases, we face the problem of malware detection method that is easily performed over HTTP but is hard to accomplish over HTTPS traffic.

The overall trends in network communication affect almost all the communicating hosts, including the malicious ones. The prime examples are botnets which use HTTP(S) for communication between bots and command and control centers (C&C). Modern botnets are switching to HTTPS for similar reasons as legitimate clients. For example, the bots are likely to accept commands only from trusted C&C centers. HTTPS provides certain level of trust and, in addition, prevents eavesdropping of the communication, on which the detection mechanisms typically rely [6].

Keeping in mind that almost everything is switching to encrypted communication protocols, we may expect a demand for methods of encrypted traffic analysis for security purposes. Client fingerprinting seems to be a suitable option as it does not interfere with the encrypted content of communication and thus cannot be considered as a violation of privacy. The question, however, is

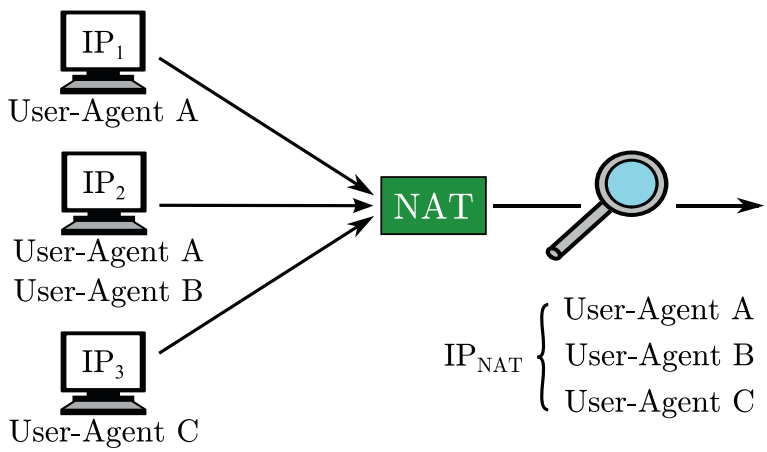


**Fig. 1** Identification and enumeration of HTTPS clients behind NAT

if the the fingerprinting can distinguish legitimate and suspicious clients. For example, malicious actions executed using common a web browser are not any different from legitimate traffic from the fingerprinting perspective. However, the malicious clients are often created for a specific purpose and their implementation complies to this. Therefore, we may assume that certain malicious clients are having unique fingerprints. This applies namely to self-propagating malware and bots communicating within a botnet.

## 3 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 shall present a short survey of case studies in network forensics and related fields.

### 3.1 Introduction to SSL/TLS

Transport Layer Security (TLS) [7] is a new version of the Secure Sockets Layer version 3 (SSLv3) protocol [8], 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 [9], 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. 2. 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 [10]. The following messages of the initial handshake are used for peer authentication using X.509 certificates [11] 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 [12]. The main part of this design is the Record Protocol [7], which is used as an envelope for all TLS messages and encrypted application data.

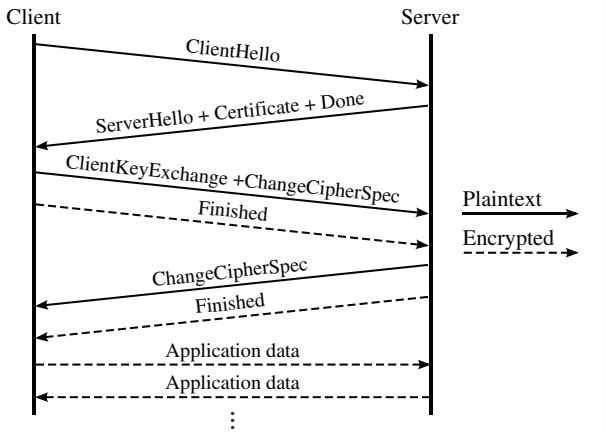


**Fig. 2** A SSL/TLS handshake

### 3.2 Related work

The long-term SSL/TLS traffic monitoring and measurement in the Internet was presented by Levillain et al. [13] 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. [14], 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. [15] which focused on an assessment of certification authorities.

The SSL/TLS protocol and its applications are continuously analyzed by Qualys SSL Lab [16]. Apart from SSL/TLS application testing, they presented the idea of HTTP client fingerprinting using an analysis of the SSL/TLS handshake. The idea was implemented in the SSLhaf [17] proof-of-concept tool for a simultaneous host-based analysis of HTTP and SSL/TLS connections. A brief analysis of fingerprints of common web browsers based on the tool was published at Internet Storm Center [18]. The idea was also implemented by Majkovski [19] as the *p0f* tool module used for fingerprinting operating systems. Another idea was presented by Bernaille and Teixeira [20], who identified underlying applications in a SSL-encrypted connection by the first SSL/TLS packet size.

A survey of web tracking and its mechanisms was proposed by Bujlow et al. [3]. The authors placed a special focus on fingerprinting as it is rich in various methods. The survey proposed a taxonomy of tracking mechanisms including a category of fingerprinting. The approaches relevant to our work are *device fingerprinting*, *operating System instance fingerprinting*, and *browser instance fingerprinting*. The methods are mostly focused on analysis of HTTP headers. However, TCP/IP headers can also be used, e.g., for OS fingerprinting. OS can be detected using information from network flows (TTL, SYN packet size, TCP window size, User-Agent) [21], DNS traffic analysis [22], or using combination of previous and prebuilt dictionary such as in *p0f* tool [23]. Regarding the client

identification, remote psychical client fingerprinting using clock skews was described by Kohno et al. [24]. Graph-based structures for client profiling were introduced by Karagiannis et al. [25]. Manual labeling of fingerprints, a drawback of fingerprint-based methods, was tackled by Abdelnur et al. [26]. The authors intended to solve the problem by an automation of fingerprint signature creation.

Regarding the browser fingerprinting, detection using a User-Agent field is considered not to be reliable, since the field is set by a browser and can be easily forged. Instead, fingerprinting methods based on CSS and HTML5 fingerprinting were introduced by Unger et al. [27]. The JavaScript engine was used for browser identification by Mulazzani et al. [28]. *Panopticlick project* [29] uses active approach in collecting fingerprints of your browser. Collected features are browser plugin details, system fonts, cookie settings, screen size, and others. Nevertheless, all of these techniques can be overcome by using a web proxy [30] or TOR [31]. In these cases, detectable fingerprints are removed or replaced by custom ones.

The case studies of network forensic analysis, including analysis based on User-Agent identification, were presented by Raftopoulos and Dimitropoulos [5]. They cited 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. Traffic flow-based methods were proposed by Gokcen et al. [32] and Krmíček et al. [33].

## 4 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 the 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 verified the capabilities of HTTPS client identification.

### 4.1 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. It is defined as a set of all packets sharing the same values of chosen packet characteristics called flow keys [34]. The flow keys of a standard flow consists of following L3/L4 parameters: protocol, IP addresses, and port numbers. Each flow is represented by a *flow record*:

$$f = (\text{proto, srcIP, dstIP, srcPort, dstPort, tSt}),$$

where proto, srcIP, dstIP, srcPort, and dstPort are the shared values of flow keys and tSt is the timestamp of the flow.

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 a HTTP flow based on the analysis of the HTTP traffic [35]. Only HTTP flows with destination port 80 were considered. The set of all extended HTTP flows with assigned User-Agent will be denoted $F_{\text{HTTP}}$.

$$F_{\text{HTTP}} = \{\, (f,\ \text{ua}) \mid f.\text{dstPort} = 80 \\ \wedge\ \text{ua} \neq \text{null} \,\}$$

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). The set of all extended HTTPS flows will be denoted $F_{\text{HTTPS}}$.

$$\text{Hello} = (\ \text{vr},\ \text{cs},\ \text{cm},\ \text{ex}\ ) \\ F_{\text{HTTPS}} = \{\, (f,\ \text{Hello}) \mid f.\text{dstPort} = 443 \\ \wedge\ \text{Hello} \neq \text{null} \,\}$$

The aim of the measurement was to get the base data for the subsequent phases of the experiment. In the next phase, we created a dictionary which allowed 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 was to get a closer look at SSL/TLS connections and to obtain basic statistics about the network traffic, primarily focusing on SSL/TLS traffic.

### 4.2 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 supposed that they are sufficient for identifying clients. Other elements of the handshake only have a few different values, therefore we did not plan to include them in the dictionary. However, we assumed they could clarify ambiguous results.

We took 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 [17]. SSLhaf enabled us to log the SSL/TLS parameters of a HTTPS connection. We logged the 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 was created by a simple combination of the cipher suite list and the User-Agent from a single connection.

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. 3. We assumed that web clients commonly communicate via both HTTP and HTTPS protocols. SSL/TLS connection monitoring, as well as HTTP monitoring, was utilized in this phase of the experiment. The method of pairing cipher suite lists and User-Agents can be described as follows: let CS denote the set of all possible cipher suite lists and UA the set of all User-Agents, then

$$\begin{aligned}\text{Dict} = \{\, (\text{cs}, \text{ua}) \in \text{CS} \times \text{UA} \mid \\ \exists f_S \in F_{\text{HTTPS}},\ \exists f \in F_{\text{HTTP}} : \\ f_S.\text{cs} = \text{cs} \wedge f.\text{ua} = \text{ua} \\ \wedge\ f_S.\text{srcIP} = f.\text{srcIP} \\ \wedge\ \forall f' \in F_{\text{HTTP}},\ \forall f'_S \in F_{\text{HTTPS}} : \\ |f'.\text{tSt} - f_S.\text{tSt}| \geq |f.\text{tSt} - f_S.\text{tSt}| \\ \wedge\ |f.\text{tSt} - f'_S.\text{tSt}| \geq |f.\text{tSt} - f_S.\text{tSt}| \,\}\end{aligned}$$

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 assumed 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 did not expect that the dictionary would provide an unambiguous translation of one cipher suite list to one User-Agent, but there would be one cipher suite list with more corresponding User-Agents and vice versa. However, we assumed that User-Agents assigned to one cipher suite list have only 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-Agent substrings.

### 4.3 Assigning User-Agents to measured HTTPS flows

The third phase of our experiment was a conjunction of the results from the previous phases as depicted in Fig. 4. 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 results contained HTTPS flows extended by information about the corresponding User-Agent or list of User-Agents:

$$\begin{aligned}F'_{\text{HTTPS}} = \{\, (f, \text{ua}) \in F_{\text{HTTPS}} \times \text{UA} \mid \\ (f.\text{cs}, \text{ua}) \in \text{Dict} \,\}\end{aligned}$$

We planned to validate the results of the assignment and verify our idea of HTTPS client identification using SSL/TLS fingerprinting. We were interested in the share of SSL/TLS traffic for which we were 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 were 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 was needed to cover the majority of the traffic.

If multiple User-Agents corresponding to a single cipher suite list were found, we planned to evaluate what is the minimal set of information provided by the set of User-Agents. For example, when multiple User-Agents

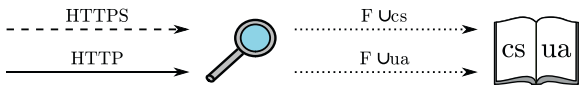


**Fig. 3** A flow-based cipher suite list and User-Agent pairing

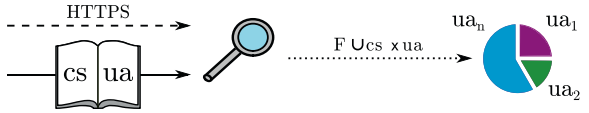


**Fig. 4** Assigning User-Agents to cipher suite lists

corresponding 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 a web browser, what is its operating system, or if it is a mobile device. Major differences would otherwise indicate an error in the pairing method.

## 5 Experiment results

In this section, we shall present the results of the experiment. First, we shall sum up the results of measuring SSL/TLS connections in the campus network. Second, we shall 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.

### 5.1 Measurement

We conducted measurements over a 7-day period in January 2015. The network bandwidth utilization on the monitored network link ranged from 3 to 5 Gbps. 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.

The observed versions are listed in Table 1. 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 unrecognised. 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. 5, the top 10 cipher suite lists represented 68.5 % of live network traffic and top 31 (out of 1598) cipher suite lists were enough to cover 90 % of the traffic.

### 5.2 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* [36], to create an initial dataset. Therefore, most of the traffic incoming to the monitoring server was artificial. 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 analyzed a 1-h 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 5043 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.

### 5.3 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 contained only 72 unique cipher suite lists, which represented 4.5 % of all cipher suite lists measured during the first phase. However, we observed that those unique cipher suite lists covered 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 the data from a single server and the correlations from a 1-h sample of network traffic.

**Table 1** 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 | 3556 |

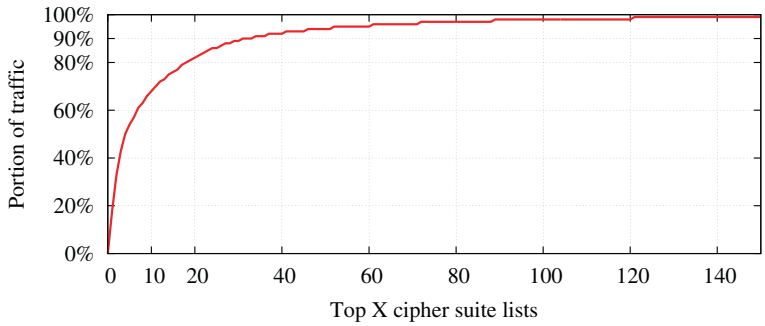


**Fig. 5** Network traffic represented by top 10 cipher suite lists

As we already mentioned, multiple User-Agents were assigned to one cipher suite list, which caused 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 differed in details, such as the version of the used software. Figure 6 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.

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

## 6 Experiment evaluation

In this section, we shall discuss the experiment's circumstances and results. First, we shall 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.

### 6.1 Measurement

The plain measurement results were similar to our expectations. We analyzed 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 used the latest TLS 1.2 protocol. However, high share of TLS 1.0 should not remain unnoticed. We further confirmed that the majority of SSL/TLS connections was represented only by a small number of cipher suites and cipher suite lists.

The interesting figure was the 1.6 % share of SSL 3.0 in the observed HTTPS connections. The SSL 3.0 protocol

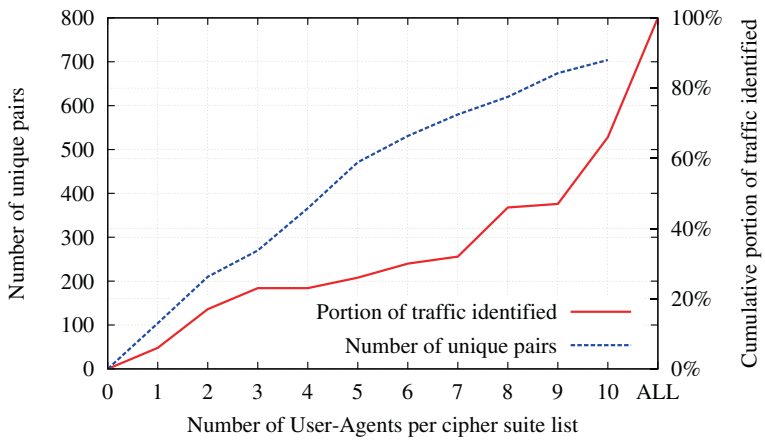


**Fig. 6** Relation of dictionary size and covered portion of network traffic

is no longer considered safe due to serious vulnerabilities discovered in 2014, such as the POODLE attack [1]. We naturally wondered 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. [13] reported a 5 % share of SSL 3.0 in 2011.

### 6.2 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-h 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.

### 6.3 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 were typically similar. For example, similar User-Agents corresponding to a single cipher suite list are presented in Table 2.

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. We identified three approaches to enrich the dictionary and give precision to the data. The options are selecting the most suitable pair to put in the dictionary by manually inspecting the User-Agents or selecting statistically most significant values from repeated measurements and correlation of results to data from other sources.

First, we can manually check the results to find most suitable pairs to put in the dictionary. This approach can significantly reduce the number of User-Agents assigned to a single cipher suite list which differ only in software version or other detail. It can also improve the quality of client type grouping. For example, User-Agents of mobile devices often include type of hardware. However, this is laborious and error-prone.

A more convenient way to improve the quality of the results is to repeat the experiment to get statistically significant data. Our assumption was that the clients communicate on both HTTP and HTTPS protocols in near time. However, the two connections from the same client may be initiated by a different software client. Repeating the experiment in different time windows or even different network settings would provide slightly different results due to random errors and inconsistencies. The resulting dictionary would not be based on a single measurement nor union of all of them. Instead, only the pairs which appear in results of multiple experiments would be added to the resulting dictionary.

Another approach to improving the results is a correlation to additional data. Considering only the network-based method, we can extend the fingerprinting to TCP/IP. Operating system of a client can be estimated by TTL value, TCP SYN packet size, and TCP Window Size [21]. The User-Agent strings often include identifiers of operating systems, which may be correlated to the operating systems estimated by the TCP/IP fingerprinting. In addition, web crawlers can be identified by their hostname or WHOIS record. For example, reverse DNS query may validate User-Agent of clients such as Googlebot.

## 7 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. First, we shall present a breakdown of the dictionary according to identifiers found in User-Agents, e.g., types of a client application or a device. Second, we shall focus on the most interesting type of clients, web browsers, and operating systems, which can be analyzed in more details. Applicability of our results is discussed in the context of browser fingerprinting, network security, and network forensics. We shall also discuss possible defences against SSL/TLS fingerprinting.

**Table 2** User-Agents corresponding to a single cipher suite list

| |
|---|
| Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.132 Safari/537.36 OPR/21.0.1432.67 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36 OPR/22.0.1471.70 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36 OPR/23.0.1522.77 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.94 Safari/537.36 OPR/24.0.1558.53 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2050.0 Iron/38.0.2150.0 Safari/537.36 |
| Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36 Sleipnir/6.1.2 |
| Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36 |
| Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.65 Safari/537.36 |

### 7.1 Classification of network traffic

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 per one cipher suite list. Thus, we can extract common information, e.g., a type of application, from them.

We used HTTP::BrowserDetect tool [37] to mark and classify the User-Agent strings. The tool extracts general information on a given client, e.g., browser name, version, vendor, and operating system. Although the tool was designed to analyze User-Agents of web browsers, it can recognize web crawlers as well. The interesting option is the detection of a mobile device, its type, and vendor.

We extracted four pieces of information from the User-Agents: device type, operating system, application type, and type of a web browser. If the cipher suite list corresponded to more than one User-Agent, we selected the most frequent values. For example, if the cipher suite list corresponded to Chrome four times and to Firefox one time, we assigned the Chrome browser. This method is not the most accurate but provides the most probable value. In case of wide variability of values or inability to parse User-Agent, the *unknown* value is used. The result is a dictionary with concatenated values corresponding to the client fingerprint for every cipher suite list.

The share of client types in the dictionary is presented in Fig. 7. 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. Nevertheless, we were at least able to detect the application types using unknown device type records from the dictionary. Over one fifth of the client types remained unknown both for device and application type, though. Desktop and mobile applications typically communicate only to specific servers with a specific service. This demonstrates the contribution of the flow-based pairing method.

### 7.2 Client identification and browser fingerprinting

The grouping presented in Section 7.1 reflects the structure of clients in the dictionary. To discover the structure of clients in the HTTPS traffic, we assigned dictionary records to the real traffic. Moreover, we performed further analysis on the shares of operating systems in HTTPS traffic and on shares of web browsers in HTTPS traffic in order to gain deeper insight into the encrypted traffic.

The shares of client types in the live network HTTPS traffic are presented in Fig. 8. We observed that a majority of connections were initiated by browsers. This approves the fact that we analyzed HTTPS connections primarily designed for web communication. About a third of the connection were initiated by desktop browsers. One interesting figure is the relatively high amount of traffic initiated by mobile devices. We were also able to capture machine-generated HTTPS connections by identification of traffic belonging to crawlers and updates. Only 4.6 % of the HTTPS traffic remained completely unknown.

The group of browsers, desktop and mobile, still dominated in both categories. Therefore, we picked the browser's part of analyzed HTTPS traffic to perform further analysis in more detail. The results of the analysis are presented in Fig. 9. Nearly one half of the total connections were represented by Chrome web browser, followed by Firefox with one fourth and Internet Explorer with 14.6 % of browser's encrypted traffic.

Described distribution of browsers in HTTPS traffic strongly correlates with the distribution of browsers in national browser usage statistics showed in Table 3 [38]. Even though the dictionary did not provide exact translation, the table shows that our browser fingerprinting method gave accurate results.

In the last analysis of HTTPS traffic, we were not interested in client's application type, but rather in its

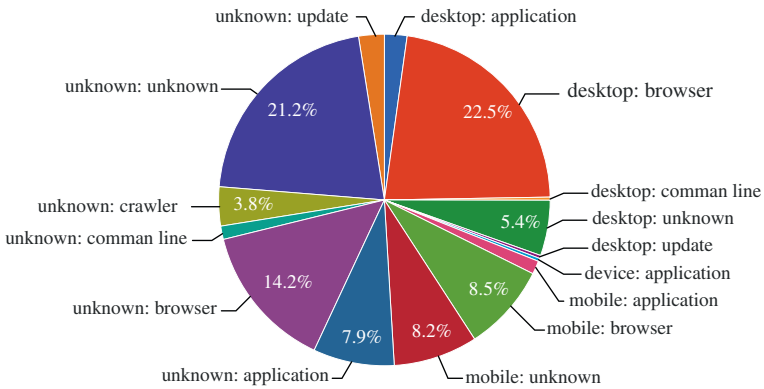


**Fig. 7** Shares of HTTPS client types in the dictionary

implementation, i.e., operating system. The results of the analysis are presented in Fig. 10. We used our dictionary to assign an User-Agent to the cipher suite list of encrypted connection. The User-Agent was then further analyzed to retrieve operating system information. The majority of the operating system represents the Windows platform. Unfortunately, there was a high share of the traffic for which we were not able to determine the client's operating system due to missing or contradictory information in the User-Agent string. The shares of other operating systems were evenly distributed. To obtain a more precise operating system identification, we could use additional information, such as TCP window size or SYN packet size.

### 7.3 Network security and forensics

The next step following the classification of the network traffic and identification of the clients was the detection of specific or unusual clients in the network traffic. The problem arises in the fields of network security and forensics. Network forensics use cases which typically focus on tracing an activity of a specific host in the network. If a client is known to be malicious, e.g., it is a malware, then it is also relevant for network security. Monitoring the activity of clients, which are known or suspected of being malicious, helps in detection and prevention of attacks or malware spreading. However, recognizing malicious activity and marking a client as malicious is a hard problem.

One of the most interesting pairs, from the network security perspective, included characteristic sequence of a Bash vulnerability Shellshock in the User-Agent string. The sequence "`() { ;; };`" appeared in other HTTP headers as well in this case. The string clearly is not a User-Agent, which also implies that we have no other identifier of the client. The fingerprint was obtained via host-based method and is trustworthy. We searched for the corresponding cipher suite list in the network monitoring data but did not find any other match. Therefore, we may assume there is a unique fingerprint of malicious client. Although we do not know more about the client, we can use the knowledge to detect suspicious cipher suite lists in the network traffic. Detection of malicious clients, which

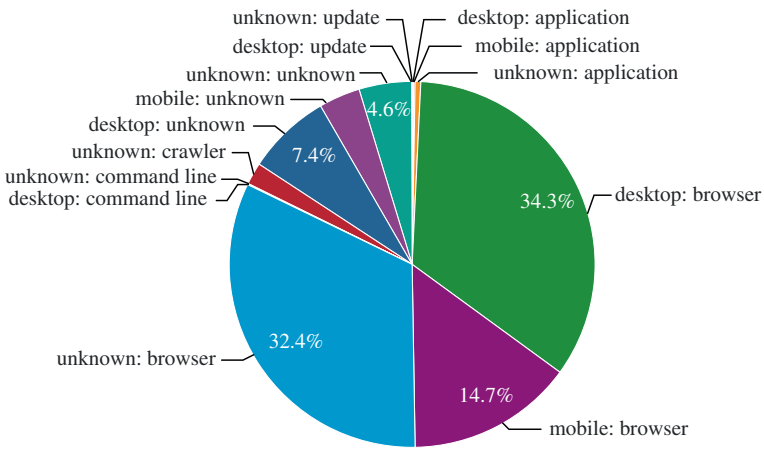


**Fig. 8** Shares of HTTPS client types in live network traffic

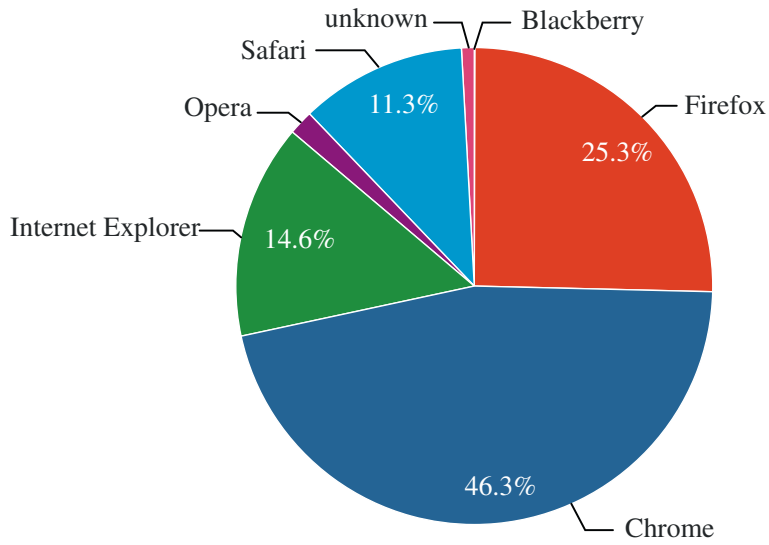


**Fig. 9** Shares of web browsers

are known to exploit a vulnerability, is another use case of fingerprinting in network security.

### 7.4 Defense against fingerprinting

Bujlow et al. [3] surveyed defense strategies against currently known tracking mechanisms, e.g., browser fingerprinting. We are not aware of any method which would prevent SSL/TLS fingerprinting apart from using a proxy or manually changing the cipher suite list.

Using a proxy, for example, mitmproxy [39], causes the fingerprinting method to detect cipher suite list of the proxy instead of cipher suite list of a client. On the other hand, the cipher suite list of a proxy can be recognized and the corresponding network traffic can be marked accordingly.

The second option of defense is manually changing the cipher suite list of a client. This can be done by forced forbidding of certain cipher suites. Then, a client is communicating with reduced cipher suite list. The fingerprinting method cannot recognize the reduced cipher suite list and thus fails at finding a corresponding User-Agent. On the other hand, it is not easy to reduce a cipher suite list to forge cipher suite list of a different client.

**Table 3** Shares of used web browsers for Czech Republic [38]

| Browser | Traffic share [%] |
|---|---|
| Chrome | 41.94 |
| Firefox | 26.39 |
| Internet explorer | 17.27 |
| Safari | 5.52 |
| Opera | 4.59 |
| Other | 4.29 |

## 8 Conclusions

In this paper, we have shown that it is possible to estimate the User-Agent of a client in HTTPS communication. This was 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 was lightweight and avoided decrypting traffic.

First, we investigated the parameters of the SSL/TLS handshake, which could 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 suites supported by the client. The cipher suite list differed among various client applications and

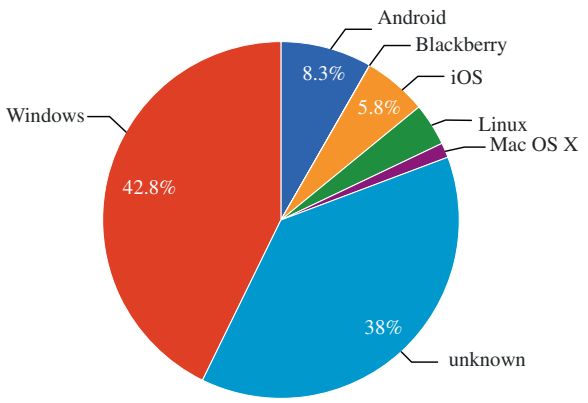


**Fig. 10** Shares of operating systems

their versions, which made it 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, were interesting for analysis but unusable for client identification due to the limited number of distinct values.

Second, we studied the relationship between cipher suite lists and HTTP User-Agents. The User-Agent is a common client identifier in HTTP. However, in HTTPS, it is not directly 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 was limited by the set of clients accessing the monitoring server and we obtained a smaller number of pairs. The flow-based method used network monitoring and was 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 was 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 appeared in network traffic. On the other hand, only a 1-h sample of the HTTPS traffic contained almost all the cipher suite lists which were observed over the week-long measurement. Therefore, we used the dictionary obtained using 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 shared the same client identifier and differed only in their version or a similarly attainable value.

The fourth research question regarded application of SSL/TLS fingerprinting in network security. We discussed an example security incident which could be detected using network monitoring and SSL/TLS fingerprinting. A client, which tried to exploit a Shellshock vulnerability, exhibited a unique cipher suite list among other clients. Therefore, we could claim certain clients as suspicious and detect their activity in the network traffic to protect hosts in monitored network.

In conclusion, our work enhanced the capabilities of network forensics by introducing the network-based identification of HTTPS clients. Our network-based approach is lightweight, is 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.

**Competing interests**
The authors declare that they have no competing interests.

**Authors contributions**
MH contributed to the experiment design, performed the data collection, and has been involved in drafting the manuscript. MČ mainly contributed on dictionary creation and experiment evaluation. TJ participated in network data analysis and manuscript revision. PČ participated in the design of the detection method and has been involved in manuscript critical revision. All authors read and approved the final manuscript.

**Acknowledgements**
The dataset, containing the dictionary and aggregated measurement data, is publicly available at https://is.muni.cz/repo/1321931/dataset-https_client_identification.zip.

Received: 20 October 2015 Accepted: 11 January 2016

**References**

1. B Möller, T Duong, K Kotowicz, This POODLE bites: exploiting the SSL 3.0 fallback. PDF online. (2014). https://poodlebleed.com/ssl-poodle.pdf. Accessed 12 Jan 2015
2. P Velan, M Čermák, P Čeleda, M Dra r, A survey of methods for encrypted traffic classification and analysis. Int J Netw Manag. **25**(5), 355–374 (2015)
3. T Bujlow, V Carela-Español, J Solé-Pareta, P Barlet-Ros, Web tracking: mechanisms, implications, and defenses. CoRR. **abs/1507.07872** (2015). http://arxiv.org/abs/1507.07872
4. I Zeifman, Was that really a Google bot crawling my site? (2012). https://www.incapsula.com/blog/was-that-really-a-google-bot-crawling-my-site.html. Accessed 15 October 2015
5. E Raftopoulos, X Dimitropoulos, in *Security and Privacy Workshops (SPW), 2013 IEEE*. Understanding network forensics analysis in an operational environment (IEEE, New York, 2013), pp. 111–118
6. P Wang, S Sparks, CC Zou, An advanced hybrid peer-to-peer botnet. IEEE Trans Dependable Secure Comput. **7**(2), 113–127 (2010)
7. T Dierks, E Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2. IETF. Updated by RFCs 5746, 5878, 6176 (2008). http://www.ietf.org/rfc/rfc5246.txt
8. A Freier, P Karlton, P Kocher, The Secure Sockets Layer (SSL) Protocol Version 3.0. IETF (2011). http://www.ietf.org/rfc/rfc6101.txt
9. E Rescorla, HTTP Over TLS. IETF. Updated by RFCs 5785, 7230 (2000). http://www.ietf.org/rfc/rfc2818.txt
10. IANA – Internet Assigned Numbers Authority, protocol registries (2014). Web page. http://www.iana.org/protocols. Accessed 28 Jan 2015
11. P Yee, Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. IETF (2013). http://www.ietf.org/rfc/rfc6818.txt
12. C Meyer, 20 Years of SSL/TLS Research: An analysis of the Internet's security foundation. PhD thesis (2014). http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/MeyerChristopher/diss.pdf. Accessed 15 Jan 2015
13. O Levillain, A Ébalard, B Morin, H Debar, in *Proceedings of the 28th Annual Computer Security Applications Conference. ACSAC '12*. One year of SSL Internet measurement (ACM, New York, NY, USA, 2012), pp. 11–20
14. R Holz, L Braun, N Kammenhuber, G Carle, in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference. IMC '11*. The SSL

landscape: a thorough analysis of the x.509 PKI using active and passive measurements (ACM, New York, NY, USA, 2011), pp. 427–444
15. Z Durumeric, J Kasten, M Bailey, JA Halderman, in *Proceedings of the 2013 Conference on Internet Measurement Conference. IMC '13*. Analysis of the HTTPS Certificate Ecosystem (ACM, New York, NY, USA, 2013), pp. 291–304
16. Qualys SSL Lab, HTTP client fingerprinting using SSL handshake analysis (2014). Web page: https://www.ssllabs.com/projects/client-fingerprinting/. Accessed 23 Jan 2015
17. Ristić, Passive SSL client fingerprinting using handshake analysis (2014). https://github.com/ssllabs/sslhaf. Accessed 30 Jan 2015
18. JB Ullrich, Browser fingerprinting via SSL Client Hello messages (2014). https://isc.sans.edu/forums/diary/Browser+Fingerprinting+via+SSL+Client+Hello+Messages/17210. Accessed 15 Sept 2015
19. M Majkowski, SSL fingerprinting for p0f (2012). Web page: https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f. Accessed 28 Jan 2015
20. L Bernaille, R Teixeira, in *Passive and Active Network Measurement*, ed. by S Uhlig, K Papagiannaki, and O Bonaventure. Early recognition of encrypted applications. Lecture notes in computer science, vol. 4427 (Springer, Heidelberg, 2007), pp. 165–175
21. T Jirsík, P Čeleda, in *Advances in Communication Networking*, ed. by Y Kermarrec. Identifying operating system using flow-based traffic fingerprinting. Lecture notes in computer science, vol. 8846 (Springer, Cham, 2014), pp. 70–73
22. T Matsunaka, A Yamada, A Kubota, in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference On*. Passive OS fingerprinting by DNS traffic analysis (IEEE, New York, 2013), pp. 243–250
23. M Zalewski, p0f v3 (2014). http://lcamtuf.coredump.cx/p0f3/. Accessed 25 Jan 2015
24. T Kohno, A Broido, K Claffy, in *2005 IEEE Symposium On Security and Privacy*. Remote physical device fingerprinting (IEEE, New York, 2005), pp. 211–225
25. T Karagiannis, K Papagiannaki, N Taft, M Faloutsos, in *Passive and Active Network Measurement*, ed. by S Uhlig, K Papagiannaki, and O Bonaventure. Profiling the end host. Lecture notes in computer Science, vol. 4427 (Springer, Berlin, Heidelberg, 2007), pp. 186–196
26. HJ Abdelnur, R State, O Festor, in *Recent Advances in Intrusion Detection*, ed. by R Lippmann, E Kirda, and A Trachtenberg. Advanced network fingerprinting. Lecture notes in computer Science, vol. 5230 (Springer, Berlin, Heidelberg, 2008), pp. 372–389
27. T Unger, M Mulazzani, D Fruhwirt, M Huber, S Schrittwieser, E Weippl, in *Availability, Reliability and Security (ARES), 2013 Eighth International Conference On*. SHPF: Enhancing HTTP(S) session security with browser fingerprinting (IEEE, New York, 2013), pp. 255–261
28. M Mulazzani, P Reschl, M Huber, M Leithner, S Schrittwieser, E Weippl, in *Web 2.0 Workshop on Security and Privacy (W2SP)*. Fast and reliable browser identification with JavaScript engine fingerprinting, (2013). http://w2spconf.com/2013/papers/s2p1.pdf. Accessed 19 October 2015
29. P Eckersley, in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*. How unique is your web browser? PETS'10 (Springer, Berlin, Heidelberg, 2010), pp. 1–18
30. Proxy4Free.com, Free proxy servers—protect your online privacy with our proxy list (2015). Web page http://www.proxy4free.com/. Accessed 19 October 2015
31. R Dingledine, N Mathewson, P Syverson, in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. TOR: the second-generation onion router. SSYM'04 (USENIX Association, Berkeley, CA, USA, 2004), pp. 21–21
32. Y Gokcen, VA Foroushani, A Heywood, in *Security and Privacy Workshops (SPW), 2014 IEEE*. Can we identify NAT behavior by analyzing Traffic flows? (IEEE, New York, 2014), pp. 132–139
33. V Krmíček, J Vykopal, Krejčí, in *Proceedings of the 5th International Student Workshop on Emerging Networking Experiments and Technologies*. Netflow based system for NAT detection. Co-Next Student Workshop '09 (ACM, New York, NY, USA, 2009), pp. 23–24
34. R Hofstede, P Čeleda, B Trammell, I Drago, R Sadre, A Sperotto, A Pras, Flow monitoring explained: from packet capture to data analysis with NetFlow and IPFIX. IEEE Commun Surv Tutorials. **16**(4), 2037–2064 (2014)
35. P Velan, T Jirsík, P Čeleda, in *Advances in Communication Networking*, ed. by T Bauschert. Design and evaluation of HTTP protocol parsers for IPFIX measurement. vol. 8115 (Springer, Heidelberg, 2013), pp. 136–147
36. cURL Contributors, cURL - command line tool and library for transferring data with URL syntax (2015). http://curl.haxx.se/. Accessed 25 Jan 2015
37. O Alders, HTTP::BrowserDetect (2015). https://github.com/oalders/http-browserdetect. Accessed 15 Sept 2015
38. StatCounter, StatCounter Global Stats (2015). http://gs.statcounter.com/#all-browser-ww-monthly-201506-201506-map. Accessed 01 October 2015
39. A Cortesi, mitmproxy (2014). https://mitmproxy.org/. Accessed 15 Sept 2015