Network Forensics of SSL/TLS Encrypted Channels

Meng-Da Wu and Stephen Wolthusen
Information Security Group, Royal Holloway, University of London, UK
M.D.Wu@rhul.ac.uk
Stephen.Wolthusen@rhul.ac.uk

Abstract: Network forensics is increasingly hampered by the ubiquitous use of encrypted channels by legitimate and illegitimate network traffic. Both types of traffic are frequently tunneled over application-layer encryption mechanisms, generally using the ubiquitous TLS (SSL) protocol. This results in traditional network forensics tools being largely limited to recording external characteristics (source and origin addresses and ports, time and traffic patterns), but with little insight into content and purpose of the traffic. We propose that a precise characterization of encrypted traffic not only in the form of the external characteristics but also through the analysis of the exact mechanisms, variants and options used for the encrypted channel but visible without access to key material along with a fine-grained analysis of the traffic patterns itself incorporating domain knowledge of the SSL/TLS protocol can yield valuable insights and help to classify traffic into legitimate traffic, illegitimate immediate traffic (e.g. as caused by a Trojan). It can also characterize traffic that is added to an existing data stream by an illegitimate source. In this paper, we therefore present and characterize different traffic types and subsequently analyze this traffic, including the SSL/TLS protocol data units using selected sequence mining techniques.

Key words: SSL/TLS, network forensics, traffic classification, sequence alignment

1. Introduction

The identification and classification of encrypted channels presents a significant challenge for intrusion detection (ID) and forensics, which is exacerbated by the fact that many of these channels are established at the application level and therefore cannot be readily controlled by system administrators. Traditional ID and forensics tools can record only external characteristics (source and origin addresses and ports, time and traffic patterns), but have only limited insight into the legitimacy and purpose of the observed data streams.

Observing that different applications as well as permutations of client and server systems exhibit differing characteristics in the way that they use encryption protocols, we propose that it is possible to obtain information on the legitimacy of such traffic by subjecting the variants of the protocols used to classification and analysis.

In this paper we restrict ourselves to the SSL/TLS protocol suite as this represents an ubiquitous standard for application-level encrypted channels used in a variety of legitimate applications as well as malicious traffic. It should be noted, however, that the techniques discussed in this paper do, however, also apply to other protocols such as the SSH protocol family or protocols such as IPSec. This analysis can be effected without requiring access to key material, and can be used for identifying known (legitimate) encryption engines through fingerprinting as well as for classifying unknown, potentially suspicious encryption engines. Moreover, we also investigate the use of statistical patterns in the handling of encrypted payload streams, which can also be used to classify application behavior.

The remainder of the paper is therefore structured as follows: Section 2 reviews the background of protocols and techniques used along with related work. We provide an overview of techniques and metrics for classification and fingerprinting of encrypted streams in section 3 and apply these in section 4. Section 5 then reviews the results before we discuss ongoing and future research in section 6.

2. Background and related work

2.1 Traffic classification and inference

Traffic classification has recently been the main subject of network security research. HTTP, SMAP, FTP, SSH, Telnet, and SSL can be distinguished by extracting information from packet

payloads (Zhang & Paxson 2000; Moore & Papagiannaki 2005; Bernaille, et al. 2006), inter-arrival time (Early, et al. 2003; Zhang & Paxson 2000), TCP flaps (Early et al. 2003; Moore & Zuev 2005; Karagiannis, et al. 2005), and packet size (Wright, et al. 2006; McGregor, et al.2004). A number of techniques, including machine learning, statistical analysis, or customized algorithms have been applied to traffic classifications. SSL tunnels can be readily recognized using these techniques as the protocol is well-defined. Moreover, packet timing and size information can also reveal sensitive information; e.g. browsing SSL/TLS encrypted web pages can be identified by statistical traffic analysis for web pages identification (Sun, et al. 2002). Inter-keystroke timing can be inferred from IP packet to reduce search space for cracking passwords (Song, et al. 2001). More exotically, remote physical device fingerprinting can be accomplished by obtaining clock skew information from TCP headers of a device (Kohno, et al. 2005). Moreover, for the purpose of information inference, sequence alignment is also applied to this domain. For instance, web navigation patterns can be clustered by sequence alignment methods (Hay, et al. 2001). Semi-global and pair-wise alignment techniques have also recently been applied in intrusion detection for detecting masquerades in Unix commands (Coull, et al. 2003).

All of the above techniques can be applied to network forensics. For the purpose of finding more clues about SSL tunnels, each of these can play different roles. Traffic classification techniques can provide high level inference for classifying application layer, and packet timing and size information can be used in several application-specific information leakage attacks on various kinds of encrypted traffic. Characteristic actions such as web page access can also be identified. Therefore, SSL tunnel identification and fine-grained operations of SSL software can be inferred by these techniques. Implementation signature fingerprints and statistical analysis can add another dimension to this classification.

2.2 SSL/TLS sequential headers

SSL (Secure Sockets Layer) / TLS (Transport Layer Security) sits above TCP and provides a number of security services, including traffic encryption, client-side and server-side authentication, and message integrity. It was originally developed to secure connections between web servers and browsers, but is not limited to tunneling HTTP (Hypertext Transfer Protocol). Any upper-layer protocol or application which replies on TCP can integrate security services provided by SSL/TLS, such as FTP, Telnet, or POP. SSL/TLS is a multilayer protocol consisting of four separate components, including record protocol, handshake protocol, change cipher spec protocol, and alert protocol. SSL/TLS is flexible in its configuration and choice of parameters, and it can be implemented by different methods, such as server-side only authentication, both-side authentication, or (historically) using the FORTEZZA handshake. Implementations can be customized to suit application requirements (Rescorla 2000).

All key material exchange, application data delivering, and session termination are performed by exchange of SSL/TLS headers (segments), such as Client Hello, Certificate, and ChangeCipherSpec. These headers transfer different messages to establish SSL/TLS tunnels, and headers appear in different order depending on the concrete implementation as the specification allows for some variation. Moreover, some pure TCP headers without carrying any SSL/TLS messages appear in real network traffic as artifacts, permitting the use of sequential pattern identification.

2.3 Sequence alignment mechanisms

It can be observed that SSL/TLS segments form a sequential pattern whose classification and comparison represents a similar problem as the sequence alignment problem found in bioinformatics comparing RNA, DNA, or protein sequences. A sequence is usually defined as a number of elements, objects or events arranged in succession. The sequence alignment technique is a well-developed tool to measure similarity between sequences. Computational approaches to sequence alignment generally fall into three categories: global alignments, local alignments, and semi-global alignments. Various alignments allow the algorithm to search subsequence with different features. The global alignment, known as Needleman-Wunsch algorithm (Needleman &Wunsch 1970), forces the alignment to span over the entire length of both strings. Local

alignment, known as Smith-Waterman algorithm (Smith & Waterman 1981), focuses on identifying best aligned subsequences of two sequences over all possible substrings. Both prefixes and suffixes are disregarded by achieving local alignment to find best matching substrings. Finally, semi-global alignment aims to align the whole sequence based on several similar regions.

Global alignments attempt to fully align two sequences. In this case, short and highly similar substrings might be ignored because they are outweighed by the rest of the sequence. In our experiments, we assume SSL/TLS software would produce regular sequential headers which can be treated as similar pieces of subsequences. We chose local alignment to compute subsequence similarity of each tunnel. Sequence alignment is a technique used in bioinformatics which can be leveraged easily for the purposes of our discussion. The focus of this paper is primarily on the use of local alignment techniques. For this purpose we have used the sequence alignment and visualization software *Geneious*, an integrated bioinformatics tool suite for manipulating, finding, sharing, and exploring biological data which is presented by sequential symbols. In our experiment, we used this tool to compare sequential SSL segments after classification into symbols.

3. Analytical techniques

3.1 Data pre-processing

Prior to the analysis of SSL/TLS protocol data units (typically protocol headers), these need to be captured and decoded. Both functions are readily available in the form of a popular open source tool, *Wireshark*[1]. Data was obtained on a private switched network using the *tcpdump* and *WinPCap* packet capture mechanisms, resulting in 100% capture rates. Wireshark was then used to decode the SSL/TLS segments, which were then classified into different segment types and transformed into individual identifiers for further processing and analysis.

Since the properties of interest for analysis are evident only in individual TCP conversations over which the SSL/TLS protocol is executed, it is also necessary to isolate these sessions. This requires not only the separation into source and recipient IP addresses and ports, but particularly for busy systems must also take sequence numbers into account. Since encrypted sessions are typically exhibiting low traffic volumes and are clearly separated by at least the initial and usually closing handshakes, this is trivial.

Traffic intended for further analysis is filtered and exported for further use. Given that we are interested primarily in discretely classified protocol data units, which can be encoded in a finite set of symbols, this means that statistical analysis tools for sequence and pattern matching represent viable approaches. We have therefore investigated the use of tools more commonly used in bioinformatics, namely the machine learning tool *Bioweka*[2], which we use for data transformation and subsequent analysis. In addition, another tool from the application domain of bioinformatics can be used effectively, namely the *Geneious*[3] tool, which implements a number of local and global sequence alignment algorithms and also provides visualization mechanisms particularly useful in exploration.

3.2 Data analysis

This paper is concerned with both the fingerprinting of TLS/SSL session initiation and also the statistical properties of these sessions once they have started properly.

3.2.1 SSL/TLS overview

SSL/TLS is an application-layer protocol which operates solely over reliable connections, namely the TCP transport layer. It establishes a tunnel over which application data can be exchanged in accordance with the security policies of the parties participating in the session. To this end, the

[1] http://www.wireshark.org
[2] http://www.bioweka.org
[3] http://www.geneious.com

parties must negotiate security features such as authentication mechanisms, ciphers, and integrity protection mechanisms before the actual exchange of payload data can begin. This leads to the following identifiable features of a SSL/TLS session establishment as observed on the network at an abstract level:

1. A three-way TCP handshake is established between the parties. There exists a limited amount of variability in this process, which can be used for TCP session fingerprinting.
2. A SSL/TLS handshake is then established. Both the standard as well as standard interpretations and implementation defects allow for a significant element of variability between implementations, which will also vary depending on which combination of initiator (client) and server is performing the session establishment handshake.
3. The tunnel start to exchange application data between both client and server party, potentially interrupted by cipher suite renegotiations.
4. If either communication party intends to close a session, this can (but need not) be performed by initiating a SSL/TLS session teardown handshake followed by a TCP session teardown. Several other alternatives such as session resets and termination without explicit teardown are also encountered.

One important property observed in network traffic is that in addition to the establishment and teardown elements, TCP segments which do not contain SSL/TLS messages will also be observable. The occurrence of such TCP segments is not prescribed by the SSL/TLS standards and can therefore be a valuable marker for sequential pattern identification of specific implementations.

3.2.2 Fingerprinting of SSL/TLS sessions

In analogy to TCP fingerprinting, SSL/TLS is also amenable to similar approaches. Moreover, since there exist several versions and a large number of degrees of freedom in implementing this family of protocols, the feature space which can be exploited is significantly larger than that of TCP. The session establishment phase for SSL/TLS tunnels produces an unique sequential pattern of protocol data units which is variable in length and can, including the TCP handshake segments, encompass 30 to 50 messages. The precise pattern depends on a number of factors, including the releases of SSL/TLS engines used and their configuration, e.g. for backwards compatibility, protocol versions and cipher suites to be used, and the type of authentication mechanism deemed acceptable or required. It should be noted that on congested channels, noise may be introduced in this pattern e.g. through sequencing errors or retransmissions. The experiments reported in the following section did not include such noise additions. Similarly, though as discussed in the previous section not necessarily always present, the session teardown for an established SSL/TLS tunnel yields another pattern of protocol data units, typically between 5 and 7 segments long.

Moreover, these fingerprint patterns are independent of the actual data transmitted over the tunnel after establishment. There is, however, a way to influence the patterns through the configuration of the tunnel parameters, so that it is possible not only to fingerprint the combinations of SSL/TLS engines used on both sides of a session but also to indirectly include the client and server applications using this channel based on their configuration behavior. The precise behavior in a given combination of client and server system also depends on the degrees of freedom and configurability in both systems. If one of the parties is not prepared to negotiate tunnel properties, this of course limits the overall feature sub-space which can be used for this combination.

3.2.3 Patterns of TCP and SSL/TLS segments

Once the session establishment handshakes are completed, the tunnel is then ready to transmit application payload data. Except for optional re-keying and cipher suit change messages used by the SSL/TLS protocols, this means that segments containing (encrypted) payload data will be sent over the tunnel.

As noted above, it can be observed that many implementations (potentially also owing to interactions between the SSL/TLS layer and the underlying network protocol stack) will not make

optimum use of the TCP protocol by always adding administrative information such as segment acknowledgments to existing payload messages. Instead, TCP segments without SSL/TLS payloads can be observed. This allows the identification of statistical patterns in the sequencing and frequency of the SSL/TLS application protocol data units and the payload-free TCP segments during ongoing sessions, e.g. in the form of an analysis of the proportions of these messages over selected time and message frequency windows, which exhibit distinguishing patterns for given tunnel endpoint combinations.

For ease of reference, we therefore refer to “SSL/TLS application” and “TCP” segments when discussing the experimental and analytical results in the following section.

3.2.4 Sub-sequence pattern matching using payload data exchange

While, as discussed in the preceding section, “SSL/TLS application” and “TCP” segments form a characteristic pattern, there are several factors influencing the segment patterns observed. In part, these are determined by the combination of underlying network protocol stack, SSL/TCP layer, and the application’s behavior (e.g. with a TLS-based VPN transporting character-by-character terminal data using the TCP PSH flag differing significantly from a bulk data transfer using HTTP, even if retaining all other variables constant).

It is therefore insufficient to observe only global patterns as these may not exhibit sufficient uniformity for analysis. However, by concentrating on characteristic patterns occurring in smaller excerpts of the analyzed data stream and applying a *local alignment* string-matching technique also commonly used in bioinformatics, similarities can still be identified.

3.2.5 Other features supporting inference

Three additional features can be used to distinguish particular SSL/TLS implementations and their configuration (i.e. some implementations may support these features but have them deactivated or misconfigured. One of these features is the session caching mechanism, which allows endpoints to resume a session without requiring the computational cost of negotiating a new cipher suite. If no previous session exists, or if one of the parties wishes to establish a new session, it will set the session identified to 0, forcing the negotiation of a new cipher suite and generation of a new master key in the process. If a previous session exists, and both parties are willing to resume this session, the session identifier of that previous session is exchanged, resulting in re-use of both cipher suite and master key.

Another behavioral pattern is that some implementations force the sequential use of sessions within the established SSL/TLS tunnel, while others make use of the full SSL/TLS feature set and encapsulate multiple simultaneous payload sessions within the context of the same tunnel. This can be observed even without access to key material from the participating parties by extracting session headers and analyzing whether session segments overlap. The final feature is linking number of communication behavior. This can be implemented by either single tunnel of 2 TCP ports or multiple tunnels established to accomplish SSL/TLS communication.

4. Experiments

4.1 Experimental setup

Six software packages were selected as specimens for the feature analysis of message segment sequences. Table 1 shows the software specimens chosen for use in the experiments. To facilitate the evaluation of statistical properties, a training and an evaluation data set were captured for each specimen against an Apache 1.3.34 with built-in SSL/TLS module[4]. All features were extracted and analyzed for the accuracy of predictions obtainable for each combination and feature.

[4]http://slampp.abangadek.com/wiki/HomePage.

For each data set of each software, we collected 30 sample sets for analysis. Each sample set included the full communication traffic between client and the server. Each sample set in turn consists of a number of sessions which are established between 2 TCP ports, and we randomly chose one session from a sample set for extracting features. This results in the selection of 30 sessions for both the training and test sets, respectively.

Three different SSL/TLS tunnel tools were used to analyze tunneled traffic. SoftEther[5] can establish a tunnel and present it as a proxy to client systems, e.g. for accessing services on the Internet. One of the applications of SoftEther is to tunnel through enterprise firewalls, which it can accomplish by using a connection to port 443 to link to a server outside the enterprise, and to forward arbitrary traffic from that endpoint. In our experiment, traffic in which a client used the SoftEther server as such a proxy for browsing WWW sites was captured.

We used Titan FTP Server[6] and Gene6 FTP Server[7] to present SSL/TLS FTP communication implemented over SSL/TLS and chose WebDrive[8] as a SSL-capable FTP client to produce our experimental data. For the SSL/TLS FTP server, explicit or implicit methods can be used to establish a SSL/TLS tunnel with an SSL/TLS FTP client. In our experiments, we set both client and server side to implicit mode, and captured traffic as the client arbitrarily accessed, deleted, or modified files through SSL/TLS tunnels. We captured and extracted all features from both the command and data channels used by the FTP protocol. Three web scanner tools were tested in our experiments, including Nikto [9], N-Stalker[10], and WatchFire[11]. We captured traffic representing the scanner tool analyzing the test web site. For regular client-side HTTPS traffic, we used Firefox version 1.5.0.10 as our experimental browser client and captured data while a user browsed WWW sites.

In this paper, we collected all raw data as the client party communicated with the server party over an undisturbed, isolated network. Thus, we assume that we collect 100% of network traffic without sequencing errors or missing packets; the incorporation of such noise into the data set is the subject of future research.

Table 1: Software specimens using SSL/TLS

Software Category	Client Side(software version,OS)	Server Side(software version,OS)
SSL/TLS VPN	SoftEther (1.00 on XP SP2)	SoftEther (1.00 on XP SP2)
SSL/TLS FTP	WebDrive (7.21 on XP SP2)	GeneFTP (3.9.0 on XP SP2)
SSL/TLS FTP	WebDrive (7.21 on XP SP2)	TitanFTP (5.34 on XP SP2)
Web scanner	Nikto (1.34 on Linux 2.4.31)	Web Server (Apache 1.3.34 on Linux 2.6.13.2)
Web scanner	N-Stalker (6.0.1.120 on XP SP2)	Web Server (Apache 1.3.34 on Linux 2.6.13.2)
Web scanner	Watchfire (7.0 on XP SP2)	Web Server (Apache 1.3.34 on Linux 2.6.13.2)
Browser	FireFox (1.5.0.10 on XP SP2)	Web Server (Apache 1.3.34 on Linux 2.6.13.2)

4.2 Extraction of proportion-based statistical patterns

For extracting proportion-based statistical patterns, we selected 200 segments from the 31st to 230th packets of 30 sessions each from the data sets to compute this pattern. There are only 2 kinds of segment formats appearing during this period, including “SSL/TLS” application and “TCP” segments. The proportion of the statistical distribution is a pattern specific to a given software package. As the length of the fingerprint-related segments varies and does not contribute to this pattern, we have omitted the first 30 segments.

[5] http://www.softether.com
[6] http://www.southrivertech.com.
[7] http://www.g6ftpserver.com/
[8] http://www.southrivertech.com/products/webdrive/index.html
[9] http://www.cirt.net/code/nikto.shtml.
[10] http://www.nstalker.com.
[11] http://www.watchfire.com.

Table 2 shows our experimental data, and the percentage presents the proportion of "SSL/TLS" application to "TCP" segments in a specific tunnel. We used the P value of hypothesis tests to evaluate accuracy. Here, we only evaluated the mean value by hypothesis testing in which we used two-tailed tests without assumptions about the population standard deviation.

Table 2: Proportion-based statistical patterns

Traffic Category	μ Training Set	μ Testing Set	σ Training Set	σ Testing Set	P Value
SoftEther	57.71%	57.68%	0.031806	0.041032	0.9716
Https	37.08%	37.48%	0.018712	0.017542	0.2117
Gene6FTP command	92.82%	92.27%	0.023359	0.028579	0.2977
Gene6FTP data	11.00%	11.10%	0.008375	0.005632	0.3308
TitanFTP command	92.33%	91.52%	0.035703	0.018959	0.0188
TitanFTP data	11.63%	11.32%	0.028252	0.021192	0.4180

4.3 Extraction of sub-sequence matching pattern

For extracting sub-sequence matching pattern, we selected 200 segments from the 31st to 230th packets of 30 sessions of each data set to compute this pattern. We used local alignment to randomly compute every two sequences until we get 30 value of similarity. We chose the Smith-Waterman algorithm implemented in Geneious as our local alignment technique. This allows the identification of patterns using default local alignment to compute similarities. Table 3 shows our experimental data, and the similarity levels of matching sub-sequences for given specific tunnels. P values were used to evaluate accuracy. As before, only the mean value was evaluated using two-tailed tests without assumptions about the population standard deviation.

Table 3:Sub-sequence matching pattern

Traffic Category	μ Training Set	μ Testing Set	σ Training Set	σ Testing Set	P Value
SoftEther	71.17%	70.27%	0.038801	0.02983	0.1035
Https	85.54%	85.60%	0.029494	0.028333	0.3986
Gene6FTP command	94.79%	93.56%	0.028492	0.029272	0.3494
Gene6FTP data	97.60%	96.44%	0.017927	0.04175	0.1292
TitanFTP command	92.73%	93.04%	0.027427	0.023033	0.4562
TitanFTP data	98.50%	98.43%	0.008808	0.010148	0.7190

4.4 Extraction of fingerprint patterns and other features

4.4.1 Fingerprint

Domain knowledge was applied to extract abstract segment information for identifying session fingerprints. In our experiments for this paper, we classified segments into categories based on the abstract protocol data units of the SSL/TLS protocol. These classifications are given a symbolic shorthand (also used in pattern matching) as shown in table 4. If a message segment contains one protocol data unit (PDU), this results in a single classification unit (letter), whereas a segment containing multiple PDUs will be classified as a sequence of units.

Table 4: Rules of extracting abstract segment information

Letter	Protocol	Abstract Information
S	SSL/TLS	Client Hello
H	SSL/TLS	Server Hello
G	SSL/TLS	Change Cipher Spec
M	SSL/TLS	Encrypted Handshake Message
I	SSL/TLS	Certificate
D	SSL/TLS	Server Hello Done
K	SSL/TLS	Client Key Exchange
X	SSL/TLS	Server Key Exchange

A	SSL/TLS	Application Data
E	SSL/TLS	Encrypted Alert
N	TCP	TCP Flow
F	TCP	TCP FIN

In our experiments, we randomly chose 30 sessions for each tunnel. In every session, selected 35 segments from the first 30 and final 5 segments, and then classified segment information into sets of letters. We choose the mode from 30 sets of letters as the fingerprint. Table 5 shows the raw fingerprint data sets from all tunnels. As before, a noiseless capture is assumed, making this dependent solely on software stacks and combinations of client/server systems.

Table 5: Raw fingerprint information

Software Tunnels	Fingerprint
SoftEther	NNNS[HID][KGM][GM]NAAAAAAAANAAA– NAAAANANANEFNFN
Https	NNNSN[HIXD]N[KGM][GM]ANANAAAAAAN– NNNNNNNNNNFENFN
Gene6FTP Command	NNNS[HID][KGM][GM]ANAAAAAAAAAAA– AAAAAAAAAANFNFN
Gene6FTP Data	NNNNS[HID][KGM][GM]ANNNNNNNNNN– NNANNNNNNNNNNFN
TitanFTP Command	NNNS[HIXD][KGM][GM]ANAAAAAAAAAAA– AAAAAAAAAANFNFN
TitanFTP Data	NNNS[HIXD][KGM][GM]ANNNNNNNNNNNN– NNNNANNNNNNFNFN
Nikto	NNNSN[HIXD]N[KGM][GM]AFAEFN (packet size :15)
N-Stalker	NNNSN[HIXD][KGM][GM]AAENFNFN (packet size :16)
Watchfire	NNNSN[HGM][GM]AAFNFN (packet size :13)

4.4.2 Other features of interest

There are several features of SSL/TLS channels which are not present in all implementations and may not manifest in each permutation and configuration which can also be of interest for fingerprinting and classification of encrypted channels. Table 6 presents features resulting from different SSL/TLS tunnel implementations, including variations in the resumption strategies for SSL/TLS handshakes, SSL/TLS segments crossing, and linking number.

Table 6: List of other features of interest

Software Tunnels	Resuming Handshake	Segments Crossing	Linking number
SoftEther	No	No	Single
Https	Yes	Yes	multiple
Gene6FTP Command	No	Yes	multiple
Gene6FTP Data	No	Yes	multiple
TitanFTP Command	No	Yes	multiple
TitanFTP Data	No	Yes	multiple
Nikto	No	No	multiple
N-Stalker	No	Yes	multiple
Watchfire	Yes	No	multiple

4.5 Features discussion

Features of fingerprint, resuming handshake, segments crossing and linking numbers demonstrate 100% accuracy from our (noiseless) training data sets and testing data sets, indicating the expected large feature size allowing precise classification of tools even for a relatively coarse level of classification. Web scanner tools do not communicate with web servers as they interrupt communication sessions before transmitting payload data, so proportion-based or sub-sequence matching statistical patterns cannot be extracted from them. Moreover, the length of fingerprints in case of scanners is limited. As noted above, however, measurement data is simulated and occurred only over a lightly loaded local area network; both the use of different server system times

and software packages including different versions as well as their behavior under noisy conditions are subjects of future research.

5. Experimental results and discussion

The promising results of section 4 indicate that significant amounts of information useful for the forensic analysis of encrypted traffic without knowledge of keying material, albeit with significant variations in the accuracy with which individual applications (as opposed to SSL/TLS implementations over which these applications communicate) can be identified. However, The technique of extracting fingerprinting and behavioral characteristics can be used to identify several elements and permutations of the software stacks in use. Moreover, information and privacy leakage of software's operations can be identified by observing protocol implementations, and this idea can be applied to all encrypted network protocols for forensic purposes. In accordance to our observations, more detail software operations can be identified by analyzing protocol implementation behaviors. While the specific operational individual implementations were not subject of investigations reported in this paper, the techniques we describe are readily applicable to obtaining finer-grained classification results from encrypted protocols. This information leakage does not originate with the protocol, which can be considered secure. Rather, inferences rely on how software implements a secure protocol, and legitimate variations in implementation behavior that are still within the confines of the specification. Thus, information leakage from protocol implementing is inevitable and needs to be checked (and potentially masked) if privacy or evasion of intrusion detection mechanisms is a concern.

The fingerprints and behavioral patterns observed are functions of the endpoints involved in a session since negotiation behavior figures prominently in the SSL/TLS protocol. These can e.g. be formed by a SSL-capable FTP client communicating with a FTP server (directly of via proxy), communication between WWW client and server systems, or SSL VPN configurations. Fingerprinting techniques can be applied in a straightforward manner and are easily feasible in practical network forensics applications. For other traffic, statistical analysis yields more ambiguous results, so it may well be beneficial to concentrate solely on fingerprints as e.g. browsing behavior or interactions between web services and clients are more complex and would require more detailed statistical analysis incorporating more domain knowledge to gain results with adequate statistical significance. As noted above, patterns observed for SSL/TLS tunnels also depend on configuration of the software stacks in use, which introduces several additional degrees of freedom for profiling (e.g. there exist two methods to do SSL handshake in SSL FTP software, implicit and explicit methods). This can improve the accuracy with which an application/malware can be identified as this fingerprint pattern becomes more specific.

Finally, it should be noted that we have merely presented an arbitrary selection of SSL/TLS implementations and applications using this approach; actual forensic work will require the creation and maintenance of a detailed database of fingerprint patterns for the various parameters such as host operating systems, SSL/TLS implementations, software (for positive and negative identification e.g. of malware), and configuration issues. We have, however, demonstrated that these patterns exist and can be readily extracted and classified.

6. Conclusions

In this paper we have presented an extension of the successful context of TCP session fingerprinting to the domain of SSL/TLS tunnels, which can provide important forensic information on the precise implementations and application programs using these SSL/TLS implementations. Such information can be particularly useful when trying to identify whether the encrypted and hence opaque data stream is emanating from a known and legitimate application program on a given host or whether the sessions originate with a potentially hostile unknown program such as e.g. a rootkit. We have also shown that the permutations of SSL/TLS implementations, network stacks, application configuration, and application data flow can be used to further provide evidence for characterizing the applications and endpoints of encrypted tunnels, also without requiring access to key material, and from observations solely at the network layer. Particularly for the latter we have investigated the use of sequence alignment and matching tools from the application domain of

bioinformatics, which, like network forensics, must also cope with noisy, incomplete data sets yet still be capable of adequate performance in sequence matching. Further tuning and adaptation of these pattern matching techniques, particularly in congested high-traffic environments is a subject for further research, as is further expansion of the database of configuration permutations since the results reported here represent only proof of principle. We also intend to investigate the behavior of real-time channels such as encrypted chat or voice over IP connections and their behavior based on similar fingerprinting and statistical analysis of other encryption and tunneling protocols such as IPSec.

References

Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A. and Salamatian, K. (2006). 'Traffic Classification on the Fly'. *SIGCOMM Comput. Commun. Rev.* (2):23–26.

Coull, C., Branch, J., Szymanski, B. and Breimer, E. (2003). ' Intrusion Detection: A Bioinformatics Approach '. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, pp. 24–33. IEEE Computer Society.

Early, J.P., Brodley, C.E. and Rosenberg, C. (2003). ' Behavioral Authentication of Server Flows'. In *Computer Security Applications Conference, 2003. Proceedings. 19th Annual)*, pp. 46–55. IEEE Computer Socirty Press.

Hay, B., Wets, G. and Vanhoof, K. (2001). ' Clustering Navigation Patterns on A Website Using A Sequence Alignment Method '. In *Intelligent Techniques for Web Personalization : 17th* , Artificial Intelligence .

Karagiannis, T., Papagiannaki, K. and Faloutsos, M. (2005). 'BLINC: Multilevel Traffic Classification in the Dark'. In *SIGCOMM '05: Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, vol. 35, pp. 229–240, New York, NY, USA. ACM Press.

Kohno, T., Broido, A. and Claffy, K.C. (2005). 'Remote Physical Device Fingerprinting'. *IEEE Transactions on Dependable and Secure Computing* :93–108.

McGregor, A., Hall, M., Lorier, P. and Brunskill, J. (2004). 'Flow Clustering Using Machine Learning Techniques'. In *Passive and Active Network Measurement*, Lecture Notes in Computer Science, pp. 205–214. Springer Berlin .

Moore, A. W. and Papagiannaki, K. (2005). 'Toward the Accurate Identification of Network Applications'. In *Proceedings of the Passive & Active Measurement Workshop (PAM2005)*, vol. 3431 of *Lecture Notes in Computer Science*, pp. 41–54, Berlin, Germany. Springer-Verlag.

Moore, A. W. and Zuev, D. (2005). 'Internet Traffic Classification Using Bayesian Analysis Techniques'. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pp. 50–60, New York, NY, USA. ACM Press.

Needleman, S. B. and Wunsch, C. D. (1970). 'A General Method Applicable to the Search for Similarities in the Amno Acid Sequences of Two Proteins'. *Journal of Molecular Biology* (3):443–453.

Rescorla, E. (2000). *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley, Reading, MA, USA.

Smith, T. F. and Waterman, M. S. (1981). ' Identification of common molecular subsequences '. *Journal of Molecular Biology* pp. 195–197.

Song, D. X., Wagner, D. and Tian, X. (2001). ' Timing Analysis of Keystrokes and Timing Attacks on SSH'. In *In Proceedings of the 10th USENIX Security Symposium*, Washington, D.C., USA. The USENIX Association.

Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, P.N. and Qiu, L. (2002). ' Statistical Identification of Encrypted Web Browsing Traffic'. In *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pp. 19–30. IEEE Computer Socirty.

Wright, C. V., Monrose, F. and Masson, G.M. (2006). ' On Inferring Application Protocol Behaviors in Encrypted Network '. In *The Journal of Machine Learning Research* , pp. 2745–2769. Microtome Publishing.

Zhang, Y. and Paxson, V. (2000). 'Detecting Backdoors'. In *Proceedings of the 9th USENIX Security Symposium*, pp. 157–170, Denver, CO, USA. USENIX.