Classification of TLS Applications

Chris Richter, Michael Finsterbusch, Klaus Hänßgen
Faculty of Computer Science,
HTWK Leipzig, Germany
{richter|finster|haenssge}@imn.htwk-leipzig.de

Jean-Alexander Müller
Dept. of Communication and Computer Science,
Hochschule für Telekommunikation Leipzig, Germany
Jean-Alexander.Mueller@ieee.org

***Abstract*—Traffic monitoring, traffic engineering, quality of service applications, network intrusion detection systems, as well as network management systems require the basic knowledge of which traffic is transmitted over a network. The increasing number of applications which are using encryption techniques such as TLS lower the ability to determine the applications that are running within a network. In this paper, we propose a method to detect applications in TLS encrypted connections. Our method uses a hybrid approach which combines protocol decoding to identify TLS traffic and to gather reliable information about the application data. Furthermore, a machine learning algorithm is used to determine the application which is protected by TLS. We describe our approach and compare it with other related methods in theory and prove its advantages on network measurements. The results show a significant improvement on classification Recall and Precision.**

Keywords–application classification, TLS, Internet traffic, machine learning.

I. Introduction

An increasing number of network protocols and applications encrypt the payload to protect privacy and integrity of the data. One popular way of doing this is to use the Transport Layer Security (TLS) protocol [1], which is a further stage of the Secure Socket Layer (SSL) protocol standardised by the Internet Engineering Task Force (IETF). Thus, the acronyms SSL and TLS are are often used as a synonym. An Internet study [2] from 2013 revealed that 356 applications within enterprises networks used SSL in some way, while 85 did not use standard SSL ports.

In order to do their work properly network management systems and security related systems such as firewalls or Network Intrusion Detection Systems (NIDS) need to know the kind of application. Therefore, these systems have to know whether the traffic is encrypted and which kind of application is being transmitted. To solve this problem, our approach is to use a hybrid method. First, we identify the TLS traffic. Second, the TLS data is analysed to determine the application. Due to encryption, only statistical information can be used for the second step.

The remainder of this paper is structured as follows: in Section II the related work is outlined. This is followed by Section III, which describes our approach for better TLS application classification, and Section IV which demonstrates the benefit of this approach on measurement results, Section V concludes the paper.

II. Related Work

Most research on TLS application classification has been done merely with statistical analysis. In most cases different kinds of well known machine learning algorithms were used. In some papers the authors concentrate on a single statistical parameter and use a dedicated method to evaluate the results.

There are two kinds of detecting applications for TLS connections. The first is to detect whether the network traffic is TLS or not. The second is to classify different applications, e. g., Hypertext Transport Protocol Secure (HTTPS), Simple Mail Transport Protocol Secure (SMTPS), etc., which are using TLS encryption. The goal in [3] is to distinguish TLS from non-TLS traffic. The authors are using the machine learning algorithms AdaBoost, C4.5, RIPPER and NaiveBayes and the statistical parameters packet length, inter-arrival time, duration and packet count. The detection rate varies between 70% and 98% for the different algorithms and different data sets.

The most work is related to the second approach which tries to classify different applications on top of TLS. In [4] the machine learning algorithm Random Forest as well as the clustering algorithm K-Means were used to classify network traffic for an intrusion detection system. It was shown that the approach is feasible for network monitoring, but the authors do not give further information about the classification rates. The authors of [5] used only the statistical parameter packet size for application classification. Therefore, the packet size of a packet is ranged to one of 30 bytes bins. The packet size distribution for a packet flow is compared with the Chi-square test to the values of known applications. This approach has a low classification accuracy of 10% to 40% for most observed applications.

Two statistical parameters – inter-arrival time and packet length – were used in [6] in conjunction with one of the three clustering algorithms DBSCAN, K-means and EM. On a data set with the File Transport Protocol (FTP), Real-time Protocol (RTP) and the Remote Framebuffer protocol (RFB), they could reach an accuracy up to 99%. The same parameters were also used by [7], but they used feature vectors containing several sub-parameters of inter-arrival time and packet size such as minimum, maximum, mean value and standard deviation. To compare the vectors of the ongoing packets with the known data set, the Euclidean distance or the Hamming distance are used. With this approach the authors could classify 80% to 94% of the used network traffic.

[8] is a PhD thesis about the identification of applications in encrypted tunnels, with the focus rests on HTTPS tunnels. The packet size of network packets is ranged to one of 15

 ISBN: 978-1-61208-362-9

bins. Several machine learning algorithms (Naive Bayes, C4.5, Decision Tree, neural networks, Nearest Neighbour, OneR) were used to classify the applications. The results vary between 30% and 100%.

Another paper [9] uses a bayesian machine learning algorithm with some more statistical parameters: packet length (min, max, mean), inter-arrival (min, max, mean), duration and packet count. Therewith, TOR and HTTP traffic could be classified with 85% of Precision and Recall.

All the papers cited above use only machine learning algorithms. The following two papers describe hybrid methods with additional preprocessing. In [10], at first a pattern based TLS detection is used to filter all TLS traffic. Only the TLS traffic is observed with the Naive Bayes machine learning algorithm. With this method 93% to 96% of HTTP and TOR traffic can be classified. Later we refer this as 'method 1'. A more advanced TLS preprocessing is done in [11]. The authors also use a pattern based TLS detection, but they observe the TLS session and using only application traffic without TLS handshake messages. Furthermore, they pay attention to the offset added by the Keyed-Hash Message Authentication Code (HMAC) and encryption. A statical offset of 21 bytes is used in their per-packet approach. The classification rate is between 81% to 100% for the ten observed applications. We refer to this as 'method 2' in the following sections.

Our own related work was on payload-based methods for application classification [12], with particular focus on protocol decoding. The protocol decoding inspects the network traffic and tries to decode each packet. If the decoded values match to the protocol description and if it fulfils all constraints of the protocol, the protocol is detected. This method is reliable but can only be used for unencrypted network traffic.

Another related work [13] [14] was on machine learning algorithms and investigated which kind of statistical information is useful for application classification. Furthermore, we investigated 20 different machine learning algorithms to find out which algorithms are suitable for network traffic analysis.

Besides TLS, other encryption techniques exist. [15] and [16] investigated traffic characteristic changes caused by Internet Protocol Security (IPsec) and encrypted Point-to-Point Tunneling Protocol (PPTP). The authors used the Naive Bayes, Support Vector Machines and C4.5 decision tree as machine learning algorithm for classification, but used two strategies for preprocessing the feature set. Either they split the traffic into encrypted and unencrypted traffic, or they do a normalisation of the feature set from encrypted traffic. The first strategy is used to approximate the feature set of the unencrypted traffic carried by the encrypted tunnel, to use only one classification model for the whole traffic. The second strategy use two classification models for each type of traffic. Their results show significant improvements in classification.

III. Hybrid Analysis Method

To identify TLS data in network traffic and to classify its content, we are using a hybrid method. First, to identify the TLS data, protocol decoding is used. As described in [12] and related papers, protocol decoding is a very reliable method for detecting TLS traffic. Additionally, some further information from the decoded TLS record headers are extracted to provide more precise statistical values regarding statistics gathered from the Transport Control Protocol (TCP) flow. The statistical values are used in conjunction with a machine learning algorithm to classify the protocol or application transmitted within TLS.

The TLS protocol is divided into five sub-protocols: the TLS Record Protocol, three handshaking protocols and the Application Data Protocol [1]. Application data messages are carried by the record layer protocol and are compressed, fragmented and encrypted with the negotiated master secret. A TLS session starts with a handshake. The handshake consists of the negotiation of a cipher suite, the exchange of certificates and keying material (e. g., Diffie Hellman). The application messages are treated as transparent data to the record layer.

Depending on the client and server configuration (e. g., usage and size of certificates), the number of packets exchanged during connection establishment varies. Additionally, the contents (except keying material) of the handshake messages of client and server are identical, even if a TLS connection is used by different applications. Thus, all the handshaking messages should never be considered for application classification.

After the TLS handshake, application data exchange starts. The application data is processed by the TLS layer as outlined in Figure 1. The application data can be compressed, but this is optional. The integrity of the data is protected by a HMAC, which is added to the application data. Then, the data with HMAC is encrypted and a TLS record header is added, which contains the TLS version, content length and type of content (application data or handshaking protocols). Due to TLS record header and HMAC, the payload of TLS is smaller than it seems on TCP level. The TLS record header has a constant size of 5 bytes, the length of the HMAC is one of six values: 0, 8, 16, 20, 32 or 48 bytes [17]. The used HMAC length depends on the used cipher suite which is negotiated during the handshake and can be provided by the protocol decoding. We propose considering this offset when using statistical data of TLS traffic. Compression was not used in all investigated network traffic and is frequently deactivated in the most applications. This is due to a security issue called CRIME. It was first described by [18] and later published as proof of concept exploit [19]. Thus, the compression has no influence on our statistical calculation and the classification results.

With the above description of the related parts of TLS, we can define four different methods for TLS application classification. The first two methods were already described in short at Section II. Method 1 [10] simply takes statistical values on TCP level. The TLS handshake, which is in general the same for all applications, is also included in the statistical calculation as the data records. The offset of the TLS record layer and HMAC is not removed. Large application data which was fragmented into several TCP segments will be counted as single application message, but a collection of small data records will be counted as one application message.

Method 2, as described in [11], skips the TLS handshake and starts evaluating the statistical data from the TLS stream after it detects the first data record by using packet inspection. Beginning from this point, it expects that every TCP segment

 ISBN: 978-1-61208-362-9

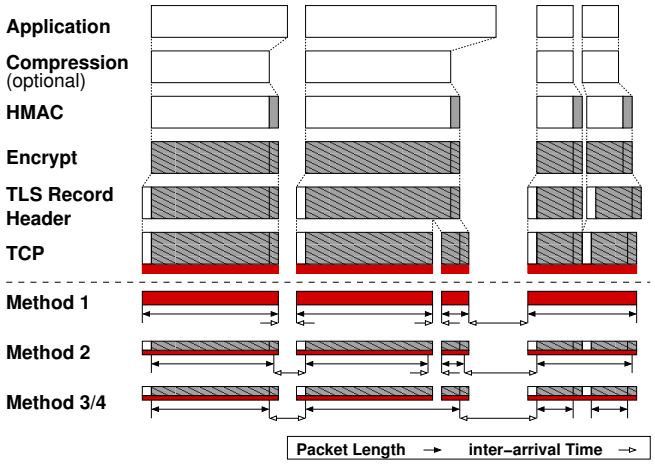

Figure 1: TLS fragmentation and network traffic statistics

transmits one TLS data record. The authors of [11] paid no attention to fragmentation. The analysis of the traffic used in Section IV showed that approximately 63% of TLS traffic is fragmented. In [11], a fixed offset of 21 bytes is removed from TCP data length because it is considered as TLS record header (5 bytes) and HMAC (16 bytes, e. g., for Message-Digest Algorithm 5 (MD5)).

We think it is necessary to extend the TLS traffic inspection to get more precise statistical data of the application messages. Therefore, we define a third and fourth method. Method 3 also skips the TLS handshake but inspects all succeeding TCP segments for TLS data records. Each data record is counted as one application message independent of the fragmentation. When a data record is split across several TCP segments, it is counted as one message. If a TCP segment contains different data records, each record is recognised as one application message. On method 3, a fixed offset of 21 bytes is used. The inter-arrival time between two data records within one TCP segment is considered as zero.

Method 4 works in the same manner as method 3, except it determines the concrete size of the used HMAC from the handshake. This results in more accurate statistical values but it increases the processing effort. This individual offset must be stored for each TLS connection. The four classification methods are outlined in Figure 1. It can be seen that method 1 and method 2 will capture values for packet length and inter-arrival time which do not match to the application data. The discrepancy between transmitted TCP segments and TLS data records can be large. The network traffic used in Section IV contains TLS data records which were split across up to ten TCP segments, but there were also TCP segments which contained up to ten data records. Only 37% of the TCP segments, which were captured in a campus network with many different clients and servers, contained one TLS data record. All other segments transmitted fragmented data. Methods 3 and 4 capture values which are very close to the application messages, while, method 4 provides the closest approximation.

We used the NaiveBayesUpdateable machine learning algorithm from WEKA Data Mining Software [20] and the statistical parameters described in [13] to process the statistical information from TLS data streams. For this work, we decided to use a packet based approach with supervised machine learning. As a result, the protocol decoding provides one data record for each TLS data record which is transmitted. These data sets are then used for learning and classification. Thus, the machine learning algorithm makes a classification decision for each TLS data record rather than for the whole flow.

We decided to use a bayesian classifier to get comparable results, because in [10] where method 1 is described, a bayesian classifier was used. Furthermore, bayesian classifiers are frequently used in the field of network traffic classification [21]. Nevertheless, the NaiveBayesUpdateable classifier can be exchanged with another machine learning algorithm and the approach will continue to work well.

IV. Experimental Results

This section discusses the classification results of the four applied methods to identify TLS encrypted traffic.

A. Metric

For evaluating and comparing the classification results, a metric is required. Various numbers of metrics, e. g., True Positive Rate, False Positive Rate, Recall and Precision, have been used in the past for evaluating traffic classification results. All of them are based on the following four metrics:

- true positive (t_p): objects belonging to protocol X and classified as protocol X
- true negative (t_n): objects not belonging to protocol X and not classified as protocol X
- false positive (f_p): objects not belonging to protocol X, but classified as protocol X
- false negative (f_n): objects belonging to protocol X, but not classified as protocol X

In this paper, the common used metrics Recall and Precision are applied to evaluate the performance of a classification method. The metric Recall defines the ratio of correct classified objects of a protocol to the total number of objects belonging to this protocol:

$$recall = \frac{t_p}{t_p + f_n} \quad (1)$$

Additionally, the accuracy of the classification is defined by the metric Precision, which defines the ratio of correct classified objects of a protocol to the number of all objects which were classified as this protocol:

$$precision = \frac{t_p}{t_p + f_p} \quad (2)$$

The primary goal of improving classification methods and an indicator for comparing the performance is to increase the Recall on the classification of protocols, and at the same time to increase the Precision.

 ISBN: 978-1-61208-362-9

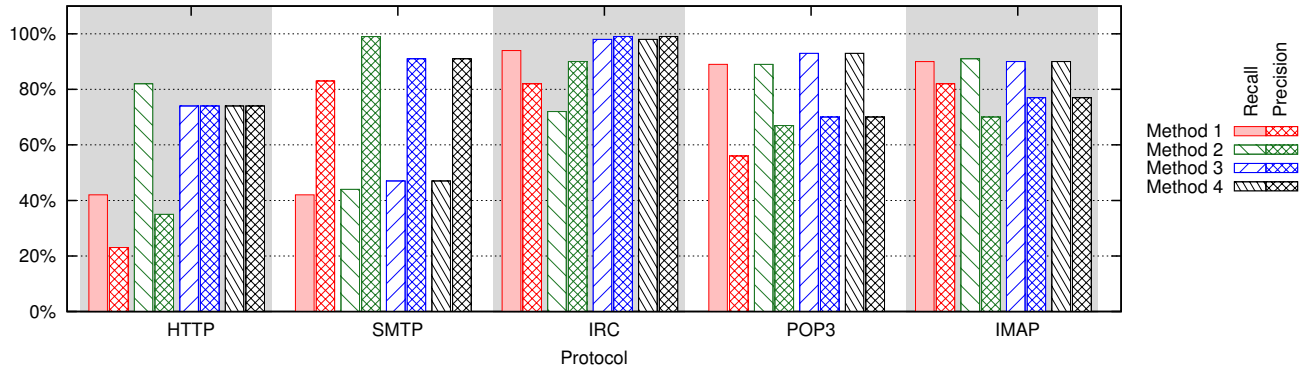

Figure 2: Classification results for all methods and protocols

TABLE I: Overview of the test data

Protocol	Method 1	Method 2	Method 3	Method 4
HTTP	21311	18516	20269	20269
SMTP	63880	60958	20456	20456
IRC	5735	5552	20800	20800
POP3	21287	17146	19682	19682
IMAP	15644	15420	20354	20354

B. Test data

TLS is used to protect a lot of applications and protocols. We decided to use the five protocols listed in Table I. These protocols was chosen because there are many publicly accessible servers to collect traffic with different server software and configurations. The HTTP traffic contains only ordinary HTML pages without Flash and video content. The e-mail protocol traces were captured at our university mail server as well as at our laboratory to capture conversations to publicly accessible server (e.g., Gmail). For the Internet Relay Chat (IRC) traces we also captured conversations to public servers. To get a realistic chat, the IRC client connected between five to ten minutes to a server which provides well-frequented IRC channels (e.g., Ubuntu support channel) without sending any chat message (only control messages). It received only chat messages of the connected channel, so the traces contain IRC talks between five to six hours for training and test, respectively.

Table I contains the data records determined from the traffic traces. We used a uniform distribution of data records (≈20.000) with respect to the data portions sent by the applications. The deviation from this values at Method 1 and 2 for SMTP or IRC results from ignoring the TLS fragmentation. Some TLS data records were split across up to ten TCP segments and some TCP segments contained up to ten data records. Table I shows only the test data. The training data for the machine learning algorithm contains the same amount of data records.

C. Classification results

Figure 2 shows the classification results of all applied methods. The protocols of the used traffic are placed on the x-axis, where the Recall and Precision results were displayed in percentage (y-axis) as bars. Each used method is represented by an own colour.

Starting with HTTP, method 1 classifies less then 50% of the HTTP traffic correctly with the Precision also lower than 25%, which implies that three out of four as HTTP classified packets are non-HTTP traffic. With method 2, Recall could be improved to 83% but nevertheless the Precision reaches only 36%. It is an improvement over method 1 but still two of three as HTTP classified packets are non-HTTP traffic. In general, it is not hard to implement a classification method with a high Recall, e.g., an algorithm that classifies each packet as HTTP reaches a Recall of 100%, but the Precision will be low according to the protocol distribution of the used traffic. In contrast, method 3 and method 4 gain a Recall of around 75%, which is less than method 2, but the Precision is improved to 74%. Thus, only one out of four as HTTP classified packets is non-HTTP traffic. This implies a higher reliability on the classification decision.

For SMTP, the Recall is continuously improved from 43% on method 1 up to 48% on method 2. Also, the Precision could be increased from 84% on method 1 up to 92% on method 2. The highest Precision could be realised with method 2 (99%). On all methods, the false negatives — SMTP traffic which was not classified as SMTP — were nearly entirely classified either as HTTP or Post Office Protocol version 3 (POP3).

With a Recall between 95% (method 1) and 99% (method 3 and method 4) the IRC protocol has the best classification results. Besides the high values for the Recall also the Precision with 82% (method 1) and 99% (method 3 and method 4) on a high level. Method 3 and method 4 achieve almost perfect classification results. Only method 2 decreases the classification accuracy; nearly all false negatives were classified as HTTP and Internet Mail Access Protocol (IMAP).

For POP3, the classification accuracy could be increased from method 1 to method 4. The recall could be enhanced from 89% to 94% and the Precision was enhanced from 57% to 71%. In contrast the Recall on IMAP was nearly constant at 91%, but the Precision was decreased from 82% on method 1 to 77% on method 4.

D. Future trend

The similarity and the missing enhancements on the classification accuracy between method 3 and method 4

 ISBN: 978-1-61208-362-9

are based on the applied cipher suites in the used traffic, respectively. The used traffic contains 11 different cipher suites, but only one cipher suite which is less than 1% of the whole traffic, uses a MD5 hash with a HMAC size of 16 bytes. All other cipher suites are using a Secure Hash Algorithm version 1 (SHA1) with a HMAC size of 20 bytes. Accordingly, nearly the entire traffic, there is only a fixed offset in the data record length between method 3 and method 4. This fixed offset causes no differences for the machine learning algorithm, and there is no improvement from method 3 to method 4 according to our data set. However, the Internet Assigned Numbers Authority (IANA) specified more than 300 cipher suites with the different HMAC sizes as described in section III. In consideration of the current lack of security, it can be supposed that stronger cipher suites will be used to secure the data. In this case, there will be a larger distribution of the used HMAC sizes and thereby the advantages of method 4 will be proved.

To determine if this assumption is right or not, we added some TLS traces from servers which support SHA256 for the HMAC to our test and training data set. Currently, only a small subset of all TLS servers support HMAC algorithms which are more secure than SHA1. Additionally, the TLS client makes a suggestion of the cipher suites to use, but only the newest versions support the stronger HMAC algorithms. Currently, only the latest web-browsers support TLS 1.2 with the new cipher suites [22]. Browsers take a pioneering role, while other applications do not support these cipher suites in the stable versions and providing support only within development versions (e.g., e-mail user agent *Mozilla Thunderbird* development version 30.0 beta 1 [23]). Furthermore, the web-browsers use their own TLS libraries, whereas other applications use the TLS libraries provided by the operating system or the used programming language (e.g., Java, C#). Only the latest versions of the operating systems and programming languages support TLS 1.2 [22] with the appropriate cipher suites. Thus, we concentrate on HTTP and IMAP. HTTP causes a significant amount of traffic in the Internet and our results of IMAP showed no improvements to the other methods.

To test the assumed enhancements of method 4 against those in method 3, we applied a set of HTTP and IMAP flows with cipher suites which are using SHA256 for calculating the HMAC with a length of 32 bytes. Due to the small set of these flows, the results can only give an indication of the behaviour for method 3 and method 4 on traffic with wider distribution of more secure cipher suites. The classification results on the test set with these new flows support our assumptions that method 4 leads to better classification results than method 3 when the investigated traffic includes different cipher suites with different HMAC sizes. Method 4 has achieved an enhancement between 2% and 3% on Recall and on Precision according to method 3. Nevertheless, further investigations with a well-balanced data set are required for a final confirmation of the enhancements of method 4 compared to method 3.

V. Conclusion

We compared four approaches for TLS application classification, each with different depth of TLS investigation. As a preparation for these methods, protocol decoding was used to filter TLS traffic from non-TLS traffic — to focus the analysis on dedicated applications — as part of our hybrid classification method. The results show an improvement of the classification accuracy according to Recall and Precision on the investigated protocols. For most applications, the reliability, which is based on Precision, could be increased from method 1 to method 4. The advantages of method 4 in contrast to method 3 will be shown on the deployment of other cipher suites on client site and server site. No significant differences could be determined between both methods on the underlying traffic. However, on an exemplary data set, an enhancement between 2% and 3% on Recall and on Precision could supported the assumption of method 4 as compared with method 3 on more secure cipher suites with larger HMACs.

As a result, method 3 and method 4 show a clear enhancement on the classification results according to Recall and Precision when compared to method 1 and method 2, which are well-known and commonly used methods for classifying TLS applications. Therefore, it is defenitly worth making the additional effort to processing the detailed statistic values for both methods. As other traffic classification methods have shown, it is expensive to improve an approach to gain the last remaining percentages which could achieve a perfect classification accuracy of nearly 100%.

In general, the used traffic is the critical fact in such evaluations, because the traffic covers only a limited part and is based on the underlying network. According to other evaluations, our classification results are in most cases not the best, but when repeating other approaches with our traffic, the results are partially quite different from the announced results. In conclusion, the stability of the statistical features strongly depends on the used traffic.

In future, the influence of the usage of compression for the classification accuracy has to be analysed, as well as the detection of further applications which are using TLS. Furthermore, the performance of other machine learning algorithms should be inspected for our presented methods.

Acknowledgements

We thank the reviewers for their valuable comments which helped to considerably improve the quality of the article. This work is supported by the European Regional Development Fund (ERDF) and the Free State of Saxony.

Gefördert aus Mitteln der Europäischen Union

Europa fördert Sachsen.
EFRE
Europäischer Fonds für regionale Entwicklung

References

[1] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008.

[2] "The Application Usage and Threat Report – An Analysis of Application Usage and Related Threats within the Enterprise," Palo Alto Networks, Tech. Rep. 10, Jan. 2013.

 ISBN: 978-1-61208-362-9

[3] C. McCarthy and A. Zincir-Heywood, “An investigation on identifying SSL traffic,” in Computational Intelligence for Security and Defense Applications (CISDA), 2011 IEEE Symposium on, April 2011, pp. 115–122.

[4] K. Ethala, R. Shesadri, and N. G. Renganathan, “The use of random forest classification and k-means clustering algorithm for detecting time stamped signatures in the active networks,” Journal of Computer Science, vol. 9, no. 7, 2013, pp. 875–882.

[5] G. Mujtaba and D. Parish, “Detection of applications within encrypted tunnels using packet size distributions,” in Internet Technology and Secured Transactions, ICITST, Nov 2009, pp. 1–6.

[6] M.-D. Wu and S. D. Wolthusen, “Network Forensics of Partial SSL/TLS Encrypted Traffic Classification Using Clustering Algorithms.” in IT Incident Managegent & IT Forensics, ser. LNI, vol. 140. Gesellschaft fuer Informatik, 2008, pp. 157–172.

[7] H. Liu, Z. Wang, and Y. Wang, “Semi-supervised Encrypted Traffic Classification Using Composite Features Set,” Journal of Networks, vol. 7, no. 8, 2012, pp. 1195–1200.

[8] G. Mujtaba, “Identification of Networked Tunnelled Applications,” Ph.D. dissertation, Loughborough University, May 2011.

[9] G.-L. Sun, F. Lang, M. Yang, and J. Hua, “Application protocols identification using Non-parametric Estimation method,” in Strategic Technology (IFOST), 2011 6th International Forum on, vol. 2, Aug 2011, pp. 765–768.

[10] G.-L. Sun, Y. Xue, Y. Dong, D. Wang, and C. Li, “An Novel Hybrid Method for Effectively Classifying Encrypted Traffic,” in GLOBECOM, IEEE, Dec 2010, pp. 1–5.

[11] L. Bernaille and R. Teixeira, “Early Recognition of Encrypted Applications,” in Proceedings of the 8th International Conference on Passive and Active Network Measurement, ser. PAM’07, 2007, pp. 165–175.

[12] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Müller, and K. Hänßgen, “A Survey of Payload-Based Traffic Classification Approaches,” Communications Surveys Tutorials, IEEE, vol. PP, no. 99, 2013, pp. 1–22.

[13] M. Finsterbusch, C. Richter, and J.-A. Müller, “Parameter Estimation for Heuristic Based Internet Traffic Classification,” in ICIMP 2012: The Seventh International Conference on Internet Monitoring and Protection, IARIA, Ed. Stuttgart, Germany: IARIA, 2012, pp. 13–22, ISBN: 978-1-61208-201-1 .

[14] C. Richter, M. Finsterbusch, K. Hänßgen, and J.-A. Müller, “Impact of Asymmetry of Internet Traffic for Heuristic Based Classification,” International Journal of Computer Networks (IJCN), vol. 4, no. 10, 2012, pp. 167–176.

[15] Y. Okada, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka, “Application identification from encrypted traffic based on characteristic changes by encryption,” in Communications Quality and Reliability (CQR), 2011 IEEE International Workshop Technical Committee on, May 2011, pp. 1–6.

[16] ——, “Comparisons of machine learning algorithms for application identification of encrypted traffic,” in Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on, vol. 2, Dec 2011, pp. 358–361.

[17] E. Rescorla. Transport Layer Security (TLS) Parameters. Internet Assigned Numbers Authority. [retrieved: Jan., 2014]

[18] J. Kelsey, “Compression and Information Leakage of Plaintext,” in Revised Papers from the 9th International Workshop on Fast Software Encryption, ser. FSE ’02. London, UK, UK: Springer-Verlag, 2002, pp. 263–276.

[19] J. Rizzo and T. Duong, “CRIME exploit – crime.py,” 2014, URL: https://gist.github.com/stamparm/3698401 [accessed: 2014-05-11].

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update,” SIGKDD Explor. Newsl., vol. 11, no. 1, Nov. 2009, pp. 10–18.

[21] T. Nguyen and G. Armitage, “A survey of techniques for Internet traffic classification using machine learning,” Communications Surveys Tutorials, IEEE, vol. 10, no. 4, 2008, pp. 56–76.

[22] Java Platform Group, “JDK 8 will use TLS 1.2 as default ,” 2014, URL: https://blogs.oracle.com/java-platform-group/entry/java_8_will_use_tls [accessed: 2014-05-20].

[23] Mozilla Foundation, “Thunderbird,” 2014, URL: www.mozilla.org/thunderbird [accessed: 2014-05-20].

 ISBN: 978-1-61208-362-9