Detecting encrypted traffic: a machine learning approach

Seunghun Cha and Hyoungshick Kim

Department of Software, Sungkyunkwan University, Republic of Korea
{sh.cha, hyoung}@skku.edu

Abstract. Detecting encrypted traffic is increasingly important for deep packet inspection (DPI) to improve the performance of intrusion detection systems. We propose a machine learning approach with several randomness tests to achieve high accuracy detection of encrypted traffic while requiring low overhead incurred by the detection procedure. To demonstrate how effective the proposed approach is, the performance of four classification methods (Naïve Bayesian, Support Vector Machine, CART and AdaBoost) are explored. Our recommendation is to use CART which is not only capable of achieving an accuracy of 99.9% but also up to about 2.9 times more efficient than the second best candidate (Naïve Bayesian).

1 Introduction

Conventional deep packet inspection (DPI) techniques are ineffective in analyzing encrypted traffic. Therefore, two different procedures could generally be used according to whether the inspected packets are encrypted or not. We first classify network packets into encrypted and unencrypted packets and then inspect only unencrypted ones in real time; encrypted ones are just recorded in log files for offline analysis. This approach might be helpful to improve the performance of intrusion detection systems by avoiding unnecessary efforts to analyze encrypted packets in their DPI engines. Moreover, the use of encryption is one of the important features for distinguishing bot traffic from normal traffic because payload encryption is a popularly used to hide bot communication [9]. Hence, detecting encrypted packets is becoming an essential part of intrusion detection systems.

To identify encrypted traffic, the most widely used techniques are pattern matching in packet payloads [21] and entropy analysis [7, 13, 20]. Although the entropy model is more effective for new botnet traffic, it also incurs both false positives and false negatives that cannot be ignored when a simple entropy-based approach is used for classification. To make matters worse, those approaches using the entropy estimation alone cannot deal with the distinction between encrypted and compressed traffic since both might have a high entropy measure [18]. In this paper, we will show that the entropy-based detection method performed very poorly on distinguishing encrypted traffic from compressed traffic through experiments with real datasets including a significant proportion of compressed packets (see Section 5).

Recently, some researchers [2, 3] proposed a more promising approach using machine learning to overcome the limitation of simple threshold models in entropy analysis. However, their designs have mainly focused on developing classification models using flow based features rather than packet based features where temporal information such as inter-arrival time is required. Although the flow-based detection systems have advantage over the packet-based systems because payload inspection is not needed anymore for detecting bot communication [10], a long time interval is required to extract time-based flow features from the network traffic.

Unlike their approach, we aim to develop a real-time classifier even with a single packet (or a few packets). To achieve this goal, we narrow down the problem scope to detecting encrypted traffic only rather than botnet communication. Our proposed technique can be incorporated into a firewall as its pre-filter to screen candidates.

To demonstrate the effectiveness of our approach, we experimented with four well-known machine learning techniques (Naïve Bayesian, Support Vector Machine, CART and AdaBoost) compared with a simple entropy-based detection method. For experiments, we used a real traffic dataset (HTTP: 161,948, FTP: 1,309, Telnet: 205, SSH: 30,185 packets) collected from a honeypot. The experimental results showed that the proposed machine learning based detection methods significantly outperformed the simple entropy-based detection method. In particular, CART produced the best results, which is not only capable of achieving a F-measure of 0.997 but also up to about 2.9 times more efficient than the second best candidate (Naïve Bayesian). Moreover, since the results of CART algorithm can directly be used to generate a series of `if-then` rules from a binary decision tree representation, we highly recommend using CART for building a high-performance DPI system.

The rest of this paper is organized as follows. In Section 2, we survey related work in developing encrypted packet classification techniques. In Section 3, we present the proposed detection technique using a machine learning algorithm with randomness tests. In Section 4, we briefly explain the classification algorithms used in our experiments. Then, in Section 5, we evaluate their accuracy and performance by conducting experiments in real-world environments. Finally, we conclude in Section 6.

2 Related work

A *botnet* is a network of compromised computers controlled by a remote attacker called *botmaster*, which is typically used for performing illegitimate activities such as distributed denial-of-service (DDoS) attacks, spam emails distribution. For botnet communication, the IRC protocol was the most widely used due to its simple construction. Recently, however, attackers are moving towards using encryption (from simple XOR [12] to sophisticated AES-256 [15]), which makes it much harder to detect their communication against firewalls and intrusion

detection systems [16, 20]. Thus detecting encrypted traffic becomes one of the most challenging issues in designing a bot detection tool.

Entropy-based classification is the most common method to distinguish encrypted parts from plaintexts. Lyda and Hamrock [13] used entropy analysis to identify encrypted and packed malware, but they only focused on offline executable files. Dorfinger et al. [7] proposed a classification method based on entropy estimation to detect encrypted traffic used for Skype and eDonkey. For each packet, their proposed system compares the entropy of the packet with the estimated entropy of uniformly distributed random payload of the same length. If the difference is greater than a pre-defined threshold, the packet is classified as unencrypted traffic. White et al. [19] revisited this problem through intensive experiments to classify opaque traffic including not only encrypted but also compressed communication. They also used entropy-based tests and demonstrated that their technique is able to identify opaque data with about 95% accuracy, on average, while examining less than 16 bytes of payload data. Zhang et al. [20] also proposed a similar technique using the entropy measure and showed that the proposed technique might be effectively deployed to prevent encrypted bot communication. However, those approaches had to ignore the difference between both encrypted and compressed traffic since both might have a similar level of entropy estimation. Wang et al. [18] addressed this limitation in using the entropy estimation alone for detecting encrypted contents and proposed an alternative metric to identify the distinction between encrypted and encoded/compressed data.

We extend their work into a more sophisticated model using machine learning algorithms and several randomness tests; while Wang et al. [18] empirically selected a threshold for the chi-square randomness test result, we build a detector with machine learning algorithms and several randomness tests to improve the accuracy of encrypted communication detection.

Although the idea of using machine learning algorithms is not new in the field of traffic classification, previous studies [1, 3] focused on using flow based features rather than packet based features, which inherently requires an observation time interval to monitor time based flow features from the network traffic. The scenario for flow based classification is fundamentally distinct from the setting we explore in this work: for real-time analysis on the network, it is preferred to individually process every packet without requiring the maintenance of state for every connection.

3 Encryption detection with randomness tests

The key idea is simple: When the proposed system receives a packet, the header and payload of the packet are first separated, and the randomness of the payload is analyzed by running several randomness tests. This is because the payload of a packet is only encrypted for hiding bot communication while its header is in the clear (see Figure 1). Surely, in practical intrusion detection systems, packet header information is generally used before inspecting the payload of packets.

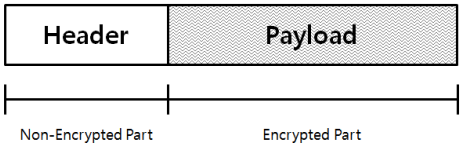


Fig. 1: A typical form of packet

The proposed method may be applied only to packets when it is difficult on the packet header alone to decide whether the packets are encrypted or not.

With the randomness test results, we can determine the nature of the packet (i.e., the encrypted payload or not) using a classifier trained with data from previous test results. Unlike previous studies [7, 13, 20] using a pre-determined threshold of entropy alone, we use machine learning techniques to improve the accuracy of a classifier with the following three randomness tests that are commonly used, particularly for evaluating the randomness of pseudorandom number generation algorithms [17].

- **Entropy:** The Shannon entropy is a traditional estimator of measuring the unpredictability of an information stream. Let X be a random variable that takes on the possible values x_1, x_2, $\cdots$, x_n. Here, x_i is the ith byte value because we calculated the entropy in bytes. The Shannon entropy $H(X)$ is defined by the following formula:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i) \tag{1}$$

  where $p(x_i)$ is the probability that $X = x_i$.
- **Chi-square test:** The chi-square distribution is used to test how well a sample fits a theoretical distribution (e.g., Poison distribution). Here, for ith byte value, the number of the observed values o_i is compared with that of expected values e_i from a uniform distribution. The chi-square distribution χ^2 is defined by the following formula:

$$\chi^2 = \sum_{i=1}^{n} \frac{(o_i - e_i)^2}{e_i} \tag{2}$$

  where n is the number of values.
- **Arithmetic mean:** The arithmetic mean can simply be calculated by summing all of the byte values in the tested data (i.e., a packet) and dividing by the number of those byte values. If the data are generated from a uniform distribution, the arithmetic mean should be about 127.5. The arithmetic mean A is defined by the following formula:

$$A = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3}$$

  x_i is the ith byte value in a packet.

The results of those randomness tests are used to construct a classifier to detect encrypted traffic. Not surprisingly, a classification method should be carefully selected based on constraints such as the desired level of accuracy, the time available for training and testing, and the nature of classification problems. The classification methods used in our experiments are described in the following section.

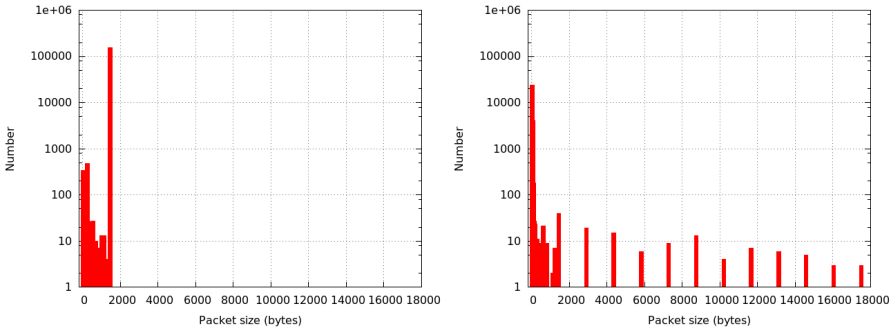
4 Applying Classification Methods

We select four different classification methods which are popularly used for many applications. We carried out a number of experiments to evaluate those classification methods and find the best performing one. This section introduces those classification methods as follows:

- **Naive Bayesian:** Naïve Bayesian (NB) classification [6] is one of the most successful learning algorithms for text classification. Based on the Bayes rule, which assumes conditional independence between classes, this algorithm attempts to estimate the conditional probabilities of classes given an observation. The joint probabilities of sample observations and classes are simplified because of the conditional independence assumption. While this Bayes rule assumption is often violated in practice, NB is known to perform well in many security applications such as spam email filtering.
- **Support Vector Machine:** Support Vector Machine (SVM) [5] is known as one of the best supervised learning techniques for classification with high dimensional feature space and small training set size. The idea is to find the optimal separating hyperplanes that classify data by maximising the geometric margin space between the classes' closest points. SVM is receiving great attention due to some excellent performance it has achieved on real-world applications [11].
- **Classification and Regression Trees:** Classification and Regression Trees (CART) [4] is an interesting non-parametric technique to build a binary decision tree based on historical data. CART constructs a binary tree by repetitively splitting the high dimensional space of the objects in the training set into subspaces so that the observations within each subspace are more homogeneous than the high dimensional space.
- **Adaptive Boosting algorithm:** AdaBoost [8], short for "Adaptive Boosting", is an iterative process that tries to improve the classification results by learning the results from a set of classifiers (often referred to as "weak learners"), in that their weights are adaptively updated according to the errors in previous iterations. The essence of AdaBoost is to combine multiple weak classifiers for constructing a single strong classifier.

5 Experiments

The aim of our experiments was to demonstrate the feasibility and effectiveness of our approach, and determine the best performing classification method. We



(a) HTTP, FTP and Telnet (b) SSH

Fig. 2: Histogram of packet sizes in the datasets.

Table 1: Proportions of content types in HTTP, FTP and Telnet packets

Compressed	Octet-stream	Others
78.15%	0.32%	21.53%

tested real network traffic using the four classification methods described above (NB, SVM, CART and AdaBoost).

We collected real packets captured by a honeypot (taken from the "Capture the hacker 2013 competition") and filtered TCP packets only, which consist of 161,948 HTTP, 1,309 FTP, 205 Telnet and 30,185 SSH packets. For plaintexts, we used the packets for HTTP, FTP, and Telnet application protocols while we used only encrypted payload in SSH packets for ciphertexts. Figure 2 shows the distribution of packet sizes, respectively, for those datasets; the y axis is shown in log-scale for improved visualization.

In terms of the packet size, we can see the differences between them; all packets for HTTP, FTP, and Telnet protocols are less than 2,000 bytes while a majority (about 93.6%) of packets for SSH protocol are less than 64 bytes long and large packets were sometimes used. We surmise that the default packet size differences in underlying protocols may explain this. The default Maximum Segment Size (MSS) of 1,460 bytes (which covers about 95.5% of our dataset) is generally used for HTTP, FTP, and Telnet protocols while the minimum packet size of 28 bytes is most popularly used for SSH protocol because the data exchanged might be mostly short commands for botnet communication [14]. Interestingly, the packets used in our experiments include a significant proportion (about 78.15%) of compressed ones (using either `gzip` or `bzip2`) in HTTP, FTP and Telnet packets (see Table 1).

We assigned the ciphertexts with 'Positive' answers (P) and the plaintexts with 'Negative' answers (N). True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) can be summarized as below:

Table 2: Performance of classification methods.

	Accuracy	Precision	Recall	F-measure	Training	Testing
NB	0.990	0.953	0.987	0.970	0.056s	1.604s
SVM	0.997	**0.998**	0.981	0.989	559.097s	15.034s
CART	**0.999**	0.997	0.996	**0.997**	0.528s	0.560s
AdaBoost	0.998	0.996	0.994	0.995	7.789s	125.198s
Entropy	0.159	0.157	**0.998**	0.272	**0.052s**	**0.006s**

- TP: ciphertexts correctly classified as ciphertexts;
- FP: plaintexts incorrectly classified as ciphertexts;
- TN: plaintexts correctly classified as plaintexts;
- FN: ciphertexts incorrectly classified as plaintexts.

To evaluate the performance of classifiers, we use four measures as follows:

- **Accuracy**: the proportion of correctly classified packets; $(TP+TN)/(P+N)$
- **Precision**: the proportion of packets classified as ciphertexts that actually are ciphertexts; $(TP)/(TP + FP)$
- **Recall**: the proportion of ciphertexts that were accurately classified; $(TP)/(TP + FN)$
- **F-measure**: the harmonic mean of *precision* and *recall*; $(2 * Precision * Recall)/(Precision + Recall)$

We also measured the running time of the classifiers to show the relative efficiency of the classification methods. The classifiers were implemented using the `scikit-learn` library in Python. The experiments were conducted with an Intel Core i7 (3.60GHz), running on the virtual machine of Ubuntu 14.04 LTS with 4GB RAM.

For classification, we used a 10-fold (stratified) cross-validation where the training samples are partitioned into 10 equal-sized groups with similar class distributions. Table 2 shows the performance of the four classification methods. The running time for training and testing is presented as the mean time using the whole dataset in a group from the 10 independent experiments. Moreover, we compared the tested classification methods with the simple entropy-based detection to show the superiority of the machine learning based detection algorithms over the conventional detection technique. For entropy-based detection, a threshold was chosen to maximize F-measure for each training set.

Overall, except for the entropy-based detection used as a baseline, all methods performed well (see Table 2). The entropy-based detection produced the worst results (0.272 in F-measure). Probably, the performance of entropy-based detection was strongly affected by the significant proportion (about 78.15%) of compressed packets in the dataset. The results confirmed that the entropy estimation only is not sufficient to distinguish encrypted contents from compressed

Table 3: Classification results with the first k bytes of each packet.

k	NB		CART	
	Accuracy	F-measure	Accuracy	F-measure
1	0.843	0.001	0.843	0.001
4	0.843	0.000	0.843	0.004
16	0.646	0.248	0.842	0.014
32	0.681	0.362	0.962	0.868
64	0.962	0.869	0.983	0.947
100	0.969	0.893	0.996	0.986
All	0.990	0.970	0.999	0.997

contents even though it is significantly faster than machine learning based detection algorithms in testing time.

Among machine learning based detection algorithms, CART not only produced the best accuracy results (about 0.999 in accuracy and 0.997 in F-measure), but was most efficient in testing (about 0.560s) which is up to approximately 2.9 times better than the second best candidate (NB). Although AdaBoost also performed well in terms of accuracy and F-measure, when we consider how computationally expensive the testing phase of AdaBoost is (about 125.198s), we would not recommend using AdaBoost. For the same reason, SVM is not suitable for detecting encrypted traffic in real-time. Therefore, we here focus only on the analysis of CART and NB.

Although the testing time of each classification method is relatively efficient compared with its training time, calculation of the three randomness scores (Entropy, Chi-square test, Arithmetic mean) for testing cannot be easily ignored where the running time of the calculation increases (linearly) with the size of packet observed. Thus, in terms of efficiency, it might be desirable to classify a packet with only a few (front) bytes of the packet instead of using all of the packet data. We performed additional experiments to discuss how the performance of classification methods may change with the size of observed part of the packet for testing and to identify the optimal size. Table 3 and 4 show the results with variation in the size of the observed part of the packet.

In Table 3, we can see that two classification methods (CART and NB) still performed well even with only a part of the packet (when $k \geq 64$) rather than using all of the packet data. Although those methods produced poor performance results (< 0.015) in F-measure with a very few bytes (when $k = 1, 4$ or 16), the detection accuracy of those classification methods dramatically increases when $k \geq 32$ (for CART) or $k \geq 64$ (for NB). Moreover, it also led to improvement in running time for randomness tests (see Table 4); the total running time of the randomness tests was reduced from 0.349ms (millisecond) to 0.263ms on average,

Table 4: Running time for randomness tests with the first k bytes of each packet (μ: mean, σ: standard deviation). Byte, Entropy, Chi-sq., and Arith. represent the time for analyzing byte frequency distribution, entropy, Chi-square distribution and arithmetic mean, respectively.

k	Byte(μ)	Byte(σ)	Entropy(μ)	Entropy(σ)	Chi-sq.(μ)	Chi-sq.(σ)	Arith.(μ)	Arith.(σ)	Total(μ)	Total(σ)
1	0.018ms	7.0e-8ms	0.147ms	9.2e-7ms	0.073ms	3.9e-7ms	0.019ms	7.0e-8ms	0.258ms	2.0e-6ms
4	0.019ms	6.0e-8ms	0.148ms	1.1e-6ms	0.073ms	3.7e-7ms	0.019ms	6.0e-8ms	0.259ms	2.2e-6ms
16	0.019ms	7.0e-8ms	0.146ms	1.2e-6ms	0.073ms	4.2e-7ms	0.019ms	7.0e-8ms	0.257ms	2.6e-6ms
32	0.021ms	8.0e-8ms	0.154ms	1.0e-6ms	0.073ms	3.8e-7ms	0.019ms	8.0e-8ms	0.267ms	2.2e-6ms
64	0.023ms	8.0e-8ms	0.148ms	1.1e-6ms	0.073ms	5.7e-7ms	0.019ms	7.0e-8ms	0.263ms	2.6e-6ms
100	0.025ms	2.5e-7ms	0.147ms	1.2e-6ms	0.073ms	4.8e-7ms	0.019ms	7.0e-8ms	0.265ms	2.9e-6ms
All	0.106ms	2.3e-6ms	0.148ms	9.8e-7ms	0.075ms	3.8e-7ms	0.021ms	7.0e-8ms	0.349ms	4.7e-6ms

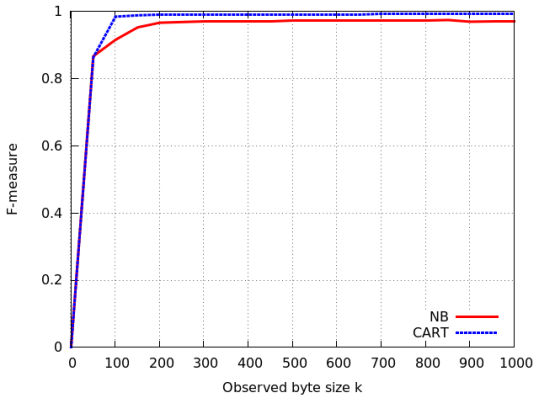
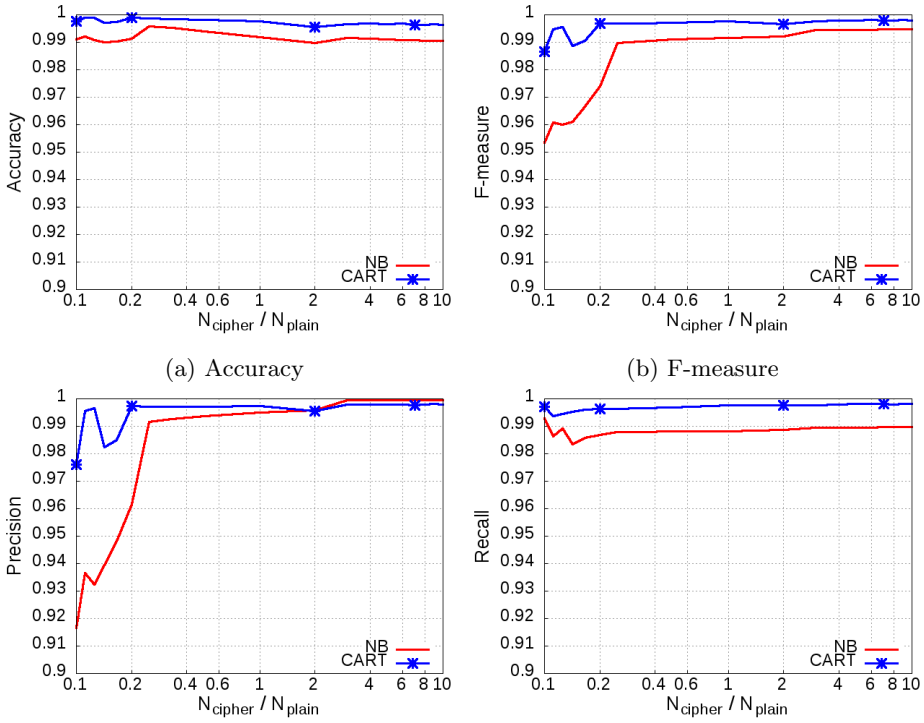


Fig. 3: F-measure of classifiers with randomly selected k bytes of each packet.

which was significantly affected by the analysis of byte frequency distribution (Byte) rather than other tasks.

To avoid this detection strategy, an attacker's best response is to generate dummy plaintext data and add the data into the first k bytes of each packet if the attacker already knows the system parameter k in a detection system. Therefore it is not enough to simply observe the first k bytes of each packet. Alternatively, we consider using a randomized approach with k bytes randomly selected from the whole packet data. The experimental results in F-measure are shown in Figure 3. Overall, CART produced the better results compared with NB. Also, we can see that the accuracy of two classification methods (CART and NB) was improved as k increased. In particular, both curves increase rapidly toward 1 until $k < 100$ and then tends to be smooth and flat when $k \geq 100$. This implies that it might be enough to use a randomly chosen part of a packet (e.g., with



(a) Accuracy (b) F-measure

(c) Precision (d) Recall

Fig. 4: Performance of classification methods with varying N_{cipher}/N_{plain} ratio.

at least 100 bytes long) instead of using the whole packet data to speed up the detection test procedure for each packet.

To examine the influence of ratio of ciphertext (N_{cipher}) to plaintext (N_{plain}), we performed additional experiments with varying N_{cipher}/N_{plain} ratio by artificially controlling the number of generated encrypted packets. However, we still used the fixed ratio of plain to cipher packets. Figure 4 shows how the performance of classification methods were changed over the different ratio of N_{cipher}/N_{plain}.

Overall, two classification methods (CART and NB) are scalable with varying N_{cipher}/N_{plain} ratio except when the ratio of N_{cipher}/N_{plain} is less than 0.2. This is because those methods demonstrate a significant decrease in precision when the number of ciphertexts is greatly less than the number of plaintexts. Hence, we would not recommend using the proposed technique for such conditions. Our recommendation would still be to use CART except those cases, even though NB is slightly better than CART in terms of precision when the ratio of N_{cipher}/N_{plain} is greater than 2.

6 Conclusion

Distinguishing encrypted data from non-encrypted packets still remains an important research problem in the field of network security. We present a machine learning approach based on three randomness tests (Entropy, Chi-square, Arithmetic mean).

For performance evaluation, we used a total of 193,647 packets (163,462 plain and 30,185 encrypted). The packets ware tested with four machine learning based classification methods —Naïve Bayesian, Support Vector Machine, CART and AdaBoost— the simple entropy-based detection method to find the most proper classification method to detect encrypted traffic. The machine learning based detection methods significantly outperformed the entropy-based detection method. In particular, our recommendation would be to use CART that not only produced the best accuracy results (0.997 in F-measure) but was most efficient in testing (about 0.560s).

In future work, we plan to increase the size of the dataset and investigate any changes in performance. It would also be interesting to deploy the proposed approach on a real intrusion detection system.

Acknowledgements

This research was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government (MSIP) (No. B0717-16-0116, Development of information leakage prevention and ID management for secure drone services)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (No. 2014R1A1A1003707)

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-R0992-16-1006) supervised by the IITP (Institute for Information & communications Technology Promotion)

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No.R-20160222-002755, Cloud based Security Intelligence Technology Development for the Customized Security Service Provisioning)

References

1. R. Alshammari and A. N. Zincir-Heywood. Investigating two different approaches for encrypted traffic classification. In *Proceedings of the 6th IEEE Annual Conference on Privacy, Security and Trust*, 2008.
2. R. Alshammari and A. N. Zincir-Heywood. Generalization of signatures for ssh encrypted traffic identification. In *IEEE Symposium on Computational Intelligence in Cyber Security*, 2009.

3. R. Alshammari and A. N. Zincir-Heywood. Can encrypted traffic be identified without port numbers, IP addresses and payload inspection? *Computer Networks*, 55(6):1326–1350, 2011.
4. L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.
5. N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 2000.
6. P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.
7. P. Dorfinger, G. Panholzer, and W. John. Entropy Estimation for Real-time Encrypted Traffic Identification. In *Proceedings of the 3rd International Conference on Traffic Monitoring and Analysis*, 2011.
8. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.
9. G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee. Active botnet probing to identify obscure command and control channels. In *Annual Computer Security Applications Conference*, 2009.
10. F. Haddadi, D. Le Cong, L. Porter, and A. Zincir-Heywood. On the Effectiveness of Different Botnet Detection Approaches. In *Proceedings of the 11th International Conference on Information Security Practice and Experience*, 2015.
11. M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.
12. T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and Mitigation of Peer-to-peer-based Botnets: A Case Study on Storm Worm. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, 2008.
13. R. Lyda and J. Hamrock. Using entropy analysis to find encrypted and packed malware. *Security & Privacy, IEEE*, 5(2):40–45, 2007.
14. A. Ramachandran, S. Seetharaman, N. Feamster, and V. Vazirani. Fast Monitoring of Traffic Subpopulations. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, 2008.
15. S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich. Analysis of the storm and nugache trojans: P2P is here. *;login:*, 32(6), 2007.
16. B. Thuraisingham. Data mining for security applications: Mining concept-drifting data streams to detect peer to peer botnet traffic. In *International Conference on Intelligence and Security Informatics*, 2008.
17. J. Walker. Ent: A pseudorandom number sequence test program. http://www.fourmilab.ch/random/, 2008.
18. R. Wang, Y. Shoshitaishvili, C. Kruegel, and G. Vigna. Steal This Movie - Automatically Bypassing DRM Protection in Streaming Media Services. In *Proceedings of the 22Nd USENIX Conference on Security*, 2013.
19. A. M. White, S. Krishnan, M. Bailey, F. Monrose, and P. Porras. Clear and Present Data: Opaque Traffic and its Security Implications for the Future. In *Proceedinds of the Network and Distributed Systems Security Symposium*. The Internet Society, 2013.
20. H. Zhang, C. Papadopoulos, and D. Massey. Detecting encrypted botnet traffic. In *Conference on Computer Communications Workshops*, 2013.
21. Y. Zhang and V. Paxson. Detecting backdoors. In *Proceedings of the 9th Conference on USENIX Security Symposium*, 2000.