Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2014, Article ID 197961, 8 pages
http://dx.doi.org/10.1155/2014/197961

Research Article

Stateless Malware Packet Detection by Incorporating Naive Bayes with Known Malware Signatures

Ismahani Ismail, Sulaiman Mohd Nor, and Muhammad Nadzir Marsono

Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia

Correspondence should be addressed to Ismahani Ismail; ismahani@fke.utm.my

Received 8 January 2014; Revised 13 March 2014; Accepted 15 March 2014; Published 15 April 2014

Academic Editor: Sebastian Ventura

Malware detection done at the network infrastructure level is still an open research problem ,considering the evolution of malwares and high detection accuracy needed to detect these threats. Content based classification techniques have been proven capable of detecting malware without matching for malware signatures. However, the performance of the classification techniques depends on observed training samples. In this paper, a new detection method that incorporates Snort malware signatures into Naive Bayes model training is proposed. Through experimental work, we prove that the proposed work results in low features search space for effective detection at the packet level. This paper also demonstrates the viability of detecting malware at the stateless level (using packets) as well as at the stateful level (using TCP byte stream). The result shows that it is feasible to detect malware at the stateless level with similar accuracy to the stateful level, thus requiring minimal resource for implementation on middleboxes. Stateless detection can give a better protection to end users by detecting malware on middleboxes without having to reconstruct stateful sessions and before malwares reach the end users.

1. Introduction

Content based malware detection can be done using antivirus solution at the user's end station. This requires the codes (as packets payloads) to be fully constructed into files at the end station for malware detection. Even if partial codes could be detected by the antivirus, the codes have already reached the end station. Detection at this level has its limitations since complete observability; that is, reassembly and stateful detection on the Internet byte streams are required [1]. As network speed increases, reassembly inside network nodes, even on network boundaries, requires increasing computational resources in terms of computation overhead [2]. Therefore, stateless detection is a better alternative to detect malware whilst the codes are in transit between the source (router or gateway) and destination to relax stateful restrictions such as packets buffering and reassembling. This provides early detection and the possibility of not having to construct the whole flow for malware detection.

The use of intrusion detection system (IDS) [3] and intrusion prevention system (IPS) [4] is a popular retrofit strategy to complement the limitations of malware detection at end stations. However, the evolution of today's modern malware makes these signature based methods ineffective in detecting fast spreading sophisticated malware (e.g., polymorphic malware [2]). Machine learning technique has the potential to successfully detect zero-day malware [5, 6]. This technique commonly needs feature extraction, feature selection, and classification. Instead of using plain text data, n-gram is commonly used [5, 6] to represent informative features. This n-gram technique has the ability to capture implicit features of the malware contents that are difficult to detect explicitly. However, the issues when using n-gram features are computational overhead and feature search space. Therefore, newer approaches are needed to overcome the stated shortcomings.

This paper proposes the incorporation of Naive Bayes training with Snort signatures [7] to detect malware at the packet level. The signatures are applied to the machine learning technique to potentially complement the protection of network using exact pattern matching techniques such as IDS, IPS, and the method proposed by Varghese et al. [1]. Comparison with the detection at the stateful level is also included to assess the effectiveness of the proposed work. Based on offline traffic, the measurement of detection

accuracy at the stateless level is around 2% lower compared to the stateful level. This finding shows the viability of the proposed approach to detect malware in transit.

2. Related Works

Malware strings can be detected at the network infrastructure level as well at the host level. Network security operators today have deployed security middleboxes such as IDS and IPS to protect networks against incoming and outgoing malware traffic [3, 4]. IDS has been deployed by almost every major organization motivated by the security awareness. IDS such as open source tool Snort [7] usually associates detection with alerts to the administrator. However, the emergence of fast-moving attacks (such as worms) can cause considerable damage while the IDS is raising an alert, well before the human administrators can respond. This caused the IDS market to morph into the so-called IPS market. IPS allows necessary action to drop any packet that matches any rule, while IDS can simply tap the data to generate alerts.

The problem with IDS/IPS is the evasion attacks [8]. Attackers change the contents of the attacks using fragmented packets scenario which can bypass the pattern matching process. This problem is studied and tackled by Varghese et al. [1]. The work suggested a new detection scheme called Split-Detect to complement the traditional IDS/IPS in detecting evasion attacks successfully. The basic idea is to provide a fast detection by splitting IDS/IPS signatures into signature substrings, using exact string match and two processing paths, fast path and slow path. The signature pieces are used to detect the presence of attacks in packets payload in the fast path. If the packets are not detected, the fast path processor releases the packets as benign packets. If the packets contain plausible evidence (e.g., contain a signature piece), the suspicious packets are diverted to the slow path processor for further action either to forward the packets to the receiver or drop the packets and reset the TCP connection. The Split-Detect scheme has been proven capable of detecting all byte streams evasions. However, the requirement of two processor paths may seem too high in terms of the operational cost on middleboxes.

Content based classification by [5, 6] was host based malware detection where they observed byte n-gram of malware executables at end stations. Classification based on contents especially when using n-gram features requires huge training features which need large search space and high computational overhead [9]. High computational overhead is not appropriate for network based detection especially when the model needs to be retrained over time to reflect the changing trends of traffic in a network. This paper proposes the use of n-gram and machine learning techniques to detect unknown malware. The reasons of using byte n-grams are the following. First, n-gram has the capability to extract malware bytes implicitly from network traffic in form of packets or byte stream. Therefore, the possibility to detect malware contents in unexpected patterns is higher than in plain text patterns. Second, detection on byte n-grams allows the detection of malware at the network level (e.g., middleboxes) instead of at the host level only. The detection can be done on the fly by extracting malware packet payload whilst the packet is transmitted in the network.

Based on the ideas of splitting signatures into pieces to detect attacks by Varghese et al. [1], similar idea is applied in this paper. The new approach proposed here is for Snort signatures [7] to be extracted as signatures n-gram (fixed length and overlapping n-gram). The main reason of using these signatures is to limit features to be trained to address the issue of computational overhead practically. Second, the use of known malware signatures as the features to generate model will likely detect malware more frequently. Therefore, the use of signatures n-gram and aggregation (classification) technique potentially does not require separate processing paths (fast and slow paths) and is expected to be able to detect zero-day malware.

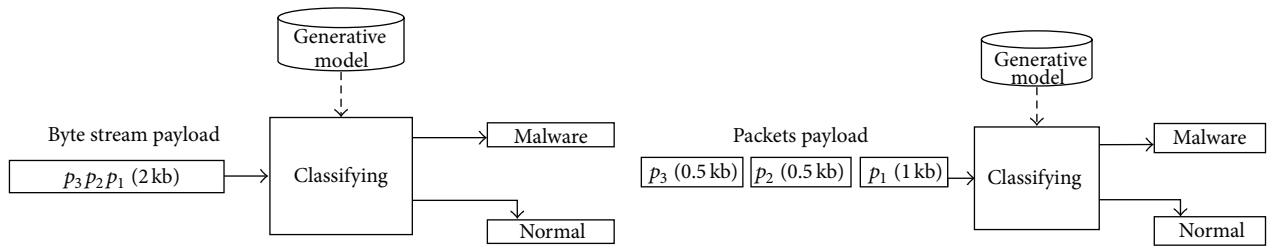
3. Snort Signatures Assisted Classification Methodology

3.1. Deployment of Generative Model. Data transmission (e.g., TCP traffic) occurs between hosts after the connection is established through a three-way handshake process [10]. TCP packets that belong to the same flow (e.g., a flow p) during the transmission between the two machines will be reassembled to the original flow at the destination machine. The stateful data are observed as a complete byte stream (e.g., packets payload after being reassembled at the destination host), whilst the stateless data as individual packets payload (e.g., packets payload before reassembly) [10]. The difference of classification process of payload at the stateful and the stateless level is illustrated in Figure 1.

At the stateful level, complete byte streams are classified either as malware or normal as illustrated in Figure 1(a). For example, a flow that contains payload with the size of 2 Kb is observed by generative model to determine whether the flow is malware or normal. In contrast, at the stateless level, the same payload remains as individual packets payload as illustrated in Figure 1(b). The payloads with the size of 1 Kb, 500 b, and 500 b are carried by separate packet and classified independently by the generative model.

Compared to the stateful level, stateless detection is different in the sense that all packets are partially classified whilst the packets are transmitted through the network. In other words, stateless detection is done without the need for packets reassembly.

Figure 2 shows the top level learning process incorporating Snort signatures [7] to generate the model (Naive Bayes). The generative model is generated after learning some samples traffic (training set). In this supervised learning approach, samples traffic is correctly labeled and kept in their respective classes. The samples are preprocessed for each class by only extracting the payload (excluding the header information) to n-gram features. For example, a payload sequence, of dd 25 64 fe 69 d3 a3 1f, generates the corresponding 4 grams of dd2564fe, 2564fe69, and 64fe69d3 as illustrated in Figure 3. We set n-gram of size four since our study in [5, 6] observed this number as the optimum feature size for text classification. The extraction process produces



(a) Payload classified in byte streams form

(b) Payload classified in packets form

Figure 1: Classification process for stateful and stateless level.

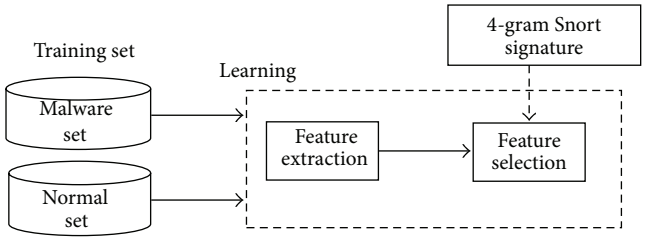


Figure 2: Learning process incorporated by known malware signatures.

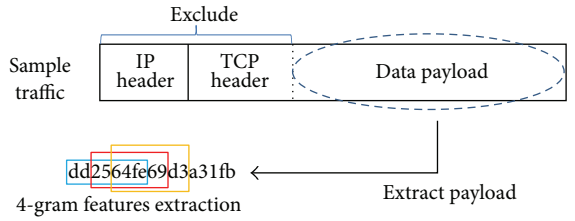


Figure 3: Extract 4-gram features from traffic payload in training set.

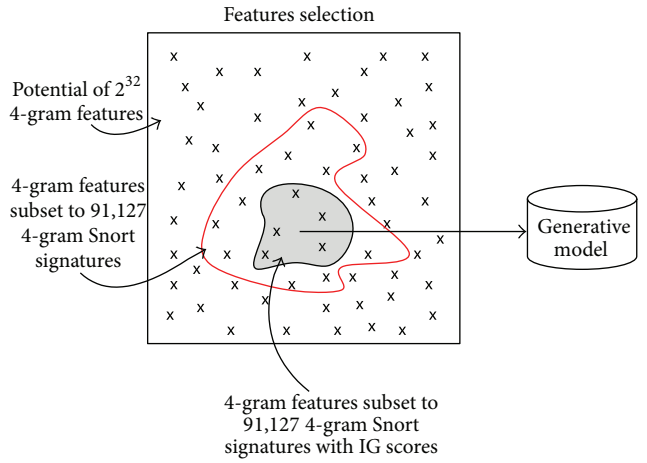


Figure 4: Training dataset based on selected features (subset of Snort signatures) with high IG.

million numbers of features (e.g., for n size of 4, the number of features is 2^{4*8} [11]).

Feature selection selects the most informative features to be trained. In this paper, the number of features is pruned based on the Snort signatures. The original signatures consist of variable length of signature string. The same feature extraction process is applied to the signatures strings to produce 4-gram signatures. Signatures strings shorter than 4 grams are eliminated. After eliminating duplicate 4 grams, only 91,127 4-gram signatures are used to assist the feature selection as shown in Figure 2. Figure 4 shows the process of generating the generative model with the assistance of Snort signatures. Only 4-gram features that are subset to the Snort signatures are selected. Nevertheless, certain features out of the 91,127 features can be further reduced using other feature selection techniques such as information gain (IG) [12]. IG is a statistical measure and it is computed separately for each feature. IG value is always nonnegative, and higher scores indicate more discriminative features. It ranks the features that are common in both malware and normal class lower and ranks the features effective discriminators for a class higher. Although IG computation time is not negligible, the IG feature selection does not have to be in real time. The final selected features (shown by the shaded color) are trained to generate Naive Bayes model. Since the overlapping n-gram is used, the occurrence of these n-gram features is considered as independent.

3.2. Classification of Stateless Payload Using Naive Bayes with Snort Signatures. Our previous work in [13] has shown that Naive Bayes has a processing time and accuracy tradeoff compared to other classifiers, that is, J48 and SMO. Since the future aim is to deploy the malware detection as hardware implementation, Naive Bayes model is a good candidate to be applied to our proposed method in this work.

Using the generated Naive Bayes model, payloads at the stateful and stateless level are classified to either normal or attack classes (see Figure 1). Based on the Naive Bayes theorem, the a posteriori probability of a flow or packet d_i being in class c_j is

$$P\left(c_j \mid d_i\right) = \frac{P\left(c_j\right) P\left(d_i \mid c_j\right)}{\sum_{k=0}^{|\text{allclasses}|} P\left(c_k\right) P\left(d_i \mid c_k\right)}, \quad (1)$$

where $P(d_i \mid c_j)$ is the probability of d_i occurring in trained class c_j and it depends on the Naive Bayes model used [14].

$P(c_j)$ is the a priori probability of a flow or packet being in class c_j where $j \in \{0, 1\}$ for a 2-class classification.

From (1), we can write the a posteriori probability of a flow occurring in class c_j in the form of

$$P\left(c_j \mid x\right) = \frac{P\left(c_j\right) P\left(x \mid c_j\right)}{P\left(x\right)}, \tag{2}$$

where $x = \{x_1, x_2, \ldots, x_v\}$ is an array of selected features from a complete flow. $P(x)$ is the probability that x occurs in the training set. By assuming feature independence, the likelihood probability of x occurring in class, c_j, $P(x \mid c_j)$, is given by

$$P\left(x \mid c_j\right) = \prod_{i=1}^{v} P\left(x_i \mid c_j\right). \tag{3}$$

Basically, at this level, the detection is done only after all the transmitted packets have arrived at the destination host and have been reconstructed into the original flow.

At the stateless level, features are selected uniformly in each packet. A flow that consists of w packets can be approximated by w feature arrays as follows:

$$\begin{aligned} x &= \{x_1, x_2, \ldots, x_v\} \\ &\approx \{x_1, x_2, \ldots, x_w\}. \end{aligned} \tag{4}$$

Equation (4) represents w feature arrays, each from an individual packet. The feature array for a packet, x_i, is

$$x_i = \left\{x_{i,1}, x_{i,2}, \ldots, x_{i,|xi|}\right\}, \tag{5}$$

where $|x_i|$ is the number of features selected from a packet and used for classification. The likelihood probability of feature events x_i occurring in class, c_j, $P(x_i \mid c_j)$, is given by

$$P\left(x_i \mid c_j\right) = \prod_{k=1}^{|x_i|} P\left(x_{i,k} \mid c_j\right). \tag{6}$$

Inserting (6) into (2), we can calculate the probability score for each packet and classify the packet either as malware or normal. These classified packets that belong to a flow are reassembled to the original flow at the destination with the class of the flow being determined based on the packets classes. For example, a flow contains classified packets $p1$, $p2$, and $p3$. When at least one of these packets is an attack packet, then the flow is determined as an attack.

The decision making process that is done by the model for classifying a flow or packet is described in Algorithm 1.

In Algorithm 1, the classification process is divided into a few parts. The first part (lines 1 to 6) describes the extraction of important information such as features frequency and probability from the Naive Bayes model. The second part (lines 7 to 9) describes the extraction of n-gram features in each testing flow. The last part (lines 10 to 21) describes the overall classification steps including matching n-gram features between the model and the testing flow, determining the testing flow probability value, and determining the testing flow class.

```
(1) for all selected features do
(2) features, fre_norm, pb_norm, fre_att,
    pb_att => splitFlow
(3) push (feature) => field1
(4) push (pb_norm) => field3
(5) push (pb_att) => field5
(6) end for
(7) for each captured flow do
(8) extract payload => allFlow
(9) end for
(10) for allFlow rows do
(11) split row
(12) if test feature eachrow = field1 then
(13) accumulate differ field5, field3
(14) end if
(15) count Probability (Pb) using (2)
(16) if Pb > threshold then
(17) flow => malware
(18) else
(19) flow => normal
(20) end if
(21) end for
```

Algorithm 1: Flow/packet classification.

4. Experimental Design

4.1. Dataset Collection. Since this work uses 2-class machine learning classification, both normal and malware datasets are needed. To obtain a reliable generative model, malware and normal traffic should be taken from the same place at the same time to sample traffic with the same trend. Here, the traffic is captured from our campus network in the way as shown in Figure 5(a). This section explains the method used to capture the training and classification datasets. Tools such as Snort [7] and Tcpdump [15] are used in the capturing process. Snort is applied to sniff the malicious traffic that propagates in the academic network, whilst Tcpdump is executed to capture the background traffic to extract the nonmalware traffic.

Figure 5(b) shows the process to sample the malware and nonmalware traffic. When a packet matches Snort's signatures, Snort will produce an alert. This alert will trigger Tcpdump to store packets just before and after the time (e.g., t_i) of the alert. When no alert comes, no action will be taken until another alert is triggered at another time (e.g., t_{i+1}), in which the same action will be taken. By doing so, both traffic classes are collected at the same time and place.

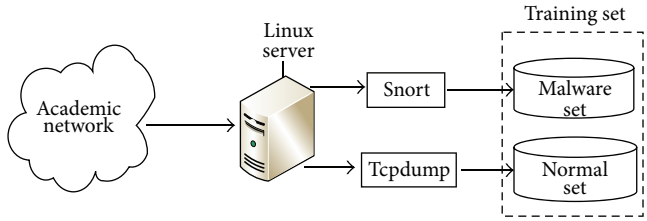
4.2. Evaluation Criteria. For the purpose of evaluating the results obtained from the classification, the following metrics are described in Table 1:

$n_{w \to w}$: number of correctly identified malware payload;

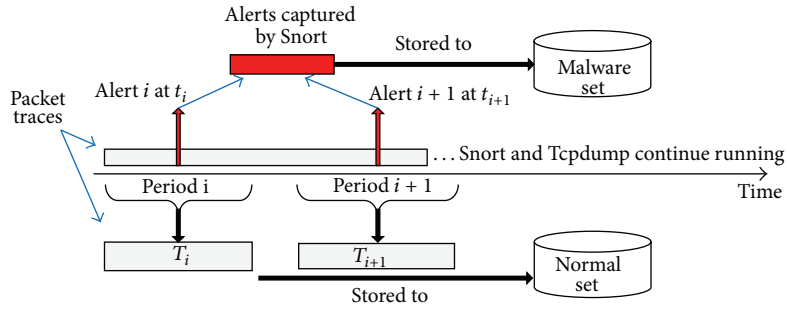
$n_{l \to w}$: number of wrongly identified normal payload;

$n_{l \to l}$: number of correctly identified normal payload;

$n_{w \to l}$: number of wrongly identified malware payload.



(a) Capturing traffic from campus network



(b) Sampling traffic

Figure 5: Dataset preparation.

Table 1: Verification criteria.

Criteria	Symbol	Expression
Recall	Rec	$\frac{n_{w\to w}}{n_{w\to w} + n_{w\to l}}$
False positive	FP	$\frac{n_{l\to w}}{n_{l\to w} + n_{l\to l}}$
False negative	FN	$\frac{n_{w\to l}}{n_{w\to w} + n_{w\to l}}$
Precision	Pre	$\frac{n_{w\to w}}{n_{l\to w} + n_{w\to w}}$

Since the focus of this work is to detect malware at the stateless level, recall and false negative are the most important measurements from the classification. Security aspect is concerned to has higher priority and should not be compromised. False positive and precision reflect the number of normal traffic classified as malware. These measurements also play an important role in our verification since we need to reduce the number of normal packets that were wrongly classified as attacks. End users may feel unhappy when their valid traffic (packets), wrongly classified as malware, is blocked by the security devices although they browsed a legitimate website without any malware threats. Comparable ratio of these measurements between the stateless and stateful levels can show the feasibility of detection inside network. The simulation also produces log file that contains IP addresses and the classification class result of classified traffic.

4.3. Experimental Setup. Since detection accuracy was the main focus and not the actual detection time, all experiments in this paper were done offline using existing machine learning algorithm (Naive Bayes) by improving the method of selecting features (using Snort signatures). This work is to prove the concept that assisted machine learning can be applied to the stateless malware detection. Detection at the stateful level is used as the benchmark. Figure 6 shows the entire malware detection method proposed in this paper. Payloads from the previous time (from offline traffic for slot t_i) are trained to generate model. The model is used to classify incoming payloads (from offline traffic for slot t_{i+1}) into their classes.

Table 2: Number of training and testing set used in the experiment.

Dataset	T1	T2	
	Stateful	Stateful	Stateless
Malware	693 flows	336 flows	549 packets
Normal	26798 flows	16965 flows	34762 packets

Table 2 presents the datasets used to analyze the detection at the stateful and stateless level. Based on Figure 6, T1 is used as the training set to generate Naive Bayes model with the assistance of Snort signatures. T1 remains in the form of complete flows (reconstructed flows), since the reliability of the generated model is considered. T2 is used as the testing set which consists of two forms: complete flows and corresponding chunk of packets. Let us assume that L3 is the stateless level whilst L4 is the stateful level. The evaluation begins by reviewing the payloads in the classification process for both levels using Naive Bayes model. The differences in accuracy when detecting partial payload at L3 and complete payload at L4 are observed.

5. Results and Discussion

5.1. Stateful versus Stateless Classifications. The simulation produces the false rate, precision, recall, and a log file that

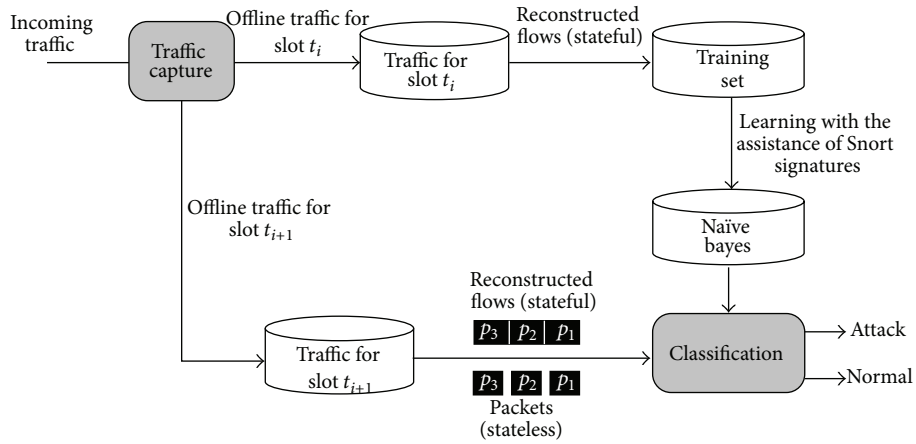


Figure 6: The proposed malware detection system.

Table 3: Detection accuracy for the stateless level versus stateful level.

Level	FN	FP	Pre	Rec
Stateless	0.042	0.091	0.173	0.958
Stateful	0.042	0.070	0.213	0.958

contains the list of traffic flows/packets with information format as addresses><indexnumber><classtype><classresult>. To obtain comparable result of detection for L3 and L4, the measurement of false negative (FN), false positive (FP), precision, and recall is done based on the number of flows and not the number of packets. Therefore, for L3, the classified packets must be reassembled to the original flow before calculating those values.

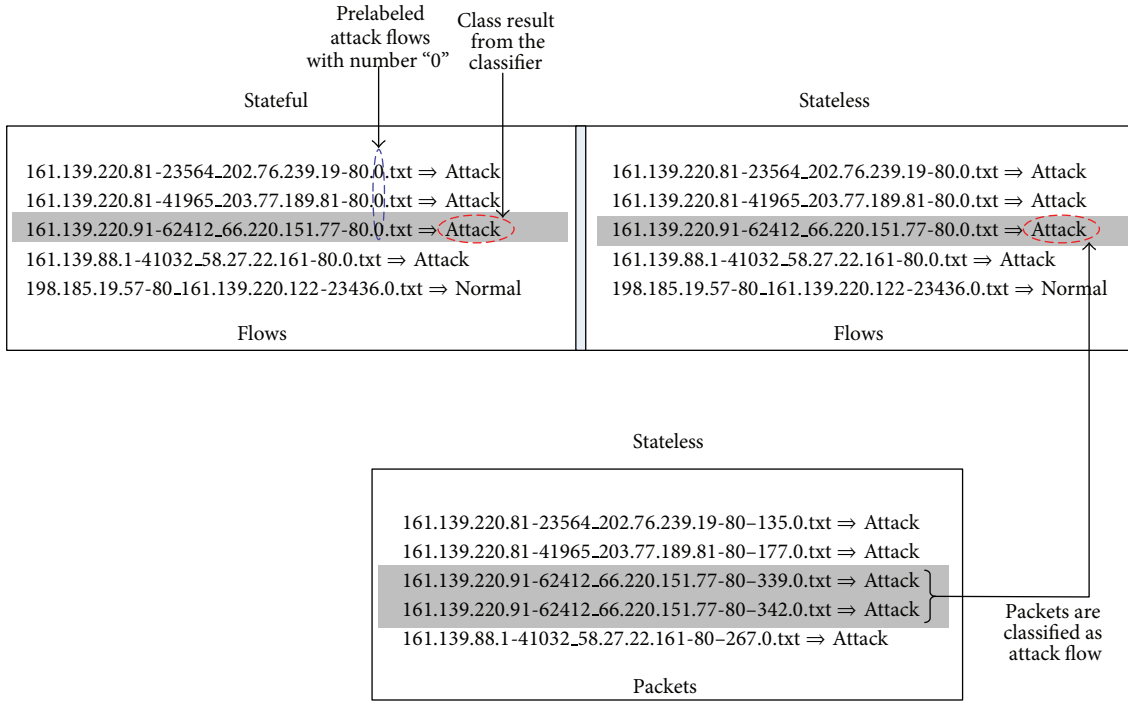
Table 3 presents the classification result for the detection at L3 and L4. Overall, the detection accuracy is found similar (accuracy difference of around 2%) at L3 and L4. FP is higher for L3, whilst FN does not change for both abstraction levels. The results show that normal traffic is misclassified more frequently at L3.

5.2. Analysis of Malware Trojan Packets and Normal HTTP Packets. This section describes and explains the results from the classification between L4 and L3. Figures 7(a) and 7(b) show the L3 and L4 malware detection results. The IP and port addresses of malware and normal traffic are listed with their classes. Verification is done with labelled traffic. For L3, the classified packets are reassembled to the original flows before determining whether the flows are malware or normal. If one or more packets that belong to the flow are correctly classified as malware, then the flow will be defined as malware flow. This is based on the signature based method when most antivirus programs only look for a partial match in files when scanning [16].

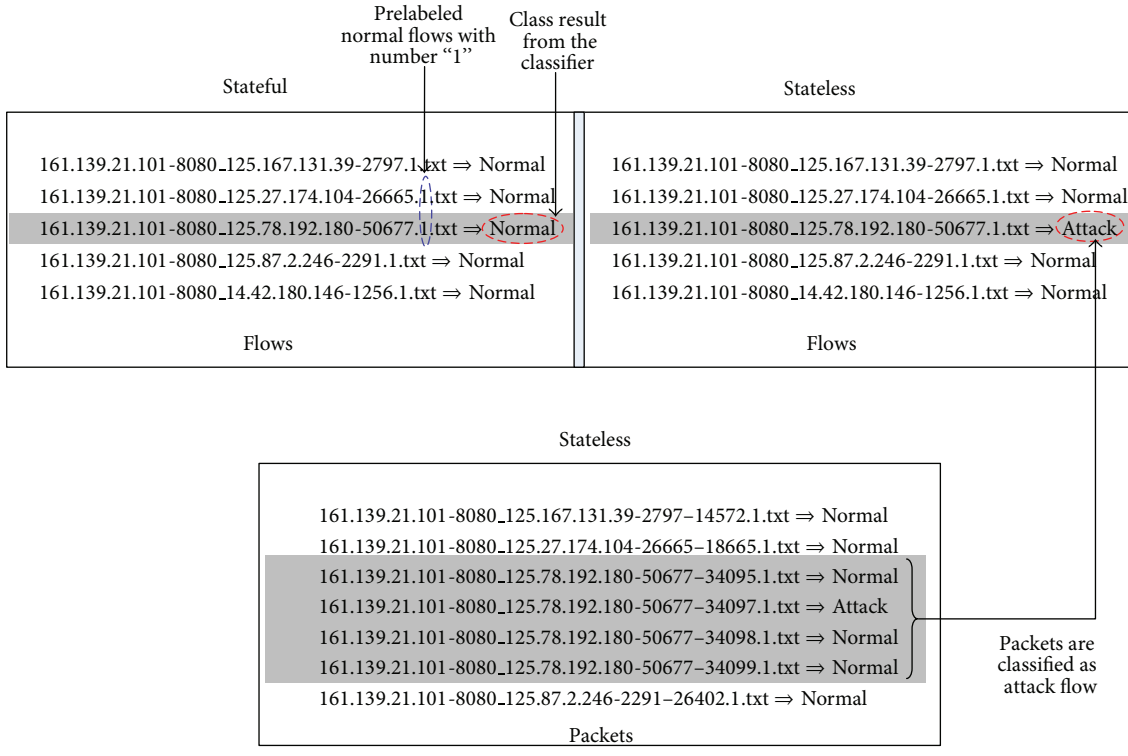
For example, a Trojan flow with source address of 161:139:220:91_62412 and destination address of 66:220:151:77_80 contains 2 packets as shown in Figure 7(a). Both of the packets are correctly classified as malware and, thus, the flow is classified as malware. Otherwise, if all the packets that belong to the flow are misclassified as normal, then the flow will be misclassified as normal. Similar approach is used to determine the class for normal traffic. If one of the packets is misclassified as malware packet, then the flow is misclassified as malware flow. For example, normal HTTP flow with source address of 161:139:21:101_8080 and destination address of 125:78:192:180_50677 contains 4 packets as shown in Figure 7(b). Three packets are correctly classified as normal except one packet from the flow is misclassified as malware and, thus, the flow is misclassified as a malware flow. Afterwards, the class of flows at L3 is compared with the class of the same flows at L4.

Based on the available dataset, it is observed that normal traffic contains longer payload (e.g., 10 Kb) compared to malware traffic (1 Kb). Therefore, within the maximum transmission unit (MTU) for the Ethernet, that is, 1.5 Kb [17], the normal payload is distributed and carried by a few packets, whilst the malware payload is carried by a single packet. For L3, each packet is detected based on partial payload compared to the complete payload for L4. The misclassification of the normal packets is caused by the lack of information that can be observed in the partial payload which contributes to the lower detection accuracy for L3 as compared for L4. Nevertheless, for the malware packets, the detection result remains similar for L3 and L4 since the malware payload is typically shorter (e.g., Slammer worm) to ensure faster time to spread [18]. Thus, the malware payload remains in a single packet whether for L3 or L4.

Based on this finding, there is a potential to apply content based detection on middlebox. The use of Snort signatures n-gram helped the generative model to immediately and accurately detect malware at the packet level without the need for the two-path processing, as done by Varghese et al. [1]. The detected malware packets can be dropped earlier while in transit. Therefore, packet buffering and packet reassembling are not required. This technique can reduce the overall detection time. Thus, detection at the stateless level using machine learning technique seems feasible to be implemented as a way to control the malware in the network.



(a) Determine class of malware payloads at the stateful and stateless level



(b) Determine class of normal HTTP payloads at the stateful and stateless level

Figure 7: Determination of class for payload at stateful and stateless level.

6. Conclusion

Malware detection is observed best at the byte streams level mainly due to the limited payload information seen at the packet level. However, this will not allow malware detection in transit. The finding of the work has proven that detection accuracy at the packet level is comparable to the full byte stream level. This paper demonstrated that malware detection at the stateless level (partial payload) is feasible to be implemented. Malware detection at the stateless level promises better malware detection, possibly on security middleboxes inside the network since malware detection at the lower abstraction can relax the stateful implementation restriction. Moreover, stateless detection can further increase the effectiveness of malware control such as increasing detection speed (e.g., detect malware near to the sources such as routers and gateways). Preventing the spread of malware at the early stage can prevent outbreaks at the end systems.

Since the proposed approach in this paper is still at the exploratory stage, the testing was done offline and not in real time. In the future, the viability of the approach on a real network especially for a fast speed network can be assessed. We also plan to further investigate malware detection at the packet level and how it performs under incomplete features, fragmentation, insertion, and evasion attacks.

Conflict of Interests

The content of this paper reflects only the opinion of the authors with independence of their affiliations. The authors do not have a direct financial relation with the commercial entities mentioned in this paper.

References

[1] G. Varghese, J. A. Fingerhut, and F. Bonomi, "Detecting evasion attacks at high speeds without reassembly," in *Proceedings of the SIGCOMM Conference*, pp. 327–338, Pisa, Italy, 2006.

[2] E. P. Markatos, "Speeding up TCP/IP: faster processors are not enough," in *Proceedings of the 21st IEEE International Performance, Computing, and Communications Conference (IPCCC '02)*, pp. 341–345, Phoenix, Ariz, USA, April 2002.

[3] P. Inella, *An Introduction to Intrusion IDS*, 2001, http://www.securityfocus.com/.

[4] N. Desai, *Intrusion Prevention Systems: the Next Step in the Evolution of IDS*, 2003, http://www.securityfocus.com/.

[5] J. Zico Kolter and M. A. Maloof, "Learning to detect and classify malicious executables in the wild," *Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2006.

[6] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, and Y. Elovici, "Unknown malcode detection via text categorization and the imbalance problem," in *Proceedings of the IEEE International Conference on Intelligence and Security Informatics*, pp. 156–161, Taiwan, June 2008.

[7] M. Roesch, *Snort*, 2001, http://www.snort.org/.

[8] T. H. Ptacek and T. N. Newsham, "Insertion, evasion, and denial of service: eluding network intrusion detection," Tech. Rep. T2R-0Y6, Calgary, Canada, 1998.

[9] M. Z. Shafiq, S. A. Khayam, and M. Farooq, "Improving accuracy of immune-inspired malware detectors by using intelligent features," in *Proceedings of the 10th Annual Genetic and Evolutionary Computation Conference (GECCO '08)*, pp. 119–126, Atlanta, Ga, usa, July 2008.

[10] C. Sarkar, *Connection Establishment in TCP Three Way Handshaking*, M. Tech—I, CSE IIT Bombay, 2009.

[11] T. Abou-Assaleh, N. Cercone, V. Keselj, and R. Sweidan, "Detection of new malicious code using N-grams signatures," in *Proceedings of the 2nd Annual Conference on Privacy, Security and Trust*, pp. 193–196, Fredericton, NB, Canada.

[12] Y. Yang and J. A. Pedersen, "Comparative study on feature selection in text categorization," in *Proceedings of the 14th International Conference on Machine Learning*, pp. 412–420.

[13] I. Ismail, M. N. Marsono, and S. M. Nor, "Detecting worms using data mining techniques : learning in the presence of class noise," in *Proceedings of the 6th International Conference on Signal Image Technology and Internet Based Systems (SITIS '10)*, pp. 187–194, Kuala Lumpur, Malaysia, December 2010.

[14] A. McCalum and K. A. Nigam, "Comparison of event models for naive bayes text classification," in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI '98)*, pp. 41–48, Madison, Wis, USA, 1998.

[15] L. M. Garcia, *Tcpdump and Libpcap*, 2010, http://www.tcpdump.org/.

[16] L. Zeltser, "Understanding Anti-Virus Software," The Monthly Security Awareness Newsletter for Computer Users, The SANS Institute, 2011.

[17] P. Simonea, "The OSI Model: understanding the seven layers of computer networks," Expert Reference Series of White Papers, Global Knowledge, 2006.

[18] C. Fosnock, "Computer worms: past, present and future," CISSP, MCSE, CNE East Carolina University, 2005.