Turk J Elec Eng & Comp Sci, Vol.17, No.2, 2009, © TÜBİTAK
doi:10.3906/elk-0905-29

Polymorphic worm detection using strong token-pair signatures

Burak BAYOĞLU[1], **İbrahim SOĞUKPINAR**[2]
[1]*National Research Institute of Electronics and Cryptology*
Department of Information Systems Security
TÜBİTAK UEKAE, P.O. Box 74, 41470, Gebze, Kocaeli-TURKEY
e-mail: bayoglu@uekae.tubitak.gov.tr
[2]*Gebze Institute of Technology, Department of Computer Engineering*
P.O. Box 141, 41400, Gebze, Kocaeli-TURKEY
e-mail: ispinar@bilmuh.gyte.edu.tr

Abstract

Malicious software has become a big threat to information systems, which are widely used to store, transfer and process information for many critical assets. Worms are one of the most harmful network-enabled malicious software that can threaten networks and applications. Two main characteristics of worms distinguish them from the well-known virus programs and as a result are much more dangerous than the virus programs. First, they do not need to attach themselves to an existing program. Second, worms do not require end-user interaction to realize the intended attack. Therefore, a large number of victims can be infected in a short time. Polymorphic worms are a special subset of worm family which are more difficult to detect. Polymorphism is the key that facilitates creating different looking polymorphic worm copies while keeping the original worm code intact. Each variant for a polymorphic worm has a different pattern that it is not effective to use simple signature matching techniques. In this work, Strong Token-Pair(STP) signature scheme has been proposed to detect polymorphic worms. Experimental results support that STP signatures can be used with low false negative and false positive rates.

Key Words: *Polymorphic worm, worm detection.*

1. Introduction

Nowadays, while Information Systems (IS) are widely used to store, transfer and process information for many critical assets, malicious software has become a large threat for IS. There are many types of threats against IS whereas we focus on the polymorphic worm threat as a self-replicating and fast-spreading malicious software. Worms exploit vulnerabilities of IS to infect them. Depending on the nature of the vulnerability, it is commonly possible to remotely execute unauthorized commands on systems. Viruses need to attach themselves to a carrier program and generally need users to execute this program to be activated. Unlike viruses, worms exploit operating system and application software vulnerabilities to infect the systems and use network infrastructure to infect other systems. This self-propagation brings the ability of quickly spreading to very large networks.

Polymorphic worms are a special subset of the worm family which are more difficult to detect when compared to regular worms. The idea behind polymorphic worms is that, by using polymorphism techniques, one can create a polymorphic worm code which has a different pattern for each copy of the same worm. This polymorphism makes it ineffective to use simple signature matching techniques [1] for detecting polymorphic worms. Therefore more flexible signature schemes need to be developed based on common information that exist in each different polymorphic worm variant. And because of its nature, malicious code authors are highly motivated to generate polymorphic worm codes which are more sophisticated and hard to detect. With this in mind, information security risks that arise from this challenging threat needs to be mitigated by information systems security researchers.

In this work, Strong Token-Pair(STP) signature scheme has been proposed for detecting polymorphic worms through content analysis of network flows. The proposed solution defines STP Conjunction and STP Subsequence signature classes and a new mathematical model to be used in the flow evaluation process. Experiments have been performed for two polymorphic worms exploiting Apache-Knacker [2] and BIND-TSIG [3] vulnerabilities. Experimental results support that STP signatures can be used with low false positive and low false negative rates. The proposed flow evaluation method successfully represents the flow probabilities in flow pools. In addition to successful classification of suspicious and innocuous traffic, a good signature based polymorphic detection technique must be resilient to changes in the worm structure. STP signature scheme is built on a complex Bayesian model which is flexible and resilient to the slight changes in the worm structure. STP signatures can also be used to improve other signature-based polymorphic worm detection techniques by providing signature flexibility while not degrading the false positive and false negative performance.

The remainder of this paper is organized as follows. In Section 2, background information and related works on polymorphic worm detection are discussed. STP signature scheme is presented in Section 3. Experimental results and analysis are given in Section 4. The paper is concluded in Section 5.

2. Background information and related works

Polymorphic code is generally achieved by utilizing encryption techniques. It is obvious that encrypting the whole code is not feasible because that would result with a meaningless code which is impossible to run on victim systems. Thus there is still some unencrypted code that must be common for each polymorphic worm variant. The unencrypted parts can be categorized as the exploit code and decryption routine. The exploit code is that portion exploits the known vulnerability on the victim system and branch the execution cycle to the decryption routine. However, there exists techniques to change decryption routine for each polymorphic worm variant discussed later in this section. The decryption routine decrypts the polymorphic worm main code and passes the execution cycle to it for performing its injurious activity and producing new worm variants to infect other systems afterwards.

Exploit code is necessary for exploiting the vulnerability and activating the decryption routine which decrypts the encrypted worm code using the decryption key. Randomly selected decryption keys are used at each iteration of a new worm variant. According to this definition, randomly selected decryption key and encrypted worm code have different patterns for each worm variant where decryption routine and exploit code remains unchanged. Different decryption routines can be produced for each polymorphic worm variant using obfuscation techniques [1]. There exist readily available polymorphic code engines [4, 5] to prepare a polymorphic code packet in which the decryption key and the obfuscated decryption routine are embedded together with the encrypted worm code. All of these polymorphic worm parts are encapsulated in a network protocol frame which has its own structure for being accepted as a regular network packet to be processed by the victim systems [Figure 1]. Being prepared for the worst case, it is assumed that the decryption routine is perfectly obfuscated and completely different for each worm variant. Thus the exploit code and network protocol frame structure are the only invariant parts on which to base our signature mechanism proposal, as also done in [1] and [6].

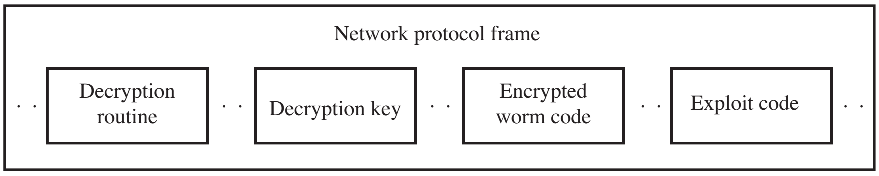


Figure 1. Polymorphic worm structure.

Honeycomb [7], Autograph [8], and EarlyBird [9] generate automatic worm signatures but all of these worm detection efforts share a common assumption that the worm content has a large enough contiguous string that identifies the worm uniquely. Since a common substring would be insufficient to detect polymorphic worms, these techniques are not appropriate for polymorphic worm detection.

Polymorphic worm detection efforts can be classified in two categories as content based approaches and behavior based approaches. Content based polymorphic worm detection techniques use the worm content to produce information that matches the worm [1, 6, 10, 11, 12, 13, 14, 15, 16]. Behavior-based approaches focus on the behavior of the worm in terms of network and system activities by watching for the abnormalities that differ from normal activity rather than inspecting the worm content [17, 18, 19, 20]. Worm detection can be performed network- or host-based. Network based techniques are deployed at the network layer before the malicious code arrives to the end systems [1, 6, 13, 14, 15, 16, 18, 20]. Host based polymorphic worm detection techniques are deployed at the end systems [10, 11, 12, 17, 19].

Content based intrusion detection systems usually check for defined attack signatures to detect the threats. One common substring to define a polymorphic worm would be insufficient to detect polymorphic worms [1, 6]. More specific worm signatures must be produced which exploit as many properties of the worm content as possible and these distinct properties must be correlated to successfully detect the polymorphic worms.

Polygraph [1] is a content based polymorphic worm detection technique. Polygraph proposes three signature classes. These are conjunction signatures, token-subsequence signatures and Bayes signatures. Substrings of the worm content which are used for signature generation are called tokens. Conjunction signatures consist of a set of tokens that match the worm if and only if all of the tokens are present in the flow but in any order. Token-subsequence signatures are similar to conjunction signatures, the only difference is that the tokens must appear in a given order. Bayes signatures consist of a set of tokens. Each token is associated with a score which is based on the probability of that token to appear in the flow pools. An overall threshold value specific to a worm is specified to be used in the decision phase. Unlike the conjunction signatures and token-subsequence signatures, Bayes signatures provide probabilistic matching information. Given a suspicious flow, a flow score is calculated which is sum of the tokens' score that appear in the flow. If it results a flow score over the pre-defined threshold, Polygraph classifies the given flow to be a polymorphic worm.

Hamsa [6] outperforms Polygraph [1] in terms of efficiency, accuracy, and attack resilience. Hamsa is a network-based automated signature generation system for polymorphic worms. Hamsa generates content-based signatures. They use the protocol frame part, exploit data, and worm content to generate the signatures. Given a suspicious flow, token extraction is performed by using a suffix array based algorithm [21] to find all byte sequences that occur in at least λ fraction of the suspicious flow pool. Hamsa defines a worm signature which is constrained to include a multi set of tokens in any order in each of the suspicious flows. The first token to appear in the worm signature is selected as the one with the smallest false positive value. The token which has the least false positive value in conjunction with the first token is selected as the second token. The rest of the tokens are evaluated similarly and the final worm signature is produced. Appropriately choosing the values for the model proposed, this greedy approach is claimed to find a good signature with the assumption that the exploit data and the protocol frame part is not under the control of the worm author. This assumption is realistic since the worm author has to exploit a software vulnerability over a static protocol structure to finally execute the worm content.

Token extraction is the preliminary process before generating the content-based polymorphic worm signatures. Polygraph includes a token if it is not a substring of another token, if so, its unique coverage must be above a specified threshold. Although a token is substring of another token and its unique coverage is less then the threshold, it may still have large coverage among the worm samples as a substring of other tokens. Hamsa treats every substring with coverage over the threshold as a token. Another concern about extracting tokens is that Polygraph performs a bottom up traversal of the suffix tree to calculate the number of occurrences of a token candidate and continuously generates the tokens via a top down traversal of the suffix tree. This operation takes asymptotically linear time but even improved suffix tree implementations consume large memory space. Each input character requires 20 bytes in the worst case [6]. Using the deepsort algorithm [21] implementation on suffix tree, Hamsa gets around 100 times speedup for token extraction when compared to Polygraph. Even if the same suffix tree based token extraction and the suffix array based false positive calculation techniques are used, Hamsa was found to be 64 to 361 times faster than Polygraph as a result of signature generation technique difference.

Polygraph [1] uses hierarchical clustering to generate worm signatures from a noisy suspicious flow pool. Initially each worm sample is considered as a cluster itself and clusters whose union give a signature with the least false positive rate are merged. This merging activity continues until there is only one cluster, or additional merging activity does not produce a cluster with least false positive rate. As stated in [6], this approach would tend to cluster a worm sample with a normal sample if the variant part of the worm samples share common tokens in normal pool. This will yield false negatives. Model-based greedy signature generation algorithm of Hamsa successfully bounds false positive and false negative rates of the signature even if the attacker has full control of including any content in the variant part of polymorphic worm samples. Hamsa [6] defines a new attack type named token-fit attack, which is a stronger form of the coincidental-pattern attack defined in Polygraph [1]. It is based on the idea that the attacker may obtain normal traffic with a similar token distribution as the normal noise in the suspicious pool. This way the worm author may inject the normal traffic tokens into worm variants. This would make the worm samples look more like the normal traffic noise in the suspicious pool, thus degrading the quality of the worm signature. According to the tests performed in [6], Polygraph cannot detect such worms (100% false negative) while Hamsa does.

PADS [11, 12] is a content based polymorphic worm detection technique which claims to fill the gap between traditional signature based schemes and anomaly-based intrusion detection schemes. Two algorithms were proposed which are based on Expectation-Maximization [22] and Gibbs Sampling [23] to generate position aware distribution signatures from the polymorphic worm samples. Hamsa and Polygraph both focus on the invariant parts of the worm content to generate the worm signatures. In addition to this invariant content, PADS also considers the variant parts of the worm content that follow certain distributions.

LISABETH [13] is an automated content-based signature generator for zero-day polymorphic worms. The authors claim that LISABETH is an improved version of Hamsa [6] in terms of resilience to flow pool poisoning attacks and signature generation performance.

Polygraph [1] and Hamsa [6] can not produce precise signatures for a special class of vulnerabilities, named feature omission vulnerabilities, as described in [14]. It is stated that both Polygraph [1] and Hamsa [6] rely on the assumption that one or more tokens need to be found as part of the exploit code which is not common in the innocuous flow pool. A stack-based buffer overflow vulnerability defined at [24] is used to generate a sample exploit for omission vulnerabilities which can not be detected (100% false negatives) by Hamsa [6] and Polygraph's token subsequence and token conjunction signatures. It is important to note that Polygraph's Bayes signatures are more resilient to such exploits. In addition, our signature scheme is also based on a complex Bayesian model similar to Polygraph's naive Bayes signature model.

Token-Pair Conjunction and Token-Pair Subsequence signatures have been proposed in [15]. Token-Pair signatures is a content based polymorphic worm detection technique. Invariant parts of the polymorphic worm were used to extract the common tokens for each polymorphic worm similar to Polygraph [1]. A score is calculated for token pairs based on their probability of appearing in the flow pools. An overall score for a flow is calculated depending on the token pairs appearing in the flow and this score is compared to a specified threshold value in the decision phase. The flow score calculation method has a deficiency in the process of specifying the token pairs which will contribute to the flow score. Some token pairs have high probability of appearing in the suspicious flow pool and low probability of appearing in the innocuous flow pool resulting with a high score associated with them. On the other hand, other token pairs have low token pair scores associated with them. Those token pairs which have relatively high scores are valuable for polymorphic worm detection efforts. However, the flow score calculation method sums up adjacent token pair's score regardless of considering their comparative contribution to the final score. This situation may result in ignoring valuable token pairs, i.e. a high scored pair T_{13} is ignored in a sequence $T_1 .^* T_2 .^* T_3$, since the proposed flow calculation method considers low scored pairs T_{12} and T_{23} instead.

3. STP signatures

STP scheme generates signatures for detecting polymorphic worms through content analysis of network packets. In order to generate STP signatures, token pairs need to be generated first. A token is the atomic substring which is used to generate the token pairs. The algorithm defined in [25] is utilized to extract tokens of a minimum length α that occur in at least K out of n worm samples in the suspicious flow pool as did in Polygraph [1]. After extracting the tokens, possible token pairs are generated and a token-pair score is calculated for each token pair. STP signatures consists of token pairs and the scores associated with them. Using the signature set generated, network flows are evaluated to decide whether they are polymorphic worm or not. STP signature scheme is an improved version of the Token-Pair signature scheme proposed in [15] in terms of flow evaluation method used. Token-Pair Signatures [15] sums up each adjacent token pair's score to calculate the overall score for a given flow. As already mentioned in Section 2, this may result in ignoring token pairs with high scores which have critical importance for successful detection. Therefore, the strong token pair concept has been introduced in STP signature scheme in order to specify the best subset of token pairs in the flow, which will be considered during the flow evaluation phase. Two signature schemes have been proposed in this work. These are STP Subsequence and STP Conjunction signatures which are presented as follows.

Network flows are treated as a sequence of tokens. Flows that are subject to evaluation and being labeled either polymorphic worm or innocuous traffic, flow samples in the innocuous and suspicious flow pools which are used for signature generation are all modeled the same way. The payload information between the discovered tokens in a flow have no use since this information is not used to evaluate the given flow. Assuming that n tokens have been discovered in a flow, the flow is represented as given in Figure 2. T_i (Token i) is the i^{th} token discovered in the flow. Similarly, T_j (Token j) is the j^{th} token discovered in the flow where $j > i$. Token pair ij (TP_{ij}) stands for the pair of tokens which includes T_i as the first token and T_j as the second (last) token of it. Tokens that exist in a token pair do not necessarily be adjacent tokens. The order of tokens in a token pair is considered in STP Subsequence signatures where the appearance order of a token in a token pair has no impact on the STP Conjunction signatures.

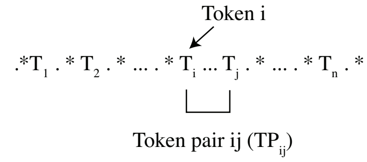


Figure 2. Network flow structure.

STP Conjunction signatures contain a set of token pairs with their pre-calculated scores. The token pairs that constitute a conjunction signature do not have an order rule. No order rule means that, regardless of the sequence of the tokens, the related pair score is added to the total score if both tokens in the token pair appear in the flow. TPC_{ij} notation is used to represent conjunction of T_i and T_j. It is important to note that strong token pairs have higher priority than the other token pairs which have lower scores. Therefore the token pair conjunctions with higher score contributes to the final flow score where the token pair conjunctions between the first and the last token of the strong token pair are ignored. A threshold value specific to a polymorphic worm in the suspicious flow pool is specified to compare with the total score. If the total score for all the token pair conjunctions detected in the flow is higher than the threshold value, the flow is labeled as a polymorphic worm.

STP Subsequence signatures also contain a set of token pairs with their pre-calculated scores. The token pairs that constitute STP Subsequence signatures, have an order rule for the pair. Token pair score is calculated for the case where the second token of a pair appears after the first token of the pair. TPS_{ij} notation is used to represent a subsequence of flow where T_j appears after T_i and $j > i$. The tokens are ordered internally but there is no restriction about which token pair to appear next. Therefore STP Subsequence signature scheme still considers TPS_{ij} appearing after TPS_{km} where $m > k > j > i$. Similar to the STP Conjunction signatures, strong token pairs in the flow have higher priority than the weak token pairs. A threshold value specific to a polymorphic worm in the suspicious flow pool is specified to compare with the total flow score. If the total score for all the token-pair subsequences detected in the flow is higher than the threshold value, the flow is labeled as a polymorphic worm.

The notation used in the remainder of the paper is given in Section 3.1. STP signature generation process is described in Section 3.2. An overview of the STP scheme is also depicted in Figure 3. Flow evaluation method and the mathematical model for flow score calculation is given in Section 3.3 and Section 3.4, respectively. Decision rule, the final phase of the process, is given in Section 3.5.

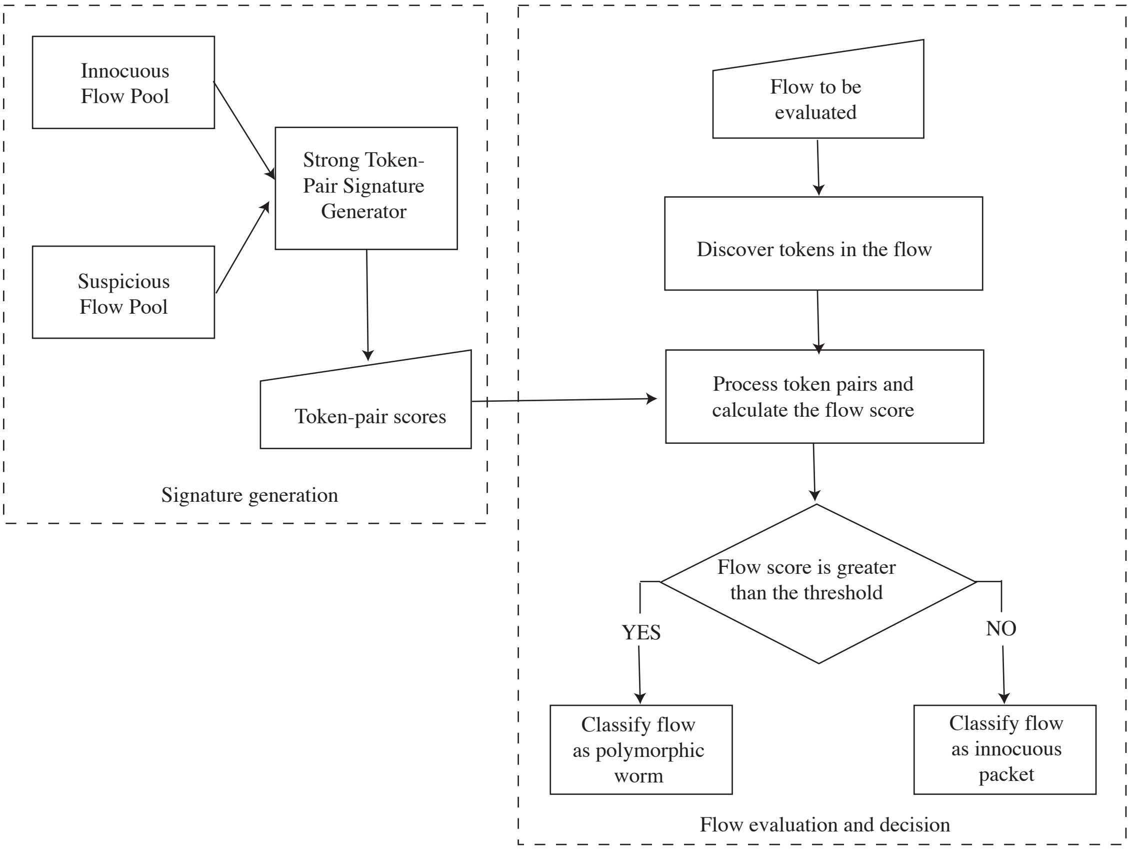

Innocuous Flow Pool
Suspicious Flow Pool
Strong Token-Pair Signature Generator
Token-pair scores
Signature generation
Flow to be evaluated
Discover tokens in the flow
Process token pairs and calculate the flow score
Flow score is greater than the threshold
YES
NO
Classify flow as polymorphic worm
Classify flow as innocuous packet
Flow evaluation and decision

Figure 3. STP signature scheme overview.

3.1. Notation

The notation used in the remainder of this paper is as follows.

x	: Flow to be classified.
$\{T_i\}_{1\leq i\leq n}$	: Token set of n to define a signature specific to a polymorphic worm.
$L(x)$	: Worm label for the flow as {true, false}. ($L(x) =$ true means flow x contains worm)
TPC_{ij}	: Token-pair conjunction where i^{th} token T_i and j^{th} token T_j are both present in the flow (regardless of order).
TPS_{ij}	: Token-pair subsequence where i^{th} token T_i is present in the flow and, j^{th} token T_j is present in the flow following T_i.
TP_{ij}	: Token-pair that consists of i^{th} token T_i and j^{th} token T_j (either in conjunction or subsequence).
$S(x)$	: Overall Bayes score for the flow.
E	: Threshold value specific to a polymorphic worm.

3.2. Signature generation

Two basic flow pools, namely the suspicious flow pool and innocuous flow pool, are used to generate token-pair signatures. The structure of flow pools is given in Figure 4. The suspicious flow pool contains variants of a polymorphic worm (Figure 4(a)). The innocuous flow pool contains worm-free innocuous traffic packets (Figure 4(b)). The network packets in the innocuous flow pool may or may not contain tokens which are discovered using suspicious flow pool. Two different signature schemes have been proposed; these are STP Conjunction signatures and STP Subsequence signatures. Both signature schemes use token pairs as the atomic signature parts. The token pairs need to be ordered in an expected way in case of using STP Subsequence signatures. STP Conjunction signatures require that both of the tokens appear in the flow regardless of any order rule. A token in a given flow can contribute to the total score once as the first token of a pair and once as the last token of a pair. That is required for avoiding unnecessary and excessive calculations in the flow score calculation method which is detailed in Section 3.4.

Token extraction must be performed first before generating the token pair signatures. Tokens are extracted for each worm in the suspicious flow pool based on the algorithm defined in [25] as explained in [1]. This process is performed in linear time in the total length of the worm samples [25]. Having extracted the tokens, flow pools can be represented in terms of tokens with the rest of information ignored as depicted in Figure 4(a) and Figure 4(b).

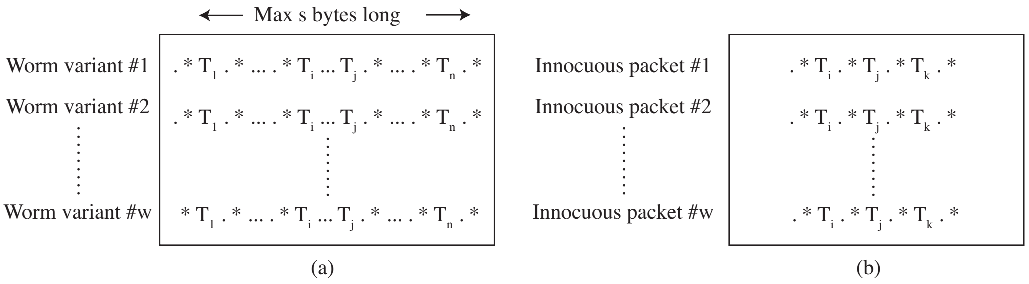
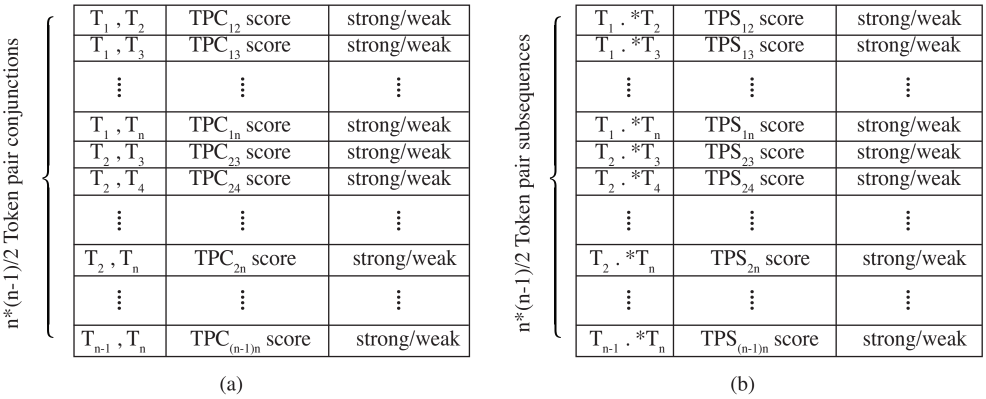


Figure 4. (a) Suspicious flow pool structure. (b) Innocuous flow pool structure.

General structure of STP Conjunction signatures is given in Figure 5(a). There are $C(n,2) = n!/2*(n-2)! = (n*(n-1))/2$ possible token pair combinations for a set of n tokens. These combinations are used for defining STP Conjunction signatures. There are two tokens in a token pair conjunction which are symbolized as T_i and T_j, and the corresponding token pair conjunction is symbolized as TPC_{ij}, where $1 \leq i, j \leq n$. For each TPC_{ij} a score is calculated which is based on the probability of i^{th} token T_i and j^{th} token T_j to appear in flow pools together regardless of an order rule. If the suspicious pool consists of w samples max s bytes long, the running time of token extraction algorithm is $O(s)$ [25]. Therefore token extraction algorithm runs in linear time with complexity $O(s)$, where s is a constant value. After that, $(n*(n-1))/2$ token pair conjunctions are searched for in the network packets of flow pools in linear time also. The ratio of the flows in the flow pools that contain the related token pair conjunction will designate the token pair conjunction score. This operation is repeated $(n*(n-1))/2$ times so the running time for calculating the token pair conjunction scores is $O(n^2)$. Both STP Conjunction and STP Subsequence signature set also have a “strong” or “weak” label for each token pair, as shown in Figure 5(a) and Figure 5(b). Strong and weak token pair concept is explained later in this section.



n*(n-1)/2 Token pair conjunctions

T_1, T_2	TPC_{12} score	strong/weak
T_1, T_3	TPC_{13} score	strong/weak
⋮	⋮	⋮
T_1, T_n	TPC_{1n} score	strong/weak
T_2, T_3	TPC_{23} score	strong/weak
T_2, T_4	TPC_{24} score	strong/weak
⋮	⋮	⋮
T_2, T_n	TPC_{2n} score	strong/weak
⋮	⋮	⋮
T_{n-1}, T_n	$TPC_{(n-1)n}$ score	strong/weak

(a)

n*(n-1)/2 Token pair subsequences

$T_1 . *T_2$	TPS_{12} score	strong/weak
$T_1 . *T_3$	TPS_{13} score	strong/weak
⋮	⋮	⋮
$T_1 . *T_n$	TPS_{1n} score	strong/weak
$T_2 . *T_3$	TPS_{23} score	strong/weak
$T_2 . *T_4$	TPS_{24} score	strong/weak
⋮	⋮	⋮
$T_2 . *T_n$	TPS_{2n} score	strong/weak
⋮	⋮	⋮
$T_{n-1} . *T_n$	$TPS_{(n-1)n}$ score	strong/weak

(b)

Figure 5. (a) STP Conjunction signature structure. (b) STP Subsequence signature structure.

There are $P(n,2) = n!/(n-2)! = n*(n-1)$ possible token pair subsequences for a set of n tokens. After extracting the tokens in linear time, the token order rule is also discovered which is common for all the worms in the suspicious flow pool. Assume that there exist n tokens extracted for a polymorphic worm, it is guaranteed that all of the worm variants in the suspicious flow pool are in the form $.*T_1.*T_2.*....*T_n.*$. For that reason, it is not necessary to calculate a score for TPS_{ij}, where $i > j$. Therefore half of the permutations is eliminated and finally we have $(n*(n-1))/2$ token-pair subsequences similar to the token-pair conjunctions. These permutations are used for defining STP Subsequence signatures. The structure of STP Subsequence signatures is given in Figure 5(b). The tokens in a token subsequence are symbolized as T_i and T_j, and the corresponding token pair subsequence is symbolized as TPS_{ij} where $1 \leq i < j \leq n$. For each TPS_{ij}, a score is calculated which is based on the probability of i^{th} token T_i to appear first and j^{th} token T_j to appear after T_i in flow pools. The running time for calculating the token pair subsequence scores is $O(n^2)$ similar to the token pair conjunction signatures.

Given TPC_{ij}, where $1 \leq i,j \leq n$ and $i \neq j$, score for $x = TPC_{ij}$ is calculated via the statement:

$$S(x) = \log\left(\frac{\text{Prob}\,[TPC_{ij}|L(x) = \text{true}]}{\text{Prob}\,[TPC_{ij}|L(x) = \text{false}]}\right). \tag{1}$$

Given TPS_{ij}, where $1 \leq i < j \leq n$ and $i \neq j$, score for $x = TPS_{ij}$ is calculated via statement:

$$S(x) = \log\left(\frac{\text{Prob}\,[TPS_{ij}|L(x) = \text{true}]}{\text{Prob}\,[TPS_{ij}|L(x) = \text{false}]}\right). \tag{2}$$

The scores defined in equations (1) and (2) are the ratio of the probability of the corresponding token pair in the suspicious flow pool and the probability of the same token pair in the innocuous flow pool. The probability value is calculated considering the order of the tokens for the STP Subsequence signatures, regardless of the token order for the STP Conjunction signatures. The logarithm of the probability value ratio is used as the final token pair score for the sake of readability. This helps us to project the scores in a more human friendly value range especially for big ratio values. The score values are positive when the probability of a token pair in the suspicious flow pool is greater than the probability value of the same token pair in the innocuous flow pool. Those token pairs are the ones that contribute to the final flow score. If the probability of a token pair in the suspicious flow pool is equal to or lower than the probability value in the innocuous flow pool, the corresponding flow score will be equal to zero or negative. Those token pairs can not be used to detect a polymorphic worm and for that reason they are simply ignored.

After calculating the scores for each corresponding token pair, k-means clustering algorithm is applied to token pair scores to distinguish the strong token pairs. A strong token pair is defined as the token pair that is member of a higher priority token-pair cluster, which means that it has a higher value for the ratio of the probability of appearing in a worm sample and the probability of appearing in normal traffic when compared to the lower priority token pair cluster. For the sake of simplicity, we prefer to separate the token pairs in two clusters, namely the "strong pairs" and "weak pairs". Token pairs of both STP Conjunction and STP Subsequence signature sets are labeled as "strong" or "weak," as shown in Figure 5(a) and Figure 5(b) respectively, as a result of the clustering operation.

STP signature scheme consists of token pairs with their related score and strong/weak label. Token-pair scores and strong/weak labels are used in the flow evaluation phase where an overall flow score is calculated. This flow score is used in the decision phase where the flow is classified either as polymorphic worm or innocuous packet.

3.3. Flow evaluation

To classify a flow x either as polymorphic worm or innocuous packet, the probability of a given flow x being worm or not is computed. The corresponding probabilities that must be computed are $\text{Prob}\left[L(x) = \text{true}|x\right]$ and $\text{Prob}\left[L(x) = false|x\right]$. Similar to the scores of the token pairs, the flow score $S(x)$ will be calculated using the ratio of above defined probabilities, which will guide us through the decision phase.

Using Bayes Theorem, $\text{Prob}\left[L(x) = \text{true}|x\right]$ and $\text{Prob}\left[L(x) = \text{false}|x\right]$ can, respectively, be written as

$$\text{Prob}\left[L(x) = \text{true}|x\right] = \frac{\text{Prob}\left[x|L(x) = \text{true}\right]}{\text{Prob}\left[x\right]} * \text{Prob}\left[L(x) = \text{true}\right] \tag{3}$$

$$\text{Prob}\left[L(x) = \text{false}|x\right] = \frac{\text{Prob}\left[x|L(x) = \text{false}\right]}{\text{Prob}\left[x\right]} * \text{Prob}\left[L(x) = \text{false}\right]. \tag{4}$$

Then, the ratio of these probabilities is

$$\frac{\frac{\text{Prob}[x|L(x)=\text{true}]}{\text{Prob}[x]} * \text{Prob}\left[L(x) = \text{true}\right]}{\frac{\text{Prob}[x|L(x)=\text{false}]}{\text{Prob}[x]} * \text{Prob}\left[L(x) = \text{false}\right]} = \frac{\text{Prob}\left[x|L(x) = \text{true}\right]}{\text{Prob}\left[x|L(x) = \text{false}\right]} * \frac{\text{Prob}\left[L(x) = \text{true}\right]}{\text{Prob}\left[L(x) = \text{false}\right]}. \tag{5}$$

Initially the probability of a given flow to be a worm or not is equal as stated in equation (6), because we do not have any information before the token analysis phase. Statistical data could also be used to decide on the initial probabilities but that would not have a crucial effect on the flow score. The ratio of these values will multiply all scores with a constant value which means that ignoring this value would also work for us:

$$\text{Prob}\left[L(x) = \text{true}\right] = \text{Prob}\left[L(x) = \text{false}\right] = 0.5. \tag{6}$$

Thus the ratio of the probabilities in equation (5) reduces to

$$\frac{\text{Prob}\,[x|L(x)=\text{true}]}{\text{Prob}\,[x|L(x)=\text{false}]}. \tag{7}$$

Similar to the token pair scores defined in Section 3.2, a token pair based flow score calculation model is needed to calculate equation (7). The mathematical model has been proposed and discussed in Section 3.4.

3.4. Mathematical model for flow score calculation

Given a flow x, the tokens that appear in the flow are discovered and all of the possible token pairs are generated. Ratio of the probabilities given in equation (7) is calculated based on the token-pair probabilities. $\{TP_i\}_{1\leq i\leq m}$ represents $m \leq n$ tokens in the flow and it is a subset of $\{T_i\}_{1\leq i\leq n}$. It means that our flow score calculation model is flexible and can still classify a flow as polymorphic worm although all of the tokens in $\{T_i\}_{1\leq i\leq n}$ do not appear in the flow. Having discovered the m tokens, the probability of these m tokens in the flow pools is found in order to calculate equation (7).

m tokens in the flow are considered as m events. If it had been assumed that these m events were independent, as did in [1], equation (7) could easily be written as

$$\frac{\text{Prob}\,[x|L(x)=\text{true}]}{\text{Prob}\,[x|L(x)=\text{false}]} = \prod_{i=1}^{m} \frac{\text{Prob}\,[TP_i|L(x)=\text{true}]}{\text{Prob}\,[TP_i|L(x)=\text{false}]} \tag{8}$$

On the other hand, we know that the tokens are not independent in contradiction to the token independence assumption made in Polygraph [1]. Polygraph [1] states that assumption of independence can be relaxed as a future work to propose more complex models and actually that is what we do. Assume that two tokens for a web based polymorphic worm are $TP_1 = GET$ and $TP_2 = HTTP/1.1$. It is clear that TP_2 must always follow TP_1 for a legitimate HTTP request. All of the tokens in $\{T_i\}_{1\leq i\leq n}$ have similar relations bounded with the exploit logic and network protocol command structure. Therefore a decision should be made on the dependency depth of tokens in order to complete our probabilistic model.

Given n dependent events E_1, E_2, ..., E_n, probability of these n dependent events is

$$P(E_1 \cap E_2 \cap ... \cap E_n) = P(E_1) * P(E_2|E_1) * P(E_3|E_1 \cap E_2) * ... * P(E_n|E_1 \cap E_2 \cap ... \cap E_{n-1}) \tag{9}$$

If it is assumed that the tokens are fully dependent, using the conditional probability rule of equation (9), equation (7) can be re-written as

$$\begin{aligned}\frac{\text{Prob}\,[x|L(x)=\text{true}]}{\text{Prob}\,[x|L(x)=\text{false}]} &= \frac{\text{Prob}\,[TP_1|L(x)=\text{true}]}{\text{Prob}\,[TP_1|L(x)=\text{false}]} * \frac{\text{Prob}\,[TP_2|TP_1 \cap L(x)=\text{true}]}{\text{Prob}\,[TP_2|TP_1 \cap L(x)=\text{false}]} * \cdots \\ &\cdots * \frac{\text{Prob}\,[TP_m|TP_1 \cap ... \cap TP_{m-1} \cap L(x)=\text{true}]}{\text{Prob}\,[TP_m|TP_1 \cap ... \cap TP_{m-1} \cap L(x)=\text{false}]}.\end{aligned} \tag{10}$$

Equation (10) has the assumption that the tokens are fully dependent. Instead, we claim that equation (10) can be approximated with the assumption that the tokens are dependent in pairs. STP signature scheme is based on token pairs. So equation (7) is calculated using the token pairs that appear in the flow.

Using the notation

$$\frac{\text{Prob}\,[TP_{ij}|L(x)=\text{true}]}{\text{Prob}\,[TP_{ij}|L(x)=\text{false}]} \equiv \frac{\text{Prob}\,[TP_{i}|L(x)=\text{true}]}{\text{Prob}\,[TP_{i}|L(x)=\text{false}]} * \frac{\text{Prob}\,[TP_{j}|TP_i \cap L(x)=\text{true}]}{\text{Prob}\,[TP_{j}|TP_i \cap L(x)=\text{false}]},$$

equation (7) can be re-written in terms of all token pairs TP_{ij} that appear in the flow as

$$\prod \frac{\text{Prob}\,[TP_{ij}|L(x)=\text{true}]}{\text{Prob}\,[TP_{ij}|L(x)=\text{false}]}. \tag{11}$$

A final score $S(x)$ must be calculated for the flow x to be used in the decision phase. The score is the logarithm of equation (11) and is expressed as

$$S(x) = \log\left(\prod \frac{\text{Prob}\,[TP_{ij}|L(x)=\text{true}]}{\text{Prob}\,[TP_{ij}|L(x)=\text{false}]}\right) = \sum \log\left(\frac{\text{Prob}\,[TP_{ij}|L(x)=\text{true}]}{\text{Prob}\,[TP_{ij}|L(x)=\text{false}]}\right) \tag{12}$$

In fact, equation (12) means that the final score is nothing more than the sum of discovered token pairs' scores, which were already defined for STP Conjunction and Subsequence signatures in equations (1) and (2), respectively.

In accordance with equation (12), STP Conjunction score $S(x)$ is calculated as follows for all token pair conjunctions TPC_{ij} that appear in the flow:

$$S(x) = \sum \log\left(\frac{\text{Prob}\,[TPC_{ij}|L(x)=\text{true}]}{\text{Prob}\,[TPC_{ij}|L(x)=\text{false}]}\right). \tag{13}$$

STP Subsequence score $S(x)$ is calculated as follows for all token pair subsequences TPS_{ij} that appear in the flow:

$$S(x) = \sum \log\left(\frac{\text{Prob}\,[TPS_{ij}|L(x)=\text{true}]}{\text{Prob}\,[TPS_{ij}|L(x)=\text{false}]}\right). \tag{14}$$

It is important to keep in mind that strong token pairs have higher priority than the weak token pairs. There exists m tokens and $m-1$ token pairs initially. If weak token pairs appear in the flow between the first and the last token of a strong token pair, then these weak token pairs are ignored and only the strong token pair contributes to the overall score. In this case less than $m-1$ token pairs are considered in equations (13) and (14) as illustrated in Figure 6.

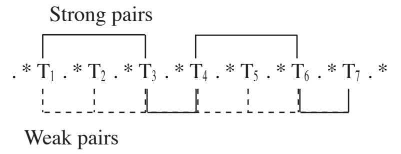


Figure 6. Token-pair processing.

For the sample flow $T_1.{*}T_2.{*}T_3.{*}T_4.{*}T_5.{*}T_6.{*}T_7$, depicted in Figure 6, assume that T_{12}, T_{23}, T_{45}, and T_{56} have been discovered as weak token pairs and T_{13}, T_{34}, T_{46}, and T_{67} have been discovered as strong token pairs. STP flow score calculation method considers T_{13} strong token pair score instead of the sum of T_{12} and T_{23} weak token pair scores. T_{46} is considered instead of T_{45} and T_{56} for the same reason. Therefore, 4 token pairs are considered by the flow score calculation model for a flow that consists of 7 tokens and 6 adjacent token pairs.

The tokens that exist in a given given flow can be extracted in linear time as described in Section 3.2. After extracting the tokens, the token pair scores can simply be added from a lookup table when using the simple flow score calculation method already described in [15]. Therefore the run time for the simple flow score calculation method is $O(n) + (n-1) * c_1 + (n-1) * c_2$, where c_1 and c_2 are the constant costs for the token pair score lookup and score addition costs, respectively.

After extracting the tokens in linear time, the proposed flow score calculation method checks whether there exists a strong token pair starting with the first token extracted. STP Conjunction and Subsequence signatures structure is given in Figure 5(a), and Figure 5(b), respectively. The signature set has the token pair, related score and strong or weak label for each row. Therefore searching for the strong token pairs that start with the first token extracted is a simple lookup in the signature set. If there is a hit in the flow, the flow score calculation method adds the score associated with the strong token pair and the same lookup is performed for the last token of the last processed token pair. If there is no strong token pair starting with the specified token, the score for the weak token pair which starts with the specified token and ends with the next token is added to the flow score. The process continues until the last token in the flow is touched. It is obvious that there are two extra lookup costs which do not exist for the simple flow score calculation method described in [15]. One extra lookup is performed to learn if there exists a strong token pair starting with the token processed, and the other is to learn whether the strong token pair exists in the inspected flow or not. In the worst case, all of the adjacent token pairs are strong token pairs so the run time will be $O(n)+(n-1)*c_1+(n-1)*c_2+(n-1)*c_3+(n-1)*c_4$, where c_1, c_2, c_3 , and c_4 are the constant costs for the token pair score lookup, score addition, strong token pair lookup and strong token pair existence lookup costs, respectively. In the best case, there will exist a strong token pair which starts with the first token found in the flow and ends with the last token found in the flow. In this case, the run time will be $O(n) + c_1 + c_2 + c_3 + c_4$. As a result of the worst case and best case run time analysis for the proposed flow score calculation method, it is seen that the method differs from the simple flow score calculation method only by a constant which means that the improvement in the simple flow score calculation method causes no major impact on the run time of the algorithm.

Having calculated the score associated to a network flow under inspection, a decision rule needs to be used in order to finalize the classification process. The decision rule defined in Section 3.5 is used to give the final decision about the classification label of the flow.

3.5. Decision rule

A score for the flow based on our model is calculated as described in Section 3.4. The ratio of probabilities for each token pair that contributes to the final score may be large and multiplication of these probabilities would yield a larger number which is not human readable and also hard to scale. For this reason the logarithmic values of these probabilities are used and sum up for calculating the overall score $S(x)$.

The score $S(x)$ is compared to a threshold value to decide whether the flow contains a polymorphic worm or innocuous packet. The threshold value E is calculated for a maximum tolerable false-positive value as did in Polygraph [1]. E is selected to minimize the false negative classifications in the suspicious flow pool while keeping the false positive rate lower than the desired maximum false positive rate in the innocuous flow pool. The decision rule is given in equation (15).

$$\text{Decision rule:} \begin{cases} L(x) = \text{true}, & \text{if } S(x) \geq E \\ L(x) = \text{false}, & \text{if } S(x) < E \end{cases} \tag{15}$$

Token pair scores are the atomic parts of the flow score to be calculated. Token pair scores depend on their probability in the flow pools. As these probability values are close to each other, the ratio of them will be close to 1, which will result a token pair score close to 0. As long as a token pair's suspicious flow pool probability is much higher than its innocuous flow pool probability, the related token pair score will also be high. This is where the strong token pair concept helps us. After calculating the token pair scores, k-means clustering algorithm is utilized to discover strong token pairs and weak token pairs. There is a possibility of ignoring a strong token pair just because of there are at least two adjacent weak token pairs where the first weak token pair starts with strong token pair's first token and the last weak token pair ends with strong token pair's last token. The flow score evaluation method considers the strong token pair instead of the adjacent weak token pairs at the decision phase.

4. Experimental results and analysis

False positive and false negative performance of STP Conjunction and Subsequence signature classes have been analyzed through experimental work performed against two polymorphic worms. A polymorphic worm that uses the Apache-Knacker exploit [2], and polymorphic version of the Lion worm [26] which use the BIND-TSIG [3] DNS vulnerability has been used. For the sake of comparing our results to Polygraph [1], these are the same worms that were used in Polygraph. In addition to that, our token-pair based mathematical model for flow score calculation has also been analyzed in comparison with the Fully Dependent and Independent models which were discussed earlier in Section 3.4. Matlab 7.0 environment has been used for simulations.

Suspicious flow pools used in experiments consist of samples generated for the mentioned polymorphic worms. A 3-day network trace of HTTP protocol for the HTTP Worm(Apache-Knacker) and 3-day network trace of DNS protocol for the BIND-TSIG worm has been used as our innocuous pool at the signature generation and token pair score calculation phase. 5-day network traces for HTTP and DNS protocols were used for evaluation purposes where the false positive performance is investigated. The network that was used to capture the traces, hosts an external Apache web server and BIND DNS server of a very large research institute.

Experimental results for the Apache-Knacker worm and BIND-TSIG worm are given in Table 1. STP signatures perform at least as good as Polygraph conjunction and token subsequence signatures. False negatives were always 0%, which means that the generated signatures never failed to detect the worm. False positive performance is also good. Polygraph [1] proposes three types of signatures. These are Conjunction signatures, Token-subsequence signatures and Bayes signatures. Conjunction signatures simply look for all of the extracted tokens in a given flow. Token-subsequence signatures require matching all of the tokens in a given order. A Bayes score is calculated for each token and total score of the tokens that appear in a flow is compared to a threshold value when using Bayes signatures. Polygraph's token subsequence and conjunction signatures are not flexible. Non-existence of only one token would cause failure to detect the worm. Naive Bayes signatures with independence assumption are much more flexible but experimental results in [1] show that it does not perform different then the Best Substring signatures, which were argued to be insufficient to detect polymorphic worms.

That is because the Bayes score for a specific token is always bigger than the threshold value and existence of other tokens do not contribute to the result. The only token to be effective on the decision is the one with the largest Bayes score. However this behavior can be relaxed by using a higher threshold value.

It is not necessary to include all of the extracted tokens in Polygraph's Conjunction and Token-subsequence signatures. That causes the signature to be longer than necessary and fail to detect polymorphic worm in case at least one of the tokens do not appear in a new version of the same worm. STP scores give us an idea about which token to appear together with another one has valuable information to us. STP scores are sometimes equal to 0 or close to 0 which means that the innocuous traffic pool contains the token pair as much as the suspicious flow pool does. A minimum threshold score that a token-pair should have, can be specified for it to appear in the conjunction or token-subsequence signatures. That makes Polygraph's Conjunction and Token-subsequence signatures more flexible to changes in new forms of the same polymorphic worm.

Let's analyze the Apache-Knacker worm. Following compares a Token set, a Polygraph conjunction signature, and Polygraph subsequence signature:

Token set	T_1='GET', T_2='HTTP/1.1\r\n', T_3=':' , T_4='\r\nHost:', T_5='\r\n', T_6='\xFF\xBF'
Polygraph conjunction signature	'GET ', ' HTTP/1.1\r\n', ': ', '\r\nHost: ', '\r\n', ': ', '\r\nHost: ', '\xFF\xBF', '\r\n'
Polygraph subsequence signature	GET .* HTTP/1.1\r\n.*: .* \r\nHost: .*\r\n.*: .*\r\nHost: .* \xFF\xBF.*\r\n

Table 1. Apache-Knacker and BIND-TSIG worms experimental results.

Signature class	Apache-Knacker worm		BIND-TSIG worm	
	False +	False −	False +	False −
STP Subsequence	0%	0%	0%	0%
STP Conjunction	0%	0%	0%	0%
Polygraph's Subsequence	0%	0%	0%	0%
Polygraph's Conjunction	0.00175%	0%	0%	0%
Polygraph's Bayes	0.00179%	0%	0.00022%	0%

Token pair score calculated for T_1 and T_2 is zero for STP Subsequence and Conjunction signatures. That is quite normal because the substring "GET HTTP/1.1" is very common for a valid HTTP request regardless of whether the payload contains a worm or not. Analyzing all of the token-pair scores, the intermediate tokens which do not really contribute to the final decision can easily be canceled for any signature based polymorphic worm detection technique proposed. This method can also be used to cancel weak token pairs of STP signatures but that would have no effect on worm detection success rate since STP signatures flow evaluation method always considers strong token pairs when there is a probability of ignoring it because of the adjacent weak token pairs. STP signatures pre-calculate the scores for all of the possible token pairs once and than use this value if and only if it matches the pair. On the other hand, canceling the weak token pairs may accelerate the polymorphic worm detection process for STP signatures by reducing the number of token pairs to search for.

The problem with naive Bayes signatures is that it performs similar to the Best Substring signatures, which is found to be insufficient to detect polymorphic worms. [1] also argues this situation and it is stated that this can be handled by using a higher threshold value (lower maximum false positive rate). Bayes signatures will not be affected by this change if only one token's score is greater than the threshold value. As a result, the token with a high score will be enough to classify the flow as a polymorphic worm which may lead false positives. Bayes signatures given in Polygraph are an impressive contribution to propose a simple and flexible way to detect polymorphic worms. STP signatures improve this powerful technique by exploiting the independence assumption which is also argued by the authors in [1]. This independence assumption is exploited by dealing with the dependence between two tokens of token pairs. STP Subsequence and Conjunction signatures are in fact the output of this improvement which combines the flexibility of Bayes signatures with the low false positive rates of Conjunction and Token-subsequence signatures defined in [1].

Flow probabilities are modeled using the token pairs appearing in the flow, as described in Section 3.4. The mathematical model for flow score calculation has been analyzed in comparison with the flow probability models with token independence assumption and token full dependence assumption. The vulnerability and network protocol logic forces a clear dependence between tokens. All of the tokens have a reason of appearing in the worm flow to realize the intended attack of the polymorphic worm. Assuming that all of the tokens are fully dependent, a unique signature can be used to detect the worm. That is the case for Polygraph's Subsequence and Conjunction signature classes. The order rule that applies to the subsequence signatures makes this signature class stronger than the Conjunction signatures in terms of representing the polymorphic worm more accurate. As a result, subsequence signatures do not perform worse than the conjunction signatures as shown in Table 1, and also in the experimental results given in [1]. Polygraph's subsequence signatures are not resilient to the slight changes in the polymorphic worm token structure. If the attacker replaces a token for a new member of the same polymorphic worm family, these signatures will fail to detect the worm until enough samples are captured and signature generation is performed again. This is the case where the power of Bayes technique joins the game. STP signatures flow probability model successfully represents the flow probability model with full dependence assumption. STP signatures are resilient to the changes in the polymorphic worm structure as long as all of the strong token pairs are not replaced. The detailed analysis of the experimental results for the Apache-Knacker polymorphic worm is given as follows.

Token set: T_1='GET', T_2='HTTP/1.1\r\n', T_3=':' , T_4='\r\nHost:', T_5='\r\n', T_6='\xFF\xBF'

All of the samples for the Apache-Knacker worm in the suspicious flow pool matches the expression T_1 .* T_2 .* T_3 .* T_4 .* T_5 .* T_3 .* T_4 .* T_6 .* T_5 . Sample flows have been produced to represent possible new versions of the Apache-Knacker worm for the cases where one or more patterns are replaced. The original expression has nine patterns to match. The sample flows include all possible subsets. For example, flow represented as T_2 .* T_3 .* T_4 .* T_5 .* T_3 .* T_4 .* T_6 .* T_5 is a sample for the subsets of eight patterns, flow T_1 .* T_2 .* T_3 is a sample for the subsets of three patterns. Simulation has been performed in Matlab 7.0 environment for 433 distinct flow samples and the probability values have been calculated using the innocuous pool which was also used for false positive and false negative detection rate experiments. Simulation results are shown in Figure 7.

Knowing the fact that the worm samples in the suspicious flow pool include all of the patterns introduced earlier, only the innocuous flow pool probabilities for the flow samples was analyzed. Innocuous pool probabilities for the flow samples have been calculated using four different probability models. These are named: Independent, Full Dependent, Token Pair Conjunction and Token Pair Subsequence probability models. *Independent* model has the assumption that the tokens in a flow are independent of each other. Therefore the flow probability is based on the independent probability of each token in the flow. *Full Dependent* model assumes that the tokens are strictly dependent which is more realistic. Therefore the flow probability is based on the dependency

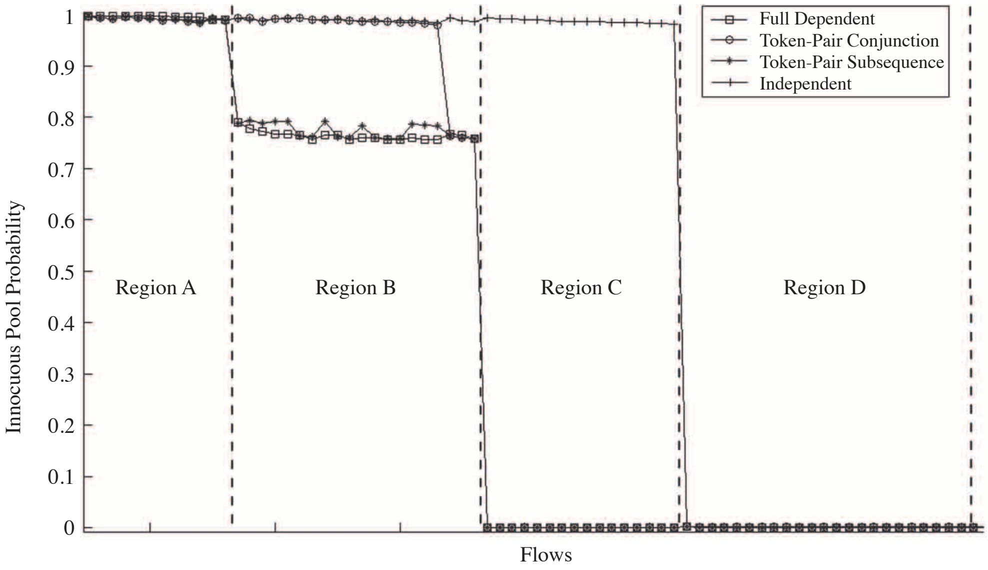


Figure 7. Flow probabilities.

of all tokens in the flow. *Token Pair Conjunction* and *Token Pair Subsequence* models have been proposed to improve the Independent model by representing the Full Dependent model in a better way. Token pair probability models are based on the probability of token pairs in the flow. The order of the tokens is ignored by the conjunction model where the order of the tokens are considered by the subsequence model. Figure 7 has been separated into four regions: "Region A", "Region B", "Region C" and "Region D". The behavior of different probability models have been discussed for each region first. An overall evaluation of the simulation results for all of the regions is given next.

Flow samples in Region A and Region B have high probability of appearing in the innocuous flow pool. The four probability models have similar results for the flow samples in Region A. These samples look more like innocuous traffic. Non-existence of a single token would cause Polygraph's token subsequence and conjunction signatures to detect false negative. Also, the overall flow score for Polygraph's Bayes and STP signatures is lower than the specified threshold value for the sample flows in Region A and Region B. In the case related tokens are replaced in the new versions of the polymorphic worm, resulting in flow samples in Region A and Region B, neither Polygraph nor Token-Pair signatures would successfully detect these worms. In Region B, Full Dependent and Token-Pair Subsequence probability models have similar results. Depending on the token pairs in the flow samples, Token-Pair Conjunction model behaves similar to Independent model or Full Dependent model. The flow samples where Full Dependent and Token-Pair Subsequence models have lower probability values than Independent and Token-Pair Conjunction models in Region B, contain token pairs that order of the tokens change the result. Anyway, the four probability models still have high probability values in Region B. As a result, polymorphic worm will not be successfully detected for the flow samples in Region B. The common characteristic for the flow samples in Region A and Region B is that they do not have strong token pairs. These flow samples only have weak token pairs. Existence of strong token pairs in the flow is critical for any signature based polymorphic worm detection technique. Otherwise the signatures would have high false positive rates due to the high innocuous pool probabilities.

Region C is the most interesting part to analyze. There is a clear distinction between Independent model and the others. Independent model results with high probability values while Fully Dependent, Token-Pair Subsequence and Token-Pair Conjunction models have very low probability values of innocuous pool. Tokens of the sample flows in Region C have high probability value independent of each other. This is the reason why Independent model also gives high probability values. These tokens have low probability of appearing together in the innocuous pool. This feature can be exploited by the assumption of fully dependent tokens or strong token pairs. The two Token-Pair models and Full Dependent model do not have the independence assumption. As a result, independent model classifies these possible new versions of the polymorphic worm in Region C as innocuous traffic which causes false negative decision. On the other hand, other probability models can successfully detect the polymorphic worm. In other words, Polygraph's Bayes signature fails to detect the worm while STP signatures still detect the new versions of the polymorphic worm containing strong token pairs in Region C.

Region D also consists of flow samples with strong token pairs. For these possible new versions of the polymorphic worm, all of the four probability models have low probability values. The difference between the flow samples in Region C and Region D is that at least one token in the flows of Region D individually has low probability value.

Root mean square (RMS) errors for Independent, Token-Pair Conjunction and Token-Pair Subsequence probability models against the Fully Dependent model are given in Table 2. Token-Pair Conjunction and Subsequence models outperform the Independent model. As a result of considering the order of tokens, Token-Pair Subsequence model is more successful than the Conjunction model. Token-Pair Conjunction model performs as good as the Token-Pair Subsequence model in Region A where the flow samples consist of token pairs with a high probability of appearing in the innocuous flow pool. RMS errors for the Token-Pair Conjunction and Token-Pair Subsequence models are very low or equal to 0 for flow samples in Regions C and D, where at least one strong token pair exists in the sample flows. The effect of considering the order of tokens can clearly be seen from the RMS error values of Regions B and D for Token-Pair Conjunction and Subsequence models. The Independent model has unacceptable performance for the flow samples in Region C with an RMS error of 0.9864, while Token-Pair models successfully represent the Fully Dependent model with an RMS error of 0. Eventually the Independent Model, which is used by Polygraph's Bayes signatures, has an overall RMS error value of 0.3916, which proves that it is not resilient to the changes in the polymorphic worm structure. On the other hand, Token-Pair Subsequence model performs approximately ten times better than the Token-Pair Conjunction model with an overall RMS error value of 0.0084. Token-Pair Subsequence model successfully estimates the Full Dependent model with a low RMS error for all regions. Therefore, STP Subsequence signature class is the best option to detect polymorphic worms.

Table 2. RMS errors.

Probability model	Region A	Region B	Region C	Region D	Overall
Independent	0.0039	0.2247	0.9864	0.0017	0.3916
Token-Pair Conjunction	0.0043	0.2034	0	0.00102	0.0885
Token-Pair Subsequence	0.0043	0.01908	0	0.000013	0.0084

5. Conclusion

Malicious codes that utilize polymorphism have become the new challenge for today's information security community. Worms deserve special interest among information security threats because they do not need human interaction and can spread over large networks very quickly. Cryptographic techniques that help us to protect our information are also used by malicious code developers to prevent detection. As the security techniques evolve, attack techniques will also evolve and more advanced detection techniques will be needed. In this work, STP Subsequence and Conjunction signatures have been proposed for this special cyber threat. Experimental results show that these signature types are as flexible as naive Bayes signatures and have low false positive / false negative rates as Conjunction and Token-Subsequence signatures proposed in Polygraph. We have also proven that token independence assumption may cause false negatives for new versions of a polymorphic worm. Strong token-pair idea contributes to the solution of this problem by exploiting the dependence between the tokens while keeping our flexible signature mechanism resilient to the changes in the polymorphic worm structure as long as strong token pairs can be discovered.

References

[1] J. Newsome, B. Karp, D. Song, "Polygraph: Automatically generating signatures for polymorphic worms", IEEE Security and Privacy Symposium, 2005

[2] C. CAN-2003-0245. Apache apr-psprintf memory corruption vulnerability.
http://www.securityfocus.com/bid/7723/discussion/. (May 2008)

[3] CERT Advisory CA-2001-02 Multiple Vulnerabilities in BIND. ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code - VU#196945. http://www.kb.cert.org/vuls/id/196945. (May 2008)

[4] The ADMmutate polymorphic engine. http://www.ktwo.ca/ADMmutate-0.8.4.tar.gz. (February 2007)

[5] The CLET polymorphic engine. http://www.phrack.org/show.php?p=61&a=9. (February 2007)

[6] Z. Li, M. Sanghi, Y. Chen, M.-Y. Kao, B. Chavez, "Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience", In Proceedings of the IEEE Symposium on Security and Privacy, 2006

[7] C. Kreibich and J. Crowcroft, "Honeycomb - creating intrusion detection signatures using honeypots", In Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II), November 2003.

[8] H. A. Kim and B. Karp, "Autograph: toward automated, distributed worm signature detection", In Proceedings of the 13th USENIX Security Symposium, August 2004.

[9] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting", In Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI), Dec. 2004.

[10] V. Yegneswaran, J. Giffin, P. Barford, and S. Jha, "An architecture for generating semantic-aware signatures", In USENIX Security Symposium, 2005.

[11] Y. Tang and S. Chen, "Defending against Internet worms: A signature-based approach", In Proceedings of Infocom, 2003.

[12] Y. Tang and S. Chen, “An Automated Signature-Based Approach against Polymorphic Internet Worms”, IEEE Transactions on Parallel and Distributed Systems, Volume 18, Issue 7, pp. 879-892, 2007.

[13] L. Cavallaro, A. Lanzi, L. Mayer, M. Monga , “LISABETH: automated content-based signature generator for zero-day polymorphic worms”, International Conference on Software Engineering, Proceedings of the fourth international workshop on Software engineering for secure systems, pp. 41-48, 2008

[14] M. Van Gundy, H. Chen, G. Vigna, “Feature Omission Vulnerabilities: Thwarting Signature Generation for Polymorphic Worms”, 23^{rd} Annual Computer Security Applications Conference, pp. 74-85, 2007.

[15] B. Bayoğlu, İ. Soğukpınar, “Polymorphic Worm Detection Using Token-Pair Signatures”, International Conference on Pervasive Services, 4^{th} International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, pp. 7-12, 2008.

[16] J. Zhang, H. Duan, L. Wang, Y. Guan, “A Fast Method of Signature Generation for Polymorphic Worms”, International Conference on Computer and Electrical Engineering, pp. 8-13, 2008.

[17] M. Christodorescu, S. Jha, S.A. Seshia, D. Song, R.E. Bryant, “Semantics-aware malware detection”, In Proceedings of the IEEE Symposium on Security and Privacy, 2005.

[18] Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna, “Polymorphic worm detection using structural information of executables”, In Proceedings of Recent Advances in Intrusion Detection (RAID), 2005.

[19] Liang and R. Sekar, “Fast and automated generation of attack signatures: A basis for building self-protecting servers”, In Proceedings of ACM CCS, 2005.

[20] S. Chen, X. Wang, L. Liu, X. Zhang, Z. Zhang, “WormTerminator: An Effective Containment of Unknown and Polymorphic Fast Spreading Worms”, In Proceedings of the ACM/IEEE symposium on Architecture for Networking and Communications systems, pp. 173-182, 2006.

[21] G. Manzini and P. Ferragina, “Engineering a lightweight suffix array construction algorithm”, Algorithmica, 40(1), 2004.

[22] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton, “Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment”, Science, vol. 262, pp. 208-214, Oct. 1993.

[23] R. Perdisci, D. Dagon, W. Lee, P. Fogla, and M. Sharif, “Misleading worm signature generators using deliberate noise injection”, In IEEE Security and Privacy Symposium, pp. 15-31, 2006.

[24] CVE-2004-0597: Multiple buffer overflows in libpng 1.2.5. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-0597. (June 2008)

[25] L. Hui, “Color set size problem with applications to string matching”, In Proceedings of the 3rd Symposium on Combinatorial Pattern Matching, 1992.

[26] SANS Institute: Lion worm. http://www.sans.org/y2k/lion.htm (June 2006)