Al-Khwarizmi Engineering Journal, Vol. 8, No. 3, PP 40 - 52 (2012)

Al-Khwarizmi
Engineering
Journal

Design a Security Network System against Internet Worms

Rana Dhia'a Abdu-Aljabar

Department of Information and Communication Engineering/ College of Information Engineering/ University of Al-Nahrain

(Received 5 September 2011; accepted 4 March 2012)

Abstract

Active worms have posed a major security threat to the Internet, and many research efforts have focused on them. This paper is interested in internet worm that spreads via TCP, which accounts for the majority of internet traffic. It presents an approach that use a hybrid solution between two detection algorithms: behavior base detection and signature base detection to have the features of each of them. The aim of this study is to have a good solution of detecting worm and stealthy worm with the feature of the speed. This proposal was designed in distributed collaborative scheme based on the small-world network model to effectively improve the system performance.

Keywords*: Worm detection, behavior base, signature base, signature generation.*

1. Introduction

A computer worm is a self-replicating computer program. It uses a network to send copies of itself to other computer terminals on the network and it may do so without any user intervention.

Currently, worms are serious security threat that may cause congestion in the network which leads to large queuing delays, and high packet loss. Since Code Red and Nimda worms were spread in 2001, Epidemic-style attacks have caused huge damages. The internet is an influential function in the economy and reckon mainstay to the life. Once the internet is broken down, it will cause a huge economic loss. So the worm handling must be automatic in order to have any chance of success because worms spread too fast [1].

The main difference between viruses and worms is the method in which they reproduce and spread. A virus is dependent upon a host file or boot sector, and the transfer of files between machines to spread, while a worm can run completely independently and spreads itself through network connections. [2].

Intrusion detection systems (IDS) serve three essential security functions: they monitor, detect, and respond to unauthorized activities. There are two basic types of intrusion detection: host-based and network-based. Host based IDSs examine data held on individual computers that serve as hosts, while network-based IDSs examine data exchanged between computers [3].

False positive (FP) and false negative detection errors are inevitably generated in any practical detection system [4]. A false positive indicates a normal string that is incorrectly identified as an alarm (suspicious string). A false negative means missing the detection of a suspicious string by incorrectly labeling a worm (or suspicious string) as a normal one [5]. Ideally, a perfect detection system would generate no false positives and no false negatives and would only raise an alarm when the actual worm is detected.

In recent years, efficient defense against distributed attacks has been a hot topic in network security community. Instead of establishing brand-new, dedicating systems, collaborating widely deployed, single-point network security applications for co-defense would be more feasible. Through collaboration, a security shield that covers infrastructure of multiple network domains could be built without significant modification. Besides keeping most of original functions, collaboration offers individual security applications wider views of dynamic situations around which otherwise may not be observed. It improves the resilience and confidence of participating security applications to handle sophisticated security problems in optimized strategies. Existing collaborative schemes for distributed defense could be classified into either centralized or decentralized category.

This paper obtained a hybrid solution between behavior base and signature base distributed over a multi sensor in the network in decentralized collaborative scheme to get a good solution for a worm attack.

The remainder of this paper is organized as follows: Section2 describes related works. Section3 shows the proposed design scheme. Section4 explains the internet worm activity in the network. Section 5 explains how the behavior base algorithm builds. Section 6 illustrates the signature generation briefly. Section 7 discussed the central alarm work. Section 8 shows the results of this design. Section 9 concludes the proposed mechanism and section 10 the references.

2. Related Work

Most recent research on detecting worms concentrates on propagation modeling. Defending against them remains a challenge due to their continuous evolution. The defense against worms is still an open problem.

X. Yang et al. [6] built algorithm for detecting the worm which has two sub algorithms, the first algorithm “short term algorithm” run well to detect worm, but the second algorithm “longer term algorithm” cannot detect some types of the stealthy worm. The algorithm also cannot hold any equations to determine specification when the equation runs in the algorithm to detect early worm if it has higher rate for value in average of failure connection. Yang algorithm focuses just on detecting which computer contain the worm.

M.M. Rasheed et al.[7] technique is concerned with detecting the internet worm and stealthy internet worm using their behavior. It is an improvement of X. Yang et al algorithm. The average of failure connections by using Artificial Immune System (AIS) is the main factor that his technique depends on in detecting the worm. They showed that their algorithm can detect new types of worms and that intelligent Failure Connection Algorithm (IFCA) operation is faster than traditional algorithm in detecting worms.

S. Behal et al.[8] analyzed the outbound traffic; i.e. extrusion traffic only instead of intrusion traffic. They conclude that only extrusion or intrusion detection is not sufficient to make a network secure but rather these two approaches complement each other to make a network more secure from the threats of malware.

It is widely believed that content-signature-based intrusion detection systems (IDSes) are easily evaded by polymorphic worms, which vary their payload on every infection attempt. In [9] Polygraph, a signature generation system that successfully produces signatures that match polymorphic worms is presented. Polygraph generates signatures that consist of multiple disjoint content substrings. In doing so, Polygraph leverages their insight that for a real-world exploit to function properly, multiple invariant substrings must often be present in all variants of a payload; these substrings typically correspond to protocol framing, return addresses, and in some cases, poorly obfuscated code. It contributes a definition of the polymorphic signature generation problem; propose classes of signature suited for matching polymorphic worm payloads; and present algorithms for automatic generation of signatures in these classes. It's evaluation of these algorithms on a range of polymorphic worms demonstrates that Polygraph produces signatures for polymorphic worms that exhibit low false negatives and false positives.

Zero-day polymorphic worms pose a serious threat to the security of internet infrastructures. Given their rapid propagation, it is crucial to detect them at edge networks and automatically generate signatures in the early stages of infection. In Hamsa[10], a network-based automated signature generation system for polymorphic worms. Evaluation based on a range of polymorphic worms and polymorphic engines demonstrates that Hamsa significantly outperforms Polygraph [9] in terms of efficiency, accuracy, and attack resilience.

In recent years, efficient defense against distributed attacks has been a hot topic in network security community. There are two popular collaboration schemes. Schnackenberg et al. proposed a centralized coordinative scheme called CITRA [11] for network intrusion detection in 2001. A central coordinator responds for coordinating countermeasures based on a complete view of the network. Janakiraman et al. [12] introduced a decentralized defense scheme for network intrusion prevention. Information is shared among trusted peers to guard the network against intrusion. The subscription-based group communication is conducted over a P2P architecture, which brings excellent scalability.

Taking advantages of the P2P network, researchers attempted to address the major challenges in large scale collaboration: the scalability and avoidance of central point of failure [13]. They merged multi-dimensional correlation for collaborative intrusion detection [14], and developed self-protecting and self-healing collaborative intrusion detection architecture for the trace-back of fast-flux phishing domains [15].

3. The Whole Proposed System Scheme

This paper used distributed collaborative scheme Based on the small-world network model.

The design of small-world network model obtained a hybrid solution between behavior base and signature base by using signature generation to have a signature of the unknown worm detected by behavior base and update the database of signature base.

The signature-base is distributed over a multi sensor (see Figure 1) in the network to get a good defense for a worm attack.

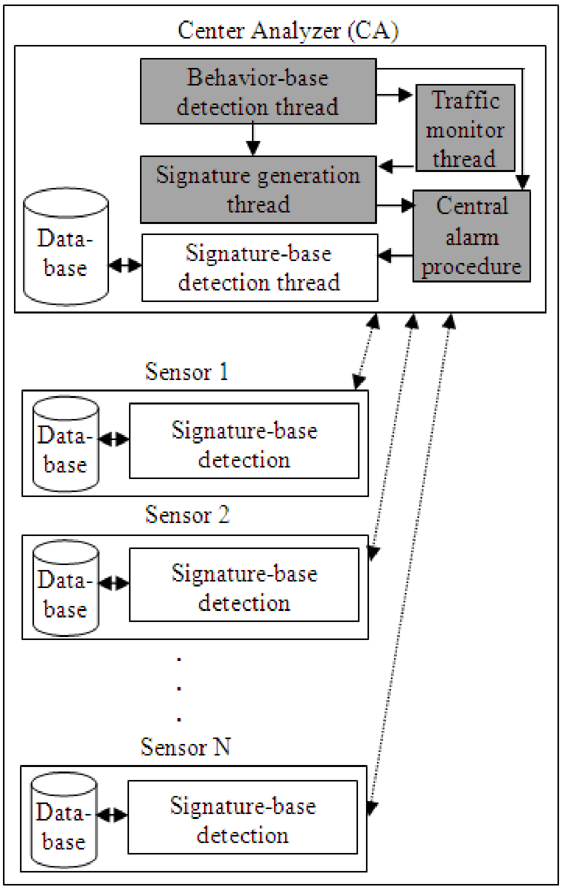

Fig. 1.The Small World Network Model for Worms' Detection .

One of the sensors has in addition to Signature base detection thread, a Behavior-base detection thread, a Signature generation thread, a Monitoring traffic thread and a Central alarm procedure. This sensor is named a *Central Analyzer (CA)*.

This multiple small world networks model is like Figure 1 connected with each other through (CA) in decentralized scheme (see Figure 2) to improve the performance by efficient defense against distributed attacks.

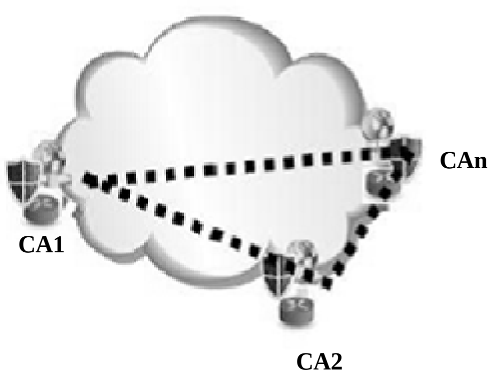

Fig. 2. Decentralized Collaboration.

The decentralized scheme is much flexible. It behaves similaryl to the manners of Peer-to-Peer (P2P) networks. This is due to the fact that most of decentralized schemes are developed on top of P2P network protocols [16]. The P2P collaborative architecture gives decentralized schemes good scalability. Theoretically, any network node features compatible collaboration protocols could participate, so that the boundary of covered network could be loose. Rather than having a collaborative server in centralized scheme, each participating node takes responsibilities for collaboration, as shown in Figure 2, which brings more flexibility for self-management. Obviously, the cost for application is relatively low, since it does not require any modification in network beyond the installation of software.

When the worm detected by the behavior-base thread it will send the source address (src.add) and source port (src.port) that the worm was use it, for both Central alarm procedure responsible for alarming all sensors of its network and other CAs (section 7), and the Signature generation thread (section 6).

The signature generation thread search in the suspensions list generated from monitoring traffic thread (section 6.1) is on the record that has the same src.add and src.port that the behavior-base sends them to it. From this record it will extract the worms samples to generate the new worm signature (section 6.2), and then it will send it to the central alarm procedure, which responsible for updating all database signature base for all sensors of its network and other CAs and motivates the signature base to make immediate scan to the all nodes it responsible for.

4. Worm Scan Activity

Worms spread many connection requests to propagate itself and infect vulnerable hosts on the Internet. When selecting target hosts, worms use a kind of scanning strategies. Code Red and SQL Slammer used random scanning method, and Blaster was a sequential scan worm. In [17], Wu et. al. introduced a selective random scan and a routable scan worm.

When a host makes a connection request via TCP, it sends a SYN packet to a destination address. A connection fails if the destination address does not exist or the destination port is not open. Specifically, if a SYN packet is sent to an unused IP address, an ICMP host which is unreachable packet is returned (see figure 3); if a SYN packet is sent to a used IP address with destination port closed, a TCP RESET packet is returned (See Figure 4).

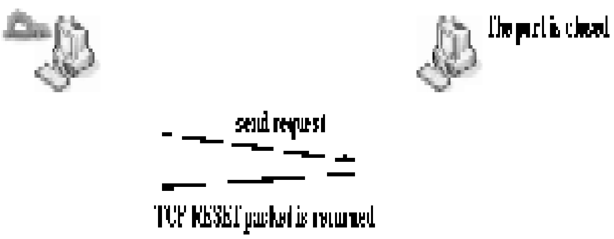

Fig. 3. Destination IP Address Does Not Exist.

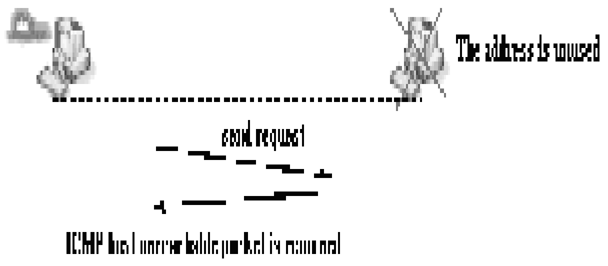

Fig. 4. Destination Port Closed.

If a worm uses random or sequential scanning to prove target hosts, the packets generated by the worm can reach unused IP addresses. In other words, the number of hosts in local network, which are targeted by the worm, largely increases. In case of normal traffic, some internal hosts are related to a specific port number. By checking the number of distinct destination addresses of inbound traffic with the port number, we find an abnormal pattern caused by a worm. The Slammer worm, for example, caused some infected hosts to send up to 30,000 scans at a second [18].

The life of a worm, after it is released, typically includes the following phases: target finding, worm transferring, worm activation, and infection. During the phase of target finding and worm transferring, the worm is active over the Internet, making it possible for network-based intrusion detection systems to catch the worm. The activities in the two latter phases are limited to local machines and are harder to detect by network-based intrusion detection systems [19].

5. Detection Based Behavior Algorithm

This paper adopted the Intelligent Failure Connection Algorithm (IFCA) design [7]; it is an improvement of improved two rotations (ITR) algorithm [6] to detect the worms.

It works in first phase of worm's life which is target finding. In this phase the behavior of worm different from normal user. Comparing with worm activities that scan hundreds of different IP address per second, normal users usually connect to different IP address and Web sites at a slower rate. Particularly, normal users may have the favorite web sites list, and don’t produce so many attempts to connect to random addresses.

Based on the fact that TCP-based worm will attempt TCP connections to different random addresses and result in a large number of connection failures, the traditional worm detection approach mainly focuses on the TCP SYN or ICMP host unreachable packets. In order to make the algorithm more accurately, a better alternative method is to monitor inbound ICMP host unreachable and TCP RESET packets.

The IFCA only records the number of inbound first failed connection packets such as ICMP and TCP RESET packets returned from the external destination address to the internal forged and monitored source IP address.

If normal connection is received; i.e., TCP SYN/ACK, “counter” will be decreased. Only the first failed connection sent from the forged source

IP address to different destination IP address is recorded. Normal network activities are considered to decrease the counter's value. IFCA will remove the "counter" every three days.

The packet should be ignored when the destination IP is recorded into the counter table.

This mechanism records the number of failed connection packets such as ICMP and TCP RESET packets that are returned from the external destination address to the internal forged. It monitored source IP address based in the router. Once detecting the first failed connection packets, the algorithm then extracts (the source address, source port, destination address, destination port) from the packet and creates the record.

In this algorithm, several new equations are applied to detect the worm. It is supposed β =100 and then X= (1 to n) average of failure connection in one minute. Threshold can be processed by the following equation:-

Sum. of threshold=2^(6.65+0.050054(β-X))

The equation depends on the average of failure connection to compute the threshold. IFCA can detect the worm early in usual time. But if the worm cannot be detected in early stage, the algorithm provides more time and new threshold to detect the worm.

T1=(sum. of threshold/average of failure connection)

T2= (time now – time start of the algorithm)

IFCA is dynamic in detecting the worm because it calculates the threshold every time. IFCA detects the worm by comparing T1 to T2 as follows: If (T2 is small or equal to T1) and (the counter is greater than or equal to the summation of Threshold) the worm is detected. Check T1, T2. If (T2 is greater than T1), then go to feed back and decrease the average with new calculate to give other chance to detect the worm. If T1 is smaller than T2, then forward the traffic because it is a normal connection. Whenever the counter value does not exceed the threshold during time cumulative computation phrase, the traffic sent from the corresponding IP address would be forwarded as normal activity (See Figure 5).

When the worm is detected the signature generation will run to find the signature of this worm. as it explained in the next section.

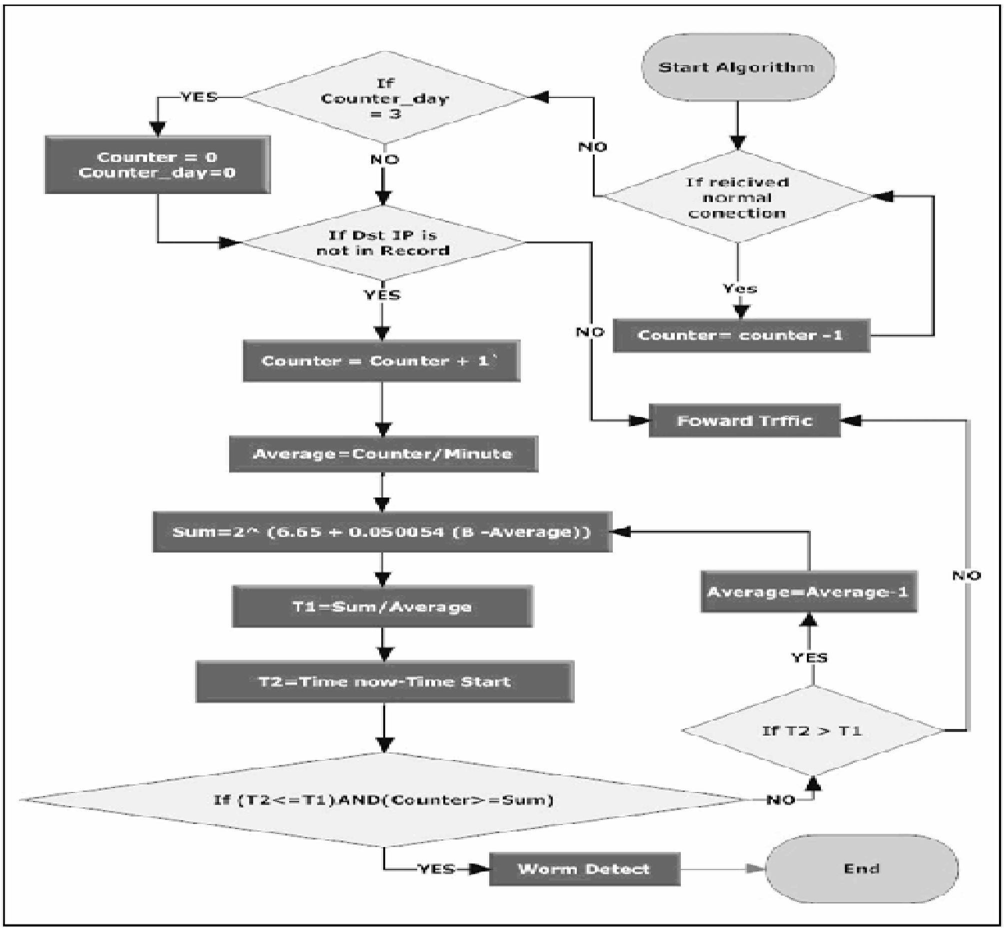

Fig. 5. The Flow Chart of the IFCA.

6. Signature generation

In previous section unknown worm is detected by its behavior. It is a very useful way, but it still has an overload because it still needs to repeat this algorithm each time to the same worm. So this paper it suggests to have full benefit of this algorithm by generating the signature of the worm that has been detected.

6.1. Traffic Monitor

The traffic-monitor thread catches all inbound and outbound data packets only and stores them in a list called suspicions list. From each packet it takes the source address, source port, destination address, and data and stores them in the suspicions list as record fields (Figure 6).

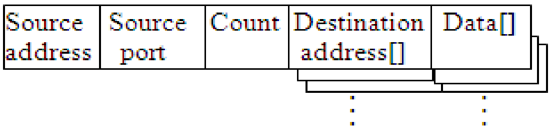

Fig. 6 . Suspicions List Record Fields.

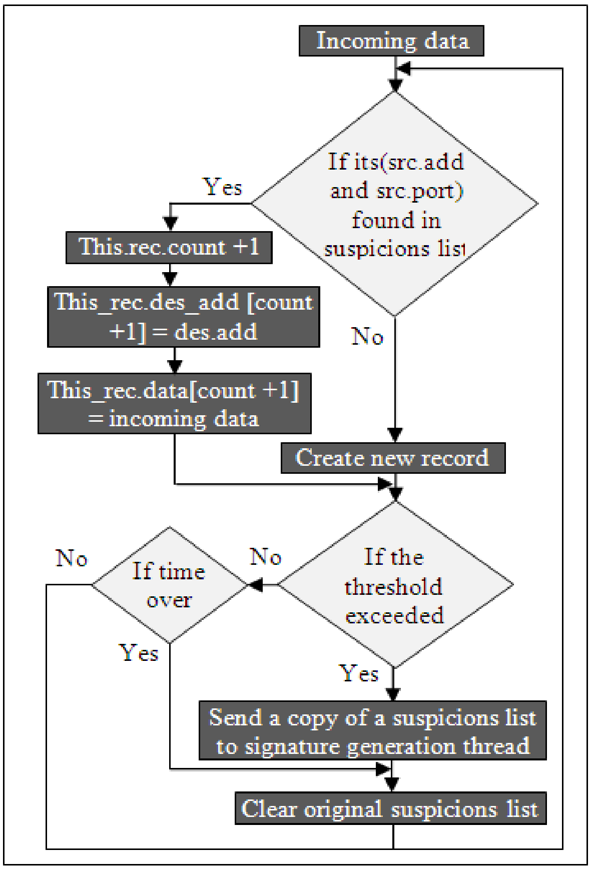
Incoming data
If its(src.add and src.port) found in suspicions list
Yes
This.rec.count +1
This_rec.des_add [count +1] = des.add
This_rec.data[count +1] = incoming data
No
Create new record
No
If time over
No
If the threshold exceeded
Yes
Yes
Send a copy of a suspicions list to signature generation thread
Clear original suspicions list

Fig. 7. Creating the Suspicions List.

After the TCP connection success; data packets will be send; these packets may be a worm packet. This thread will save all packets in a suspicions list till the threshold exceed or time over in behavior base threat then the suspicions list will be cleared and tried again. If the threshold exceeds, it means that there is a worm found by behavior base thread. So the suspicions list must be copied to signature generation thread and then will be cleared. This process is illustrated in Figure 7.

6.2. Signature Generation Algorithm

A signature algorithm is a method for signing a message stored in electronic form. As such, a signed message can be transmitted over a computer network. It represents hand-written signature on the paper. Its main method is to specify the person who is responsible for the documents. But digital signature scheme is superior to hand –written signature in that it is none-forge but easier identification [20].

The generated signatures used are in the form of Simplified Regular Expression (SRE). It will also define a “more specific than” relation on this signature model that allows two signatures to be compared to determine if one is a more specific form of the other [21]. It use this relation to analytically define “the most specific” signature for a polymorphic worm, and hence, formalize the problem of signature generation for polymorphic worms. Based on the concept of the most specific signature, it proposed a signature generation algorithm using multiple sequence alignment. The generated signature is represented in SRE, which is effective and precise because of its successful one-byte invariant extraction and emphasis on the order and distance of extracted invariant parts.

In this section, it focuses on generating accurate signatures for a polymorphic worm. It will propose a more precise signature type, the SRE signature, and it will briefly introduce a signature generation method based on multiple signature alignment (MSA). Finally it will compare the SRE with other signature types.

6.2.1. SRE Signature Type

Motivated by the insufficiency of current signature types for expressing distance restriction, the SRE signature type designed from regular expression. It is believed that regular expression has significant advantages for intrusion detection in terms of flexibility, accuracy, and efficiency

[22]. Regular expressions have been widely used in intrusion detection systems, for example, in Snort and Bro. However, the full regular expression is too complex and its numerous syntax rules are not needed for worm detection. Hence, it introduces the simplified regular expression as a way of representing worm signatures. An SRE signature is a simplified form of a regular expression that contains only two qualifiers, ".*" and ".{k}." These can each be further abbreviated by replacing ".*" with "*", which represents an arbitrary string (including a zero length string), and by replacing ".{k}" with "[k]," which represents a string consisting of k arbitrary characters. For example, "'one'[2]'two'*" is an SRE signature that is equivalent to the regular expression "one.{2}two.*". Suppose that Φ={*,[k]} is the set of the two qualifiers and ∑+ is the set of nonempty strings over a finite alphabet ∑. An SRE signature is defined as follows:

Definition 1 (SRE Signature). An SRE signature is a signature in the form of $(q_0)s_1q_1s_2$ . . . $q_{k-1}s_k(q_k)$, where qi∈ Φ is a qualifier, si ∈ ∑+ is a substring (i ∈ [0; k]), and(q_0) and (q_k) mean q_0 and q_k are optional.

The length of an SRE signature define as the total number of characters in substrings plus the number of qualifiers, and use |X| to denote the length of an SRE signature X. For example, given an SRE signature X = "*'a'[2]'bbb'[1]'cccc'*", |X|= 11 (1 2 3 1 4).

Compared with the previous signature types to be used for worm detection, the SRE signature type is a more precise signature presentation because it can express distance restrictions of adjacent invariant parts using qualifiers (e.g., [k] shows the distance of k bytes). In addition, SRE signatures, in the form of regular expressions, can be easily converted into existing intrusion detection system (IDS) rules, and vice versa. This is illustrated in section 6.2.3.

6.2.2. SRE Signature Generation

The approach comprises three main steps: multiple sequence alignment, noise elimination and signature transformation. It first transforms a set of samples (network flows) of a polymorphic worm into a set of character sequences, and then generate an SRE signature for this worm. Figure 8 illustrates the procedure. 'WormSample1' (''ONEwerTWOtyjfTHREEcxbfd'') to 'WormSample6' (''yuiddONEnsddTWOweredsTHREEnfg'') are six worm samples and 'noise1' and 'noise2' are two noise samples. The first step analyzes and aligns these worm samples and noise flows.

The alignment is represented as a colored matrix, where the greater the number of identical characters in a column, the darker its color. The next step is to identify noise samples using a noise elimination algorithm. Figure 8 shows sequences 'noise1' and 'noise2' correctly identified as noise flows. The remaining sequences are recognized as worm samples. From them, identical characters in the same columns are extracted as invariant bytes of the polymorphic worm. Step 3 produces an SRE signature ''*'ONE'[4]'TWO'*'THREE'*'' by putting distance restrictions between adjacent invariant bytes. This is the most specific signature of the worm. This approach steps will explained briefly bellow:

Step 1: Multiple Sequence Alignment (MSA)

Sequence alignment compares pair or multiple sequences by searching for a series of individual characters or character patterns that are in the same order in the sequences. The definition of multiple sequence alignment is given in Definition 2. An alignment (result) is represented as a matrix A and the row A_p within the matrix represents the aligned sequence of s_p. Gaps ('–') are inserted between elements so that elements with identical characters from different sequences can be aligned in the same columns. As can be seen in Figure 8, it use colors to indicate how many rows containing an identical character for each column of A. Columns with more rows are filled with a darker color.

Definition 2. (Multiple Sequence Alignment.)

Given a family of sequences S ={s1; .; s_k} over an alphabet ∑, $|s_p|$ is the length of the sequence s_p, and an alignment of sequences in S is a (k× N)-matrix A = $(A_{p;i})_{1\le p\le k;1\le i\le N}$ with $\max_{1\le p\le k} |s_p| \le N \le \sum_{1\le p\le k} |s_p|$, if and only if:

1. A_p, i ∈∑∪ {'–'} (where '–' ∉∑ is called a gap);
2. Upon removal of all blanks, row A_p = (A_p, i) $_{1\le i\le N}$ reduces to S_p; and
3. No column consists only of blanks.

Step 2: Noise Elimination

Owing to imperfect suspicious flow classification or worm sample clustering, there may be noise in worm samples. This step finds the noise and removes it. As can be seen in Figure 8 (step 2), sequences (rows in matrix) identified as noise do not have '*' at the end of the sequence name, but the sequences identified as worm samples do.

Imperfect suspicious flow classification or worm sample clustering can produce noise in worm samples which must be eliminated to derive accurate worm signatures. Since MSA in step 1 generates maximum element matches from each pair wise alignment, a limited number of noise flows will not influence the extraction of most common characters. That is, most worm samples can be aligned to get valuable matches even if there are a few noise flows. There are two noise samples at Step 1 in Figure 8, yet the six worm samples are properly aligned to show extractable common invariant parts.

The proposed noise elimination algorithm is designed to improve the accuracy of signature generation in a noise-tolerant way. Given the alignment of k sequences, it defines a noise tolerance rate θ (0≤ θ≤1); and selects [k. θ] sequences as noise within k sequences. The remaining k – [k. θ] sequences will be regarded as worm samples and it will use them to output the final SRE signature. The noise elimination algorithm first determines the invariant bytes of a polymorphic worm. The invariant bytes will be characters that appear more than [k. θ] times in one column. Then we determine [k. θ] noise samples as those containing fewer invariant bytes. The noise elimination algorithm is shown briefly in [23].

The selection of the noise tolerance rate θ is a key point in the noise elimination algorithm. A fixed value of θ could be impractical. If θ is too small, some noise may not be filtered out. In contrast, if θ is too large, some worm samples may be wrongly eliminated as noise and there will be not enough worm samples left. In both cases, the approach may fail to generate accurate signatures. It use an adaptive θ scheme that θ changes its value according to the total number of available sequences (k, and k is required to be over 4), as shown in Formula (1). The objective of this scheme is to ensure that for any k (k > 4), kθ ≈[0.8k – 3.2] sequences will be chosen as noise, k - kθ = k(1 - θ)≈4 + [20%(k – 4)] sequences will be chosen as worm samples. In other words, at least 4 sequences will be chosen as samples. For the remaining k -4 sequences, it chooses 20% as samples (80% as noise), a conservative policy.

$$\theta = 0.8 - 3.2/k \quad (k > 4) \qquad \ldots(1)$$

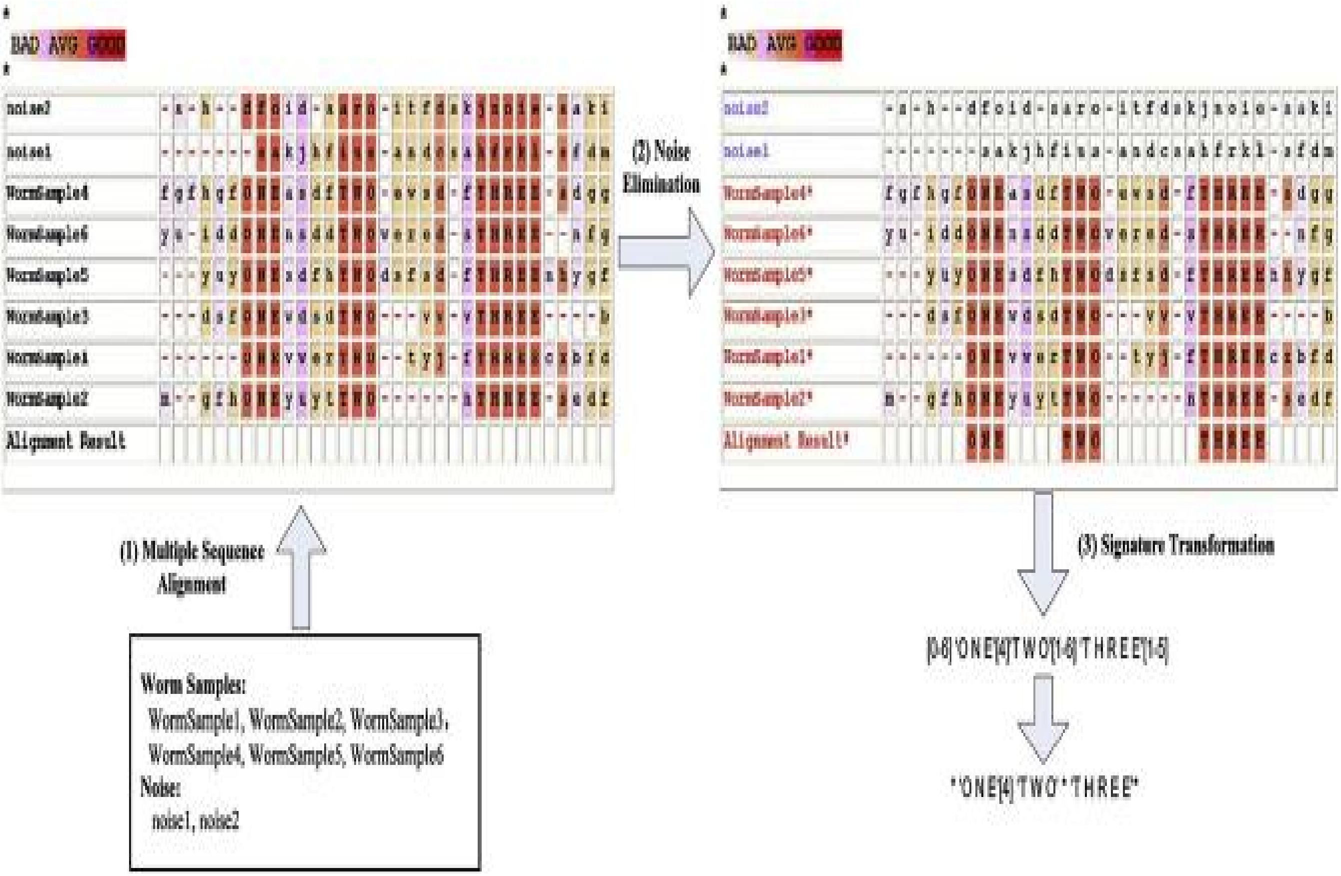

Fig. 8. SRE Signature Generation Steps.

Step 3: SRE Signature Transformation from Alignment Result

An alignment result can be easily transformed into an SRE signature. Given an alignment of multiple worm samples, after the noise elimination and invariant byte extraction, it can get the distance restriction for adjacent invariant bytes by counting the number of in-between none-blank positions. Taking Figure 8 as an example, the corresponding SRE signature is ‘‘[0,6]‘ONE’[4]‘TWO’[1,6] ‘THREE’[1,5]’’, where ‘[1,6]’ is a distance bound restriction meaning that there are at least one and at most six elements between ‘TWO’ and ‘THREE’ in worm samples (WormSample1–WormSample6). Although using ‘[k1, k2]’ to express a distance bound restriction (instead of ‘*’ in SRE signature) is a more precise way to express the range distance restriction, this algorithm [23, 24] does not adoptsuch a distance bound restriction in generated signatures. This is because even though there are some range distance restrictions in polymorphic worms that can be exactly expressed by a bound of ‘[k1, k2]’, it may not be able to extract a perfect bound given inadequate worm samples. For instance it is supposed that a polymorphic worm should have a real range distance restriction with the bound of ‘[1, 100]’. If the worm samples are inadequate, as a result, it may only gets an over- specific bound, like ‘[12,50]’, which will result in a high false negative rate. The last step in this approach outputs SRE signatures by extending each range bound of‘[k1, k2]’ to ‘*’; i.e., extending ‘‘[0,6]‘ONE’[4]‘TWO’[1,6]‘THREE’[1,5]’’ to ‘‘*‘ONE’[4]‘TWO’* ‘THREE’*’’, as in Figure 8. Note that it still keeps the fixed distance restriction in generated SRE signatures (like [4] in Figure 8).

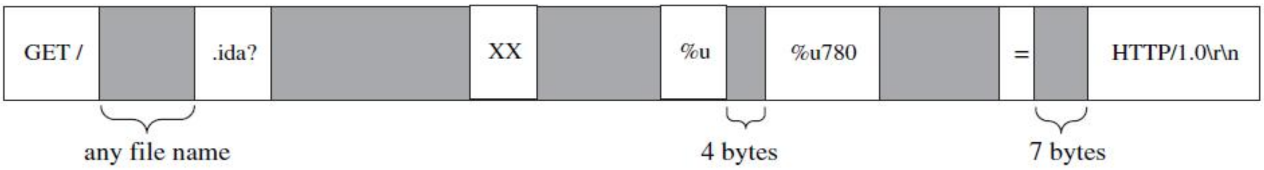

Fig. 9. One Version of Polymorphic Code Red II Worm.

6.2.3. Compared SRE with Other Signature Types.

It is natural to ask what is "*a more accurate*" and what is "*the most accurate*" signature for a polymorphic worm. Intuitively, "*the most accurate*" signature should be "*the most specific*," that is, in a balance of specific and general: specific—it contains as many features of the worm as possible so that it will not lead to false positives; general—it does not contain any useless or incorrect features of the worm so that it will not lead to false negatives.

Definition 3 (Containment, ⊵) Let X and Y be two SRE signatures, it says that Y contains X, denoted by X ⊵ Y, if L(X)⊆ L(Y). That is, the strings that match signature X must also match signature Y.

Definition 4 (More specific than,≺). Let X and Y be two SRE signatures. If X ⊵ Y and |X| > |Y|, it says that X is more specific than Y, or Y is more general than X, and this is denoted by X ≺Y. Here are some examples for Definition 3 and Definition 4:
“‘abc’*‘bcd’” ⊵ “‘ab’*‘cd’”, “‘ab’[2]‘cd’” ⊵ “‘ab’*‘cd’”; “‘aaa’*‘b’” ≺ “‘aa’*‘b’”, “‘aa’[3]‘c’” ≺ “‘aa’*‘c’”. Note that if a network flow f (as a special SRE signature with only one substring and without qualifiers) matches an SRE signature X, it also uses “f ⊵ X” to denote it. For instance, the flow “abcdef” matches “‘ab’*‘ef’” can be denoted by “abcdef” ⊵ "'ab'*'ef'".

Definition 5 (Signature of a polymorphic worm, <·). Given a polymorphic worm w, if all possible samples of w match an SRE signature X, then X is a signature of w, and this is denoted by w <· X.

Definition 6 (The most specific signature of a polymorphic worm, MSSig). Given a polymorphic worm w and its SRE signature X, if w <· X and for any other SRE signature X' such that both w <· X' and X ⊵ X' hold, then X is the most specific signature (MSSig) of w, and this is denoted by X = MSSig(w).

From the above definitions, now it can answer "what is a more accurate signature" for a polymorphic worm and are able to compare the accuracy of two signatures. Suppose that both X and Y are signatures of a polymorphic worm w (w <· X and w <· Y). Signature X is more accurate than Y if and only if $X \prec Y$. Given a number of samples of a polymorphic worm, the signature generation problem is formalized by the Problem bellow:

Problem (The most specific signature generation for a single polymorphic worm).
INPUT: s_1; . . . ; s_n are n samples of a polymorphic worm w.
OUTPUT: A signature X such that X =MSSig(w).
Given the polymorphic Code Red II worm in Figure 9, it can convert the generated signatures by previous NSG methods into the defined SRE signature format. Honeycomb [25] outputs “*‘.ida?’*”. Polygraph [9] outputs “‘GET / ’*‘.ida?’*‘XX’*‘%u’*‘%u780’*‘HTTP/1.0\r\n’”. Hamsa[10]outputs “‘GET/’*‘.ida?’*‘%u’*‘%u780’ *‘HTTP/1.0\r\n’*”.

If these SRE signatures are denoted by X1;X2, and X3, all of them are signatures of the worm according to Definition 5. However, none of them is the most specific signature according to Definition 6 because given an SRE signature Y = “‘GET /’*‘.ida?’*‘XX’*‘%u’[4]‘%u780’*‘= ’[1]‘HTTP/ 1.0\r\n’”, obviously, $Y \prec X_1$, $Y \prec X_2$, and $Y \prec X_3$. That is, Y is more specific than X_1, X_2, X_3. This implies that the signatures generated by the previous methods are not “the most accurate”.

7. The Central Alarm Procedure

When the worm has been detected by the behavior-base thread it send to central alarm procedure the source address (src.add) and source port (src.port) that the worm was use it. The central alarm procedure will register the address that have a worm and send an alarm message to all sensors to not accept any data from this src.add and src.port till the signature generation generate the signature of that worm.

When the signature generation thread finishes the signature, it gives it to center alarm procedure to send it to all sensors to update their database signature-base even to his signature-base database. And it motivates the signature bases to immediately scan in the new detecting worm for all computers it is responsible for.

8. The Result

The IFCA has a good result in detecting the worms and it is interested in finding the stealthy worms. But because the IFCA needs to repeat detecting all worms even it previously found it so it cause an overload on its algorithm runtime.

Suppose that there are N samples, each having a length L, the total time complexity of the SRE [23] is O (N^2L^2) O (NL^2). This proposal overcome the overload in both algorithms by using a hybrid solution to minimize the overall runtime; it used the IFCA to detect the unknown worms then it takes the worms samples that it collecting using traffic monitor thread to generate the signature that it will be used by signature base. By using signature base in detection the same worms if they appear again it will overcome the overload in IFCA.

Each of IFCA and the signature generation are built as threat to run in parallel, so it overcomes the time complexity in SRE algorithm.

In signature accuracy it is found that from section 6.2.3 that the tested Code Red II worm in Polygraph algorithm lost ‘‘=’’ and the distance restrictions of invariant parts and in Hamsa algorithm lost ‘‘=’’, the order and the distance restrictions of invariant parts in SRE signature, that it adopted in this design, is not missed.

In this design all data (inbound or outbound) are under monitoring using traffic-monitor thread (section 6.1) and it detects the known and unknown worms, so it overcomes the drawback of S. Behal el al.[8] design.

From the above it seen that the proposal design offer many features in detecting unknown and stealthy worms, the high speed in worm detecting, the most signature accuracy and finally the improvement in its performance by using a distributed collaborative scheme in it defenses against the worms attack.

9. Conclusion and Scope for Future Work

This paper uses a hybrid solution between the two detection algorithms; behavior based algorithm; to have it feature in detecting unknown worms and signature base algorithm to have its speed in detecting.

In behavior base detection it used IFCA algorithm which it faster in detecting the worm than other traditional Failure Connection algorithms. Also, the algorithm can detect the stealthy worms. And it used SRE algorithm in signature generation and it shows how it is the most accurate in finding the signature. A new method is built for collecting the data in a temporary list to have adequate samples for signature generation.

This design used distributed collaborative scheme based on the small-world network model to improve the system performance comparing to single-point defense scheme.

This paper proposed a design just for detecting the network worms. Future research will work on studying the protection of the network against worms and cascading failures using a link isolation strategy based on the quarantining of susceptible clusters in the network (modularity partitioning). This strategy aims to maximize the epidemic control while minimizing the impact on the clusters performance.

10. References

[1] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L.Zhang, and P. Barham, “Vigilante: End-to-end containment of Internet worms”, In Proc. of the 20th ACM Symp. On Operating Systems Principles (SOSP), Brighton, UK, Oct 2005.

[2] Computer worms information, http://virusall.com/computer%20worms/worms.php, latest site update at 18 June. 2011.

[3] Mohssen M. Z. E. Mohammed, H. Anthony Chan, Neco Ventura, Mohsin Hashim, Izzeldin Amin, Eihab Bashier, "Accurate Signature Generation for Polymorphic Worms using Principal Component Analysis", IEEE Globecom Workshops, pp. 1555-1560, 2010.

[4] C. C. Zou, N. Duffield, D. Towsley and W. Gong, “Adaptive Defense Against Various Network Attacks,” in IEEE Journal on Selected Areas in Communications, Vol. 24, No. 10, pp. 1877-1888, Oct. 2006.

[5] Miad Faezipour, Mehrdad Nourani and Sateesh Addepalli, "A Behavioral Analysis Engine for Network Traffic", 7th IEEE Consumer Communications and Networking Conference, PP. 1-5, 2010.

[6] X. Yang, J. Lu, Y. Zhu & P. Wang. “Simulation and Evaluation of a New Algorithm of Worm Detection and Containment”. Proceedings of the Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'06), Taiwan, Dec 2006, pp. 448-453.

[7] Mohammad M. Rasheed, Norita Md Norwawi, Osman Ghazali, and Mohammed M. Kadhum, "Intelligent Failure Connection Algorithm for Detecting Internet", IJCSNS International Journal of Computer Science and Network Security, Vol.9 No.5, May 2009.

[8] S. Behal and K. Kumar, "An experimental analysis for malware detection using extrusions", (ICCCT) 2'nd International Conference on Computer and Communication Technology, pp. 474-478, 2011.

[9] J. Newsome, B. Karp, and D. Song, “Polygraph: Automatically Generating Signatures for Polymorphic Worms,” Proc. 2005 IEEE Symp. Security and Privacy, pp. 226-241, 2005.

[10] Z. Li, M. Sanghi, Y. Chen, M.Y. Kao, and B. Chavez, “Hamsa: Fast Signature Generation for Zero-Day Polymorphic Worms with Provable Attack Resilience,” Proc. 2006 IEEE Symp. Security and Privacy, 2006.

[11] D. Schnackenberg, Holliday, H., Smith, R., Djahandari, K., Sterne, D, "Cooperative intrusion traceback and response architecture (CITRA)," in Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX), 2001, pp. 56-58.

[12] R. Janakiraman, Waldvogel, M., Zhang, Q., "Indra: A peer-to-peer approach to network intrusion detection and prevention," in Proceedings of 12th IEEE Workshops on Enabling Technologies,Infrastructure for Collaborative Enterprises (WETICE), Los Alamitos, 2003.

[13] C. V. Zhou, S. Karunasekera, and C. Leckie, "A Peer-to-Peer Collaborative Intrusion Detection System," in 13th IEEE International Conference on Networks, Jointly held with the IEEE 7th Malaysia International Conference on Communication, Kuala Lumpur, Malaysia, 2005, p. 6.

[14] C. V. Zhou, C. Leckie, and S. Karunasekera, "Decentralized multi-dimensional alert correlation for collaborative intrusion detection," J. Netw. Comput. Appl., vol. 32, pp. 1106-1123, 2009.

[15] C. V. L. Zhou, C. Karunasekera, and S. T. Peng, "A Self-Healing, Self-Protecting Collaborative Intrusion Detection Architecture to Trace-Back Fast-Flux

Phishing Domains "IEEE NOMS Workshops 2008, pp. 321 - 327 7-11 April 2008.

[16] Hao Chen, Yu Chen, "Comparison Study of Collaborative Strategies for Distributed Defense against Internet Worms based on Small-World Modeling", 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing, Oct. 9 – 12, 2010.

[17] J. Wu, S. Vangala, L. Gao, and K. Kwiat, “An Efficient Architecture and Algorithm for Detecting Worms with Various Scan Techniques", In Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS’04), February 2004.

[18] Weijiang Liu_y, Wenyu Quy, Gong Jianz and Li Keqiu, "A Novel Data Streaming Method Detecting Superpoints", IEEE Conference on Computer Communications Workshops, pp. 1042-1047, 2011.

[19] Pele li, Mehdi Salour, and Xiao Su, "A Survey of Internet Worm Detection And Containment", IEEE Communications Surveys; the electronic magazine of original peer-reviewed survey articles, Vol. 10, No.1, 1st quarter 2008.

[20] Wang Yun and Lu Dianjun, "An Efficient Threshold Signature Scheme Based on the Elliptic Curve Cryptosystem", 2nd IEEE International Conference on Information Management and Engineering, pp. 455-458, 2010.

[21] Yong Tang, Bin Xiao, and Xicheng Lu, "Signature Tree Generation for Polymorphic Worms", IEEE Transactions on Computers, Vol. 60, No. 4, April 2011.

[22] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection", Proc. ACM SIGCOMM, Vol. 36, pp. 339-350, 2006.

[23] Y. Tang, B. Xiao, and X. Lu, "Using a Bioinformatics Approach to Generate Accurate Exploit-Based Signatures for Polymorphic Worms", Computers & Security, Vol. 28 pp. 827-842, 2009.

[24] Y. Tang, X. Lu, and B. Xiao, "Generating Simplified Regular Expression Signatures for Polymorphic Worms", Proc. Fourth Int’l Conf. Autonomic and Trusted Computing (ATC ’07), 2007.

[25] C. Kreibich and J. Crowcroft, “Honeycomb—Creating Intrusion Detection Signatures Using Honeypots,” Proc. Second Workshop Hot Topics in Networks (Hotnets II), 2003.

تصميم نظام شبكة آمن ضد ديدان الانترنيت

رنا ضياء عبد الجبار
كلية هندسة المعلومات/ جامعة النهرين

الخلاصة

الديدان الفعالة شكلت تهديد رئيسي للانترنيت، وكثير من جهود البحوث ركز عليها. هذه الورقة مهتمة بدودة الانترنيت التي تنتشر عن طرق TCP، الذي يعتبر أغلبية حركة العمل على الانترنيت . هي تظهر طريقة تستخدم حل هجين بين خوارزميتي كشف، اعتماد كشف السلوك وأعتماد كشف التوقيع للحصول على مميزات كل منهما. إن هدف هذه الدراسة أن لها حل جيد من اكتشاف الدودة والدودة الشبحية بميزة السرعة. هذا الاقتراح صمم بمخطط تعاوني موزع مستند على نموذج الشبكة الصغير العالمي لتحسين أداء النظام بشكل فعال.