Future Generation Computer Systems 60 (2016) 67–77

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Investigations of automatic methods for detecting the polymorphic worms signatures

Shadi A. Aljawarneh[a,*], Raja A. Moftah[b], Abdelsalam M. Maatuk[b]

[a] *Software Engineering Department, Jordan University of Science and Technology, Irbid, Jordan*

[b] *Faculty of Information Technology, Benghazi University, Libya*

HIGHLIGHTS

- An Enhanced Contiguous Substring Rewarded (ECSR) algorithm is developed.
- The signature can produce a loss of vital information such as ignoring one byte token.
- The SRE needs to be updated and accurate when compared with autograph and polygraph methods.

ARTICLE INFO

Article history:
Received 25 November 2015
Received in revised form
26 December 2015
Accepted 30 January 2016
Available online 15 February 2016

Keywords:
Polymorphic worms
Simplified Regular Expression
Autograph
Polygraph
Software security
Software engineering

ABSTRACT

This paper investigates the current automatics methods used to generate efficient and accurate signatures to create countermeasures against attacks by polymorphic worms. These strategies include autograph, polygraph and Simplified Regular Expression (SRE). They rely on network-based signature detection and filtering content network traffic, as the signature generated by these methods can be read by Intrusion Prevention systems and firewalls. In this paper, we also present the architecture and evaluation of each method, and the implementation used as patterns by SRE mechanism to extract accurate signatures. Such implementation was accomplished through use of the Needleman–Wunsch algorithm, which was inadequate to manage the invariant parts and distances restrictions of the polymorphic worm. Consequently, an Enhanced Contiguous Substring Rewarded (ECSR) algorithm is developed to improve the result extraction from the Needleman–Wunsch algorithm and generate accurate signatures. The signature generation by SRE is found to be more accurate and efficient as it preserves all the important features of polymorphic worms. The evaluation results show that the signature contains conjunctions of tokens, or token subsequence can produce a loss of vital information such as ignoring one byte token or neglecting the restriction distances. Furthermore, the Simplified Regular Expression needs to be updated and accurate when compared with autograph and polygraph methods.

1. Introduction

The incidence of security threats has increased dramatically in recent years due to the direct connection of the computers to the Internet [1,2]. Worms are one of the important examples of smart design programs that can cause security threats, and can be defined as programs, which replicate themselves and exploit various types of vulnerabilities in the network hosts. These characteristics mean that worms can spread within minutes to a large number of computers, leading to network congestion ranging from file detection to denial of system services [3]. A polymorphic worm is a worm that can mutate its appearance with each infection, spreading through a network via a process of self-encryption or semantics-preserving code manipulation techniques. This affects the detection of this type of worm [4].

The most effective method of detecting worms is by signature-based detection, which is also known as content-based filtering. This signature can be used to inform researchers about the particular characteristics of the worms, for containment of the worm on infected computers. This signature can be generated by use of either exploit-based signatures, which illustrate the characteristics of an individual or a number of exploits, or vulnerability-based signatures, which illustrate the properties of individual vulnerability along with detection of all possible exploits employing this vulnerability [5]. However, Brumley et al. [6] stated that the vulnerability-based signature is generally

[*] Corresponding author.
E-mail address: saaljawarneh@just.edu.jo (S.A. Aljawarneh).

http://dx.doi.org/10.1016/j.future.2016.01.020
0167-739X/© 2016 Elsevier B.V. All rights reserved.

more efficient in making signatures if only the vulnerability-based is disclosed. On the other hand, Arce [7] argued that both types of signatures have equal abilities for the detection of worms in various applications, especially in Intrusion Detection Systems (IDSs). If malicious network traffic is detected by IDSs, an alarm may be raised. An IDS mainly functions by detecting threats that obstruct the remainder of the offending traffic and then preventing future attempts from succeeding.

The speed of worms generally outbreak in zero day and the polymorphic worm are approximately the same; they can mutate at every copy, in addition to keeping the original algorithm unchanged (invariant bytes). Within each exploit, the worms begin to modify the bytes by deleting the portions of some pieces of code, inserting or modifying some byte sequences thereby avoiding detection through simple signature matching techniques. However, the parts of code that remains unchanged can be used to characterise the signature of a polychromic worm [8]. For this purpose, security experts have developed a number of automatic and faster methods to derive more accurate and efficient signatures for worms. Firewalls or Intrusion Prevention Systems can ultimately read generating a signature automatically to quickly containment the worm spread. These automatic methods can be used to extract good quality signatures which preserve all invariant bytes and restriction distances which make identification and preventing worm easier [9].

This paper investigates the characteristics of polymorphic worms and then explains the most common forms of pattern-based detection, such as Autograph, Polygraph and Simplified Regular Expression (SRE). In addition, we investigate the enhanced algorithm in terms of accuracy only. Autograph is the earliest system, which is used to automatically generate a signature for a single string based on matches with this string. Meanwhile, Polygraph is a token-based system that chooses a set of tokens which have a high exposure to the low false positive and the suspicious pool reaction to the ordinary traffic pool [10]. The SRE uses other automatic approaches, along with multiple sequence alignment to discover polymorphic worm signatures by utilising the real polymorphic worm application level (as samples) and evaluating the accuracy of the signature arising from this approach [5].

This paper will be structured as follows: Section 2 introduces the anatomy of polymorphic worms and signature automatic systems. Section 3 provides characterisation of automatic methods that can be used to detect the signatures of polymorphic worms. Section 4 describes the proposed methodology for polymorphic worms utilising sequence alignment. Section 5 presents the results, a discussion of how these fit within the context of the paper and the limitation of the approach used. Section 6 includes a conclusion and future work.

2. Anatomy of polymorphic worms and signature automatic systems

The aim of this section is to: (i) provide a general introduction to the concept of polymorphic worms and polymorphic techniques; (ii) investigate the properties of the bodies of polymorphic worms; (iii) introduce the concept of worm signatures and various methods of detecting these signatures; and (iv) summarise the advantages and disadvantages of existing polymorphic worm detection techniques and outline the concept of pattern based worm detection.

2.1. Polymorphic worm

Noh et al. [11] stated that most of the Internet worms cause damage to networks through consumption of bandwidth that threatens the security of Internet infrastructures and the information of the platform. This threat has become increasingly likely, with the development of advanced worms such as polymorphic worms that can change their program code without human interaction by replicating themselves, enabling them to exploit operating systems and software vulnerabilities in order to contaminate a system [12]. Xiao et al. [13] explain that the propagation of a worm includes three stages:

- Target finding: each copy decides on the next victim by IP address.
- Worm transferring: after finding a target, the worm sends itself to the victim device.
- Infection stage: when the worm's code has transferred to victim machine, the code will be executed.

2.2. Polymorphic techniques

Polymorphic engines have published shellcode generators with various techniques including ADMmutate, Clet and TAPiON. These techniques have been used to write shellcode of polymorphic, which include Garbage, register shuffling, equivalent code substitution and encoding (encryption/decryption) to evade worm detection. As mentioned in 2.1, the polymorphic mechanism leads to confusion of worm detection approaches by concealing the worm's payload through the use of encoding techniques to write polymorphic shellcodes. If the worm mutates, its payload will generate a different form from its copy but still have the same function. Polymorphic worms commonly include four parts: Decryption Routine, Decryption Key, Encrypted WormCode and Exploit Code [12].

A polymorphic worm exploits an initial vulnerability and then decrypts the encrypted worm code utilising the decryption routine along with the decrypted key [12]. Various keys of encryption and decryption are applied to encrypt the worm code for each worm sample. Decryption Key along with Encrypted Worm Code models vary for each worm sample, while the Decryption Routine and Exploit code models stay unchanged [12]. So, obfuscation mechanisms are used by each worm sample to formulate different Decryption Routine models for each sample, which creates an Exploit Code in the unchanged part of the polymorphic worm code that is a source of high false positives if individually utilised to detect the worm. In addition, Bayoglu et al. [12] argued that encryption does not include the full code of the worm, as that would make the code inoperative. Each worm therefore has a part of the code that exists for the purposes of exploiting prospective victims. The unencrypted part is used afterwards to branch the implementation cycle to the decryption routine along with the initial code.

2.3. Polymorphic body

Tang et al. [14] explained that the polymorphic worm sample (infection flow) contains a string sequence. These strings include invariant bytes and wildcard bytes. Invariant bytes contain fixed values and should be present in each worm sample in order to ensure that the infection is successful. Wild card bytes change their values for each diverse worm sample. For instance, a polymorphic Code Red II worm is presented in Fig. 1, which has a sequence of seven invariant contents: "GET", ".ida?", "XX", "%u", "%u780", "=", along with "HTTP/1.0 r n". So, security professional people attempt to extract the invariant contents of polymorphic worm as a signature. Invariant bytes in a worm flow create a number of invariant contents that are essential to successful worm infection. In other words, the invariant content, "%u 7801" is 4 bytes after "%u", which illustrates the number of characters between two substrings. These distance restrictions are important for the exploitation of a vulnerable server [12].

Note that, most approaches also do not take into account the all distance restriction in the Code Red II. Despite this, each one-byte invariant components and distance restriction are crucial in worm detection.

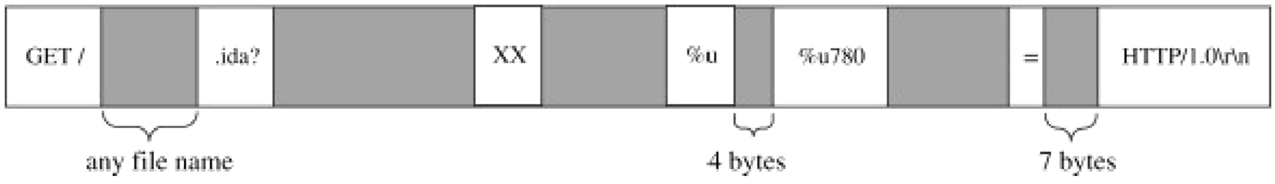


Fig. 1. Polymorphic Code Red II worm. Shaded content exemplifies wildcard bytes, unshaded content exemplifies invariant bytes [5].

2.4. Worm detection signature

Worm detection signature is an approach used to find activities of polymorphic worms. The key idea of an extracted signature is to discover match invariant substrings or sequence similarities in all difference aspects of a payload. Bayoglu et al. [12] claimed that these methods dealing with polymorphic worms can be further classified into content-based detection, behaviour based detection and pattern based detection. Content based polymorphic worm detection systems use the worm content to generate information to facilitate matches with the worm. Behaviour based approaches are concerned with the behaviour of the worm in the flow of the network along with system activities. Nevertheless, Xiao et al. [13] state that worm detection techniques should be classified into two schemas: signature-based and anomaly-based. Currently, automatic signature generation techniques can be associated between two detection schemes: Signature-Based Worm Detection and Anomaly-Based Worm Detection. Currently, automatic signature generation has become an important issue and numbers of techniques have been projected. These systems are classified into two subtypes Host-based and Network-based [13]: Host-based Signature Generation (HSG) and Network-based Signature Generation (NSG). In this paper, we focus on the pattern based detection since the proposed algorithm is based on the kind of detection.

3. Pattern based detection

This section aims to: (i) Demonstrate the Autograph, which is the early system, designed to automatically generate a signature of polymorphic worm. This is based on a single string matching and presently includes the architecture and evaluation. (ii) Investigate the next design which is a Polygraph used for detecting a polymorphic worms signature. Unlike autograph approaches, it is based on matching multiple invariant substrings. Also, this investigation includes most characteristics of this approach and an evaluation. (iii) Currently, the security community exploits the SRE system based on multiple-sequence alignment algorithms. To resolve the problems raised from systems realised. The signature generation of this system does not depend on well-classified worm flow pool as with the two approaches in. (iv) Summarise the SRE, methods and algorithms, followed by the security experts to generate an accurate signature for the polymorphic worm. This detection technique investigates and provides multiple sequence alignment based approach to generate an SRE signature.

3.1. Autograph

Autograph is one of an early pattern, which was constructed to automatically generate signatures for novel Internet worms. This approach is used to generate signatures that demonstrate high true positive rate (high sensitivity) and low false positive rate (high specificity), using content based filtering. Kim et al. [15] restricted their analysis to worms that propagate over TCP transport. The signature is a tuple, including: IP protocol number, destination port number, along with byte sequence. The content is based on filtering and considers the payload in the network stream, when it matches the byte sequences in the signature by utilising the same IP protocol destined for the destination port number, and is then classified as a worm.

All traffic crossing the Demilitarised Zone (DMZ) inputs to an Autograph monitor and outputs a list of the worm signatures. This system comprises of two phases: suspicious flow selection and signature generation. The suspicious flow selection is used to classify the network stream as a suspicious flow pool (malicious) along with non-suspicious flow pool (innocuous). The signature generation is found in the veracity of how worms propagate to exploit the software vulnerability and to execute; thus all instances consist of one or an additional common byte sequence. The worm spreads in a bulky number, so the amount of common content block is high. The suspicious flow is used to generate a signature by dividing it into smaller content blocks, and the number of suspicious flows in which every block's content arises is counted. This content block is ranked according as its prevalence, with a higher prevalence achieving a higher count. The commonly occurring content block is utilised as the signature [16].

Note that the autographing repetitively the process for the majority of prevalent content block is chosen as signature and then all flows found in this content block are eliminated. This process is repetitive, with remaining flows continuing until the fraction of the entire flows in the pool has been enclosed. Autograph will report the set of chosen signatures in Bro's signature feature at the end of the process.

The quality of the signature generated by this system depends upon the size of the content block. When the size of the content block has been made small, it is in most cases autograph signature generating. Then, the suspicious flow after that passes through the generator signatures, which automatically generates various classes of signatures. The authors provide more details about the evolution of the generation of signatures in their paper through Autograph, and they also describe their prototype implementation. However, the client can easily guess from the performed experiments that a prototype exists. Initially, they have explored the effect of the content size on the quality of signature generation. In their experiment, Autograph is supplied with packets traces from DMZ from two different research labs, each of the research's 29 IP address include the complete packet payload. For computing Autograph's true positive rate, the identical trace was experimented with initially by applying Bro with well-known signatures—the scanning-based HTTP worms Code-Red, Code-RedII, Nimda, and this was followed by applying Autograph's signatures [13,15,16].

3.2. Polygraph

Polygraph is a pattern of detection constructed to automatically generate signatures for polymorphic worms. This pattern is based on how all the instances of the polymorphic worm integrate multiple invariant substrings [17]; in contrast to the autograph approach, which is based on matching a single substring. Multiple substrings are applied in the polygraph approach, which makes it perfect for extracting a signature for the polymorphic that leans towards altering the sequence of the byte flow in each sample.

However, this mechanism is based on the content of the payload of the worms as in the autograph mechanism.

Newsome et al. [17] concluded that the single substring signature is able to robustly match the polymorphic worm through a low false positive, along with a low false negative. Also, they have created the signatures from tokens (substrings) and classified the signatures for polymorphic worm signatures into:

- Conjunction signatures: A signature, which consists of a set of tokens, and the signature match a payload when entire tokens in the set are found in it unordered. This kind of signature matches the multiple invariant tokens provided in a polymorphic worm payload. Here, matching multiple tokens is more exact than matching one of these tokens only.
- Token-subsequence signature: A signature that consists of an ordered set of tokens. A stream matches a token-subsequence signature when the flow has the same sequence of tokens in the signature with identical ordering. Signatures of this kind are simply expressed as regular expressions and allowed to be used in current IDSs. For the equivalent set of tokens, a token subsequence signature is more precise than a conjunction signature due to the presence of order constraints in a token-subsequence signature.
- Bayes signature: Bayes signature distinguishes a worm from an innocuous stream by probabilistic matching, distinct from conjunction and token subsequence signature, which are based on the exact prototype match. Signatures consist of a set of tokens, which are each associated with a score and overall threshold. It provides probabilistic matching for a given flow by computing the probability that the flow is a worm by the scores of the tokens provided in the flow. When the probability of the result is more, the threshold of the flow is classified as a worm. The advantage of Bayes signature is that it allows learning from suspicious stream pools that include patterns from unrelated worms and innocuous flow.

Polygraph is performed for experimental evolution while considering various scenarios. These are: the flow pool has flows of only one worm, the flow pool has flows of one worm along with innocuous flows, and the flow pool has flows of multiple worms as well as innocuous flows. The authors [17] stated that a polygraph evaluated by several networks traces input. The authors detected that the quality of the signatures generated depends on the number of worm patterns provided in the trace. The initial consideration is for one kind of worm and some innocuous flow outcomes from the limitation of the flow classifier, and then a more practical case where a suspicious case could have flowed from various misclassified innocuous worms.

3.3. *Generating simplified regular expression signature*

Recently, this mechanism is circuital to automatically creating an effective and accurate signature to protect against the polymorphic worm. Simplified Regular Expression (SRE) is extended from Polygraph's token-subsequence signature category. The authors' assert the previous mechanisms, autograph and polygraph, use of conjunction or subsequence token can lose essential information such as ignoring or neglecting one byte token of the distance in the sequential token. The authors propose the Simplified Regular Expression (SRE) signature, and provide their mechanisms which are based on the multiple sequences alignment algorithms. This method is extended from the pairwise sequence alignment algorithm that encourages contiguous substring (token) extraction and it can keep the distance of invariant content parts as well as sustain the wildcard string alignment in generated SRE signatures. In addition, the authors found that the signature generated using SRE can express the distance information for byte invariant content which is important and valuable for detecting the polymorphic worm [5].

In this scenario, they verify that for all their experiments on the several approaches of detection of polymorphic worm signatures, compared with present network-based signature generation systems (NSGs) approaches, the SRE mechanism produced signatures with more precise and perfect matches to polymorphic worms. The previous approaches, whether autograph which generates a single contiguous byte string signature, or polygraph which generates token-based signature (bytes sequence), occur in significant numbers of suspicious flow pools. The autograph approach proves that the signature is not effective to match polymorphic worms. Polygraph proves that the signature depends on a token-based signature and it can lose vital information associated with any sequential token such as distances [14].

An SRE signature is a simplified version of the regular expression, where there are just three reiterated qualifiers. These are: "*", "[$k1, K2$]" and "[k]". The authors have substitute the ".*" within regular expression using "*" to characterise an arbitrary string also it is including zero-length string. In addition, to characterise any string with a length from $k1$ to $K2$, they replaced ". {$k1, k2$}" by using "[$k1$, $k2$]". As well as, replacing. "{K}" by using "[k]" to characterise a string contain of k number of arbitrary character. For instance; "'one'*'two'[3]'three'[2, 5]" is an SRE signature which is equivalent to the regular expression "one.*two.{3}three.{2, 5}" [14].

In this case, the SRE signature is still a regular expression, but contains less syntax rules. In a restricted sense, the set of SRE signatures are subsets of the set of regular expressions. Thus, an arbitrary SRE signature satisfies the description of regular expression and it can be used for detection in any IDS which supports regular expression [5].

4. Methodology and implementation

We have developed the Enhanced Contiguous Substring Rewarded (ECSR) algorithm to improve the result extraction from the Needleman–Wunsch algorithm and generate accurate signatures. Our aim is to implement a program to detect accurate and efficient signature for a single polymorphic worm. The implementation process includes: (i) a brief introduction related to construction of the SRE mechanism; and (ii) execution of SRE procedures to generate signatures for single polymorphic worms.

4.1. *SRE mechanism design*

The first step in implementing the SRE mechanism is to obtain the network flows as patterns from the polymorphic worm and convert these into character sequences, which are indicated to the worm samples. These samples were used to verify the performance of the implementation provided by the authors. The samples provided that were exploited in this study contain from two sequences, (oxnxexzxtwox) and (ytwoyoneyz), which are used to generate signatures for a single polymorphic worm. The SRE process is used to detect invariant bytes of string in the network flow and extract wildcards for the sequences given. All the results will be achieved using the bioinformatics approach, which is a sequence alignment that will be carried out by the Needleman/Wunsch algorithm [18]. Needleman–Wunsch is a model of dynamic programming based on a divide-and-conquer strategy [5].

This algorithm performs alignment on two sequences as pairwise characters of the sequences by maximising a similarity score. Pairwise alignment describes when the characters are written by one sequence on the top of another [5]. The Needleman–Wunsch alignment process includes three steps [18]:

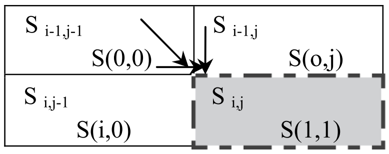


Fig. 2. The value of cell $S(1, 1)$ resulting from $S(0, 0)$, $S(i, 0)$ and $S(0, j)$.

(a) Initialisation step: initialisation matrix score is used to compare characters between two sequences;
(b) Matrix fill step: filling matrix values to select the maximum number;
(c) Traceback step: the path is followed back to the beginning of the matrix, where each branch of the traceback will represent an optimal alignment. These stages will be described in detail in the following section. Initially, a manual process is used to initialise, fill the matrix and extract the maximum score [19]. After that the study will use Perl script to deal with the sequences given to extract the optimal alignment score, which is anticipated will produce various outputs. This will lead to analysis of each result and development of the performance of SRE using the CRS algorithm, which has been converted in this study to Java script in order to obtain an accurate and efficient signature for a single polymorphic worm.

4.2. Implementation of the SRE

The global alignment process for two sequences: Seq1 = oxnxexzxtwox and Seq2 = ytwoyoneyz. Using a substation matrix S {where $S(i, j)$} is the score to align i and j. $\{i = 0, 1, 2 \dots n,\ j = 0, 1, 2, \dots, n\}$ with a linear gap penalty $\{w\}$. There are three ways to characterise the alignment between two sequences [20], as follows:

– S_i and S_j are aligned, in which case, $S_{i,j} = \{S_{i-1,j-1} + M_{i,j}\}$.
– S_i is aligned with a gap, in which case, $S_{i,\,j} = \{S_{i-1,j} + W\}$.
– S_j is aligned with a gap, in which case, $S_{i,j} = \{S_{i,j-1} + W\}$

where $M(i, j) = S(i, j)$, the score to align S_i and S_j, if there is match between the bases the $M(i, j) = 1$. If there is no match between the bases, the $M(i, j) = 0$ and W represent the gap penalty $= -1$. While $S(i, j)$ is by definition the maximum score as formula 1 [20]. Three steps include:

$$S_{i,j} = \text{Max}[\{S_{i-1,j-1} + M_{i,j}\};\ \{S_{i-1,j} + W\};\ \{S_{i,j-1} + w\}]. \quad (1)$$

- Initialisation the matrix step: This paper also explains the way in which the Needleman–Wunsch algorithm operates using a manual system in order to ensure understanding of how this algorithm is constructed. The matrix $S(i, j)$ is created with $i + 1$ row and $j + 1$ column, where i and j correspond to write the bases of one sequence as the initial row of the matrix, and bases of another sequence as the initial column of the matrix. The algorithm works by aligning two sequences Seq1 and Seq2 using the formula (1) to fill the entire matrix, working from the upper left-hand corner and finding the maximum score $S(i, j)$ for each position in the matrix. Fig. 2 shows how each cell value computes [21] and as shown in Fig. 3.

Diagonally $S(i - 1, j - 1)$ indicates to $S(0, 0) = 0$ {score to alignment 0 elements from every sequence}. Vertically, $S(i - 1, j)$ indicates to $S(i, 0) = i \times w$ {score to alignment the first i elements from S_i with 0 elements from j with gap penalty $w = -1$}. Horizontally, $S(i, j - 1)$ indicates to $S(0, j) = j \times w$ score to alignment of 0 elements from S_j with 0 elements from S_i with a gap penalty of $w = -1$.

-Matrix fill step:

After finding all parameters $S(i-1, j-1)$, $(S_{i-1,j})$ and $(i, j-1)$, the process to fill matrix will be started until the end of this matrix by use of formula 3. Last score will determine the best score for global alignment of i and j. Note that in Fig. 4 for each $S(i, j)$, it is essential to keep pointer back to the previous score from which parameters were derived [21].

-Traceback step:

Traceback captures the current last cell value and observes the neighbour cells, which can direct predecessors. The traceback then begins from the lower right corner of the matrix [21] as shown in Fig. 4.

By following these rules, the matrix will be constructed. For example in Fig. 3, after initialising the matrix using a grey colour, the fill matrix occurs at the first cell by following the formula (1).

$$S_{i,j} = \text{Max}[\{S_{i-1,j-1} + M_{i,j}\};\ \{S_{i-1,j} + W\};\ \{S_{i,j-1} + w\}] \quad (1)$$

$$\begin{aligned} S(1,1) &= \text{Max}[\{S(0,0) + M(O,Y)\};\ \{S(0,1) +\ (-1)\};\\ &\quad \{S(1,0) +\ (-1)\}]\\ &= \text{Max}[\{0+0\}; \{-1-1\}; \{-1-1\}]\\ &= \text{Max}[0]. \end{aligned}$$

So $S(1, 1) = 0$, and should keep a pointer back to S(0,0) as this is shown in cell $S(1, 1)$ with colour dark grey and pointer with colour blue. Suppose the next cell requires filling is $S(3, 1)$, then according to formula (3):

$$\begin{aligned} S(3,1) &= \text{Max}[\{S(2,0) + M(O,W)\};\ \{S(3,0) + (-1)\};\\ &\quad \{S(2,1) + (-1)\}]\\ &= \text{Max}[\{-2+0\}; \{-3-1\}; \{-1-1\}]\\ &= \text{Max}[-2]. \end{aligned}$$

So $S(3, 1) = -2$, and also keeps a pointer back to $S(2, 0)$ and $(2, 1)$ as shown in cell $S(3, 1)$, which has been coloured dark grey and with pointers in blue.

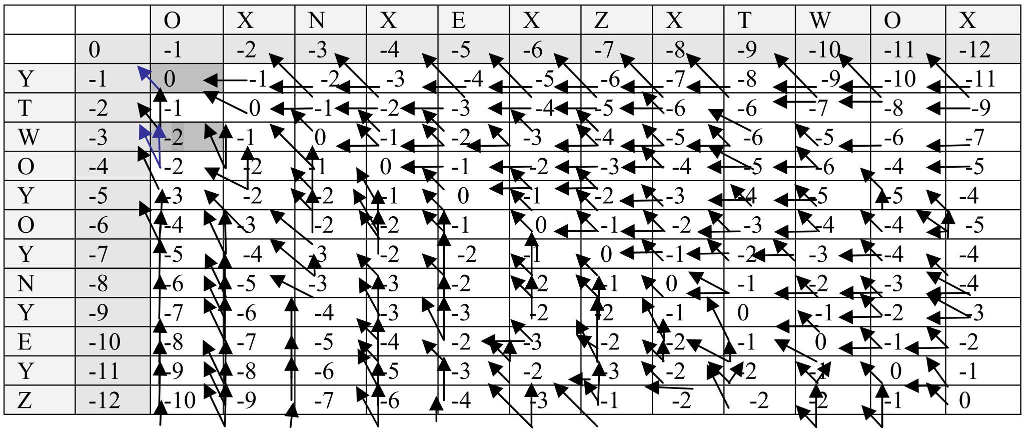
In following to the path back to the beginning of the matrix, the branch of the traceback represents an optimal alignment [22] with a maximum score equal to zero. Fig. 5 shows the traceback with blue arrows and maximum score cells in dark grey. In addition, each cell has aligned with another cell, which will take the same colour as Fig. 4; while Fig. 5 illustrates the final alignment sequences by using Perl script (see Fig. 6). So from the manual application of the Needleman/Wunsch generate characters without any match among them with optimal scores of zero. Needleman et al. [18] assigned that the characters substitution matrices may be adjusted in different ways to compensate for the character compositions of the sequences. This is attempted by inserting a gap penalty to improve the significance of the maximum match (see Fig. 7).

The author exchanged the order of two sequences using Perl script, but the study still obtains the same result mismatched alignment with a maximum score equalling zero, as shown in Fig. 8. This paper also seeks to generate accurate signatures of the polymorphic worm that should contain contiguous substrings and preserve wildcards. To repeat the comparison process between two sequences given, using gaps since attempts to obtain similar characters between them. As shown in Fig. 9, the Needleman–Wunsch algorithm resulted in alignments that consist of four parts (‘o’, ‘n’, ‘e’, ‘z’). Then this algorithm also generates numerous trivial or useless parts that will prevent the discovery of contiguous invariant content and wildcards. The formula (2) produces maximum score $= -6$.

$$\text{SC}\ (i, j) = -6\ (1 \times 4 + 0 \times 3 + 10 \times -1)$$

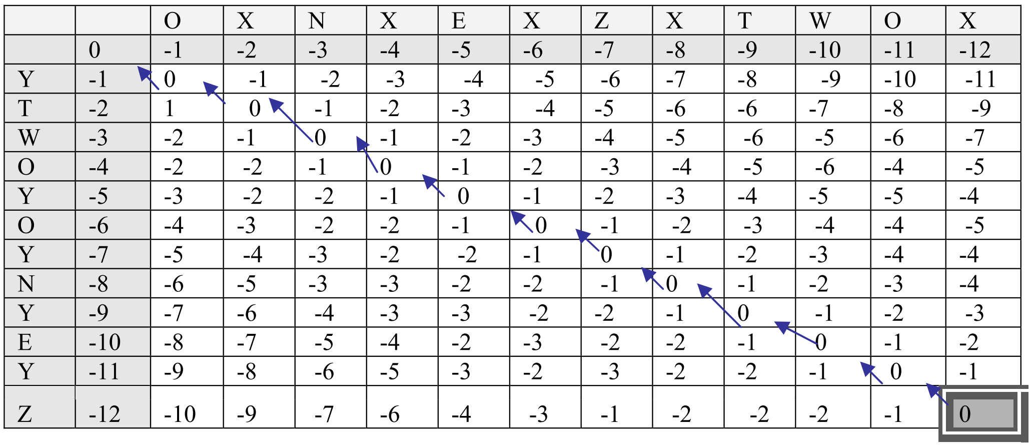
where W_d indicates the score to character mismatch, kgaps indicates the number of gaps, δ indicates to the penalty score for a gap. Tang et al. [5] indicate that setting $W_m = 1, W_d = 0, \delta = -1$.

The results from Fig. 8 are clearly better in terms of alignment as the substring ‘two’ is semantically meaningful because it is likely



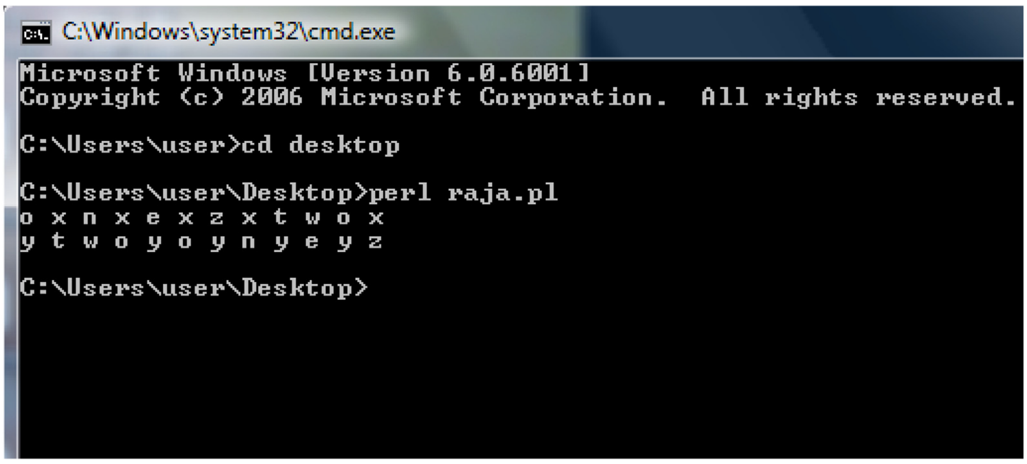
		O	X	N	X	E	X	Z	X	T	W	O	X
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
Y	-1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
T	-2	-1	0	-1	-2	-3	-4	-5	-6	-6	-7	-8	-9
W	-3	-2	-1	0	-1	-2	-3	-4	-5	-6	-5	-6	-7
O	-4	-2	-2	-1	0	-1	-2	-3	-4	-5	-6	-4	-5
Y	-5	-3	-2	-2	-1	0	-1	-2	-3	-4	-5	-5	-4
O	-6	-4	-3	-2	-2	-1	0	-1	-2	-3	-4	-4	-5
Y	-7	-5	-4	-3	-2	-2	-1	0	-1	-2	-3	-4	-4
N	-8	-6	-5	-3	-3	-2	-2	-1	0	-1	-2	-3	-4
Y	-9	-7	-6	-4	-3	-3	-2	-2	-1	0	-1	-2	-3
E	-10	-8	-7	-5	-4	-2	-3	-2	-2	-1	0	-1	-2
Y	-11	-9	-8	-6	-5	-3	-2	-3	-2	-2	-1	0	-1
Z	-12	-10	-9	-7	-6	-4	-3	-1	-2	-2	-2	-1	0

Fig. 3. Initialisation and filling matrix $S(i, j)$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



		O	X	N	X	E	X	Z	X	T	W	O	X
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
Y	-1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11
T	-2	1	0	-1	-2	-3	-4	-5	-6	-6	-7	-8	-9
W	-3	-2	-1	0	-1	-2	-3	-4	-5	-6	-5	-6	-7
O	-4	-2	-2	-1	0	-1	-2	-3	-4	-5	-6	-4	-5
Y	-5	-3	-2	-2	-1	0	-1	-2	-3	-4	-5	-5	-4
O	-6	-4	-3	-2	-2	-1	0	-1	-2	-3	-4	-4	-5
Y	-7	-5	-4	-3	-2	-2	-1	0	-1	-2	-3	-4	-4
N	-8	-6	-5	-3	-3	-2	-2	-1	0	-1	-2	-3	-4
Y	-9	-7	-6	-4	-3	-3	-2	-2	-1	0	-1	-2	-3
E	-10	-8	-7	-5	-4	-2	-3	-2	-2	-1	0	-1	-2
Y	-11	-9	-8	-6	-5	-3	-2	-3	-2	-2	-1	0	-1
Z	-12	-10	-9	-7	-6	-4	-3	-1	-2	-2	-2	-1	0

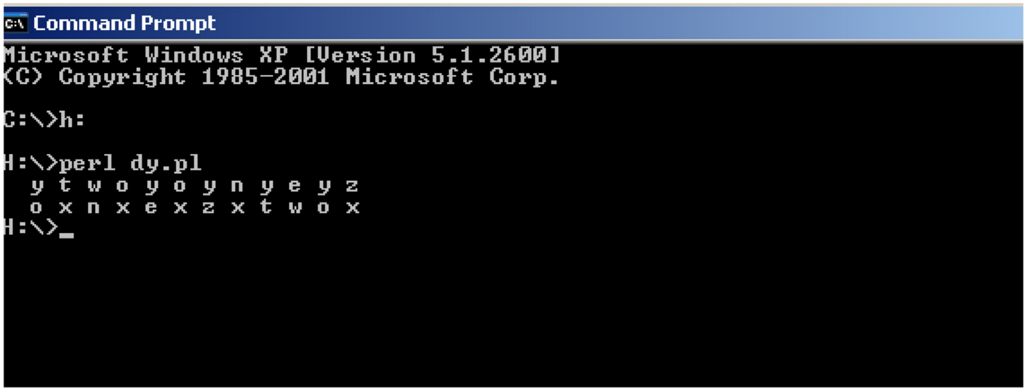
Fig. 4. Maximum score and the tracesback branch. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd desktop

C:\Users\user\Desktop>perl raja.pl
o x n x e x z x t w o x
y t w o y o y n y e y z

C:\Users\user\Desktop>
```

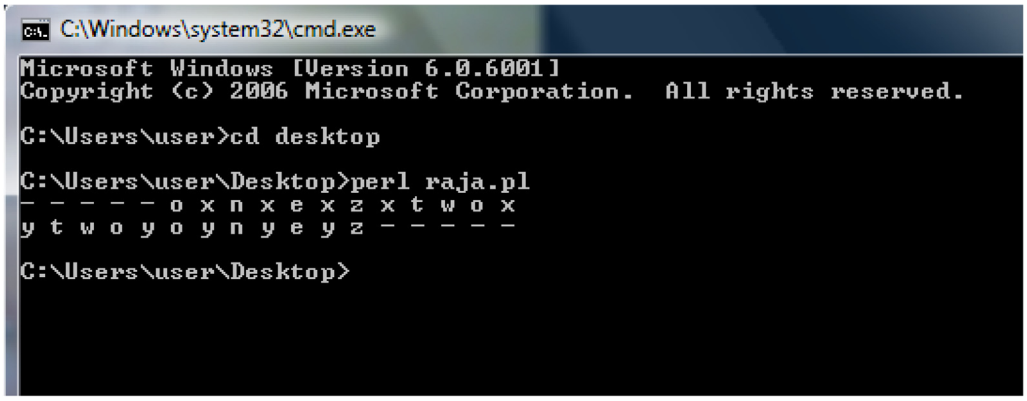
Fig. 5. Final alignment of the samples by using Needleman–Wunsch.

to be an invariant part of polymorphic worm (*******?two*******). According to formula (2): SC $(i, j) = -11\ (1\times3+0\times2+-1-\times14)$, but as a signature for polymorphic worm this is still not accurate enough because the restrict distance variable is not clear. After a number of experiments, this study found that it is relatively unimportant whether the parameters have been adjusted; the Needleman–Wunsch algorithm continually leans to create a great number of parts and then loses an amount of invariant content in polymorphic worms and distance restrictions, meaning that it is still not clear enough. Additionally, the maximum score resulting from the manual application of the algorithm is equal to zero, after which the maximum score accumulates in various values −6 and −11. The highest value is not account matching between two sequences, whereas the small one (−11) gives better alignment. The Needleman–Wunsch algorithm failed to generate contiguous substrings which are essential in worm traffic detection. Limitation in using this algorithm maximises the total number of matches as an alternative to match consecutive substring. To overcome this matter the CSR algorithm is used, which is based on matching strings for the longest common substring (LCS). The LCS algorithm



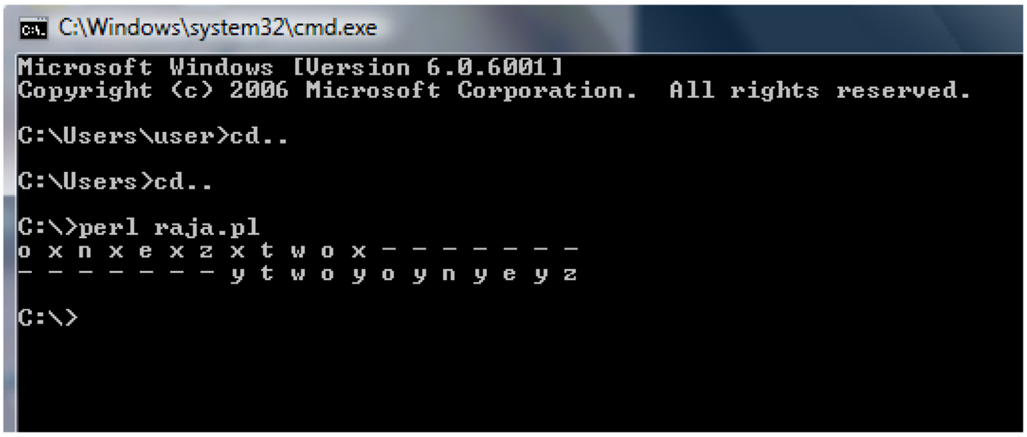
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>h:

H:\>perl dy.pl
  y t w o y o y n y e y z
  o x n x e x z x t w o x
H:\>_
```

Fig. 6. Final sequence after change the order of the samples by using Needleman–Wunsch.



```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd desktop

C:\Users\user\Desktop>perl raja.pl
- - - - - o x n x e x z x t w o x
y t w o y o y n y e y z - - - - -

C:\Users\user\Desktop>
```

Fig. 7. Perl script result alignment with four-character match between two sequences.



```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd..

C:\Users>cd..

C:\>perl raja.pl
o x n x e x z x t w o x - - - - - - -
- - - - - - - y t w o y o y n y e y z

C:\>
```

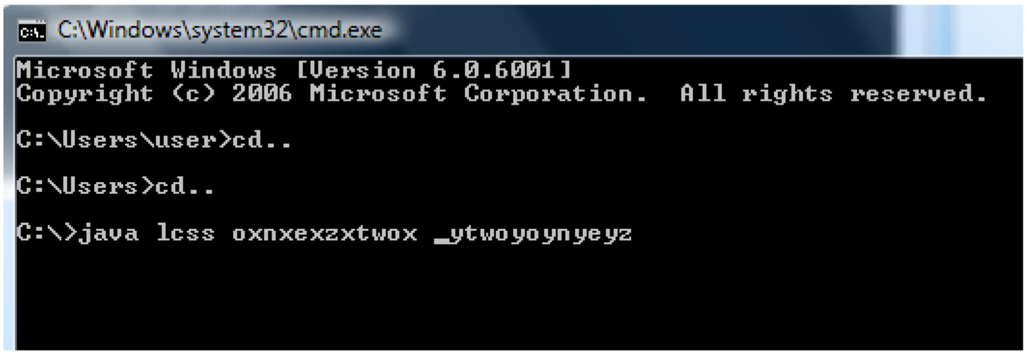
Fig. 8. Perl script result alignment with four character match between two sequences.

accumulates the longest shared byte sequences across pairs of connections.

4.3. The CSR algorithm

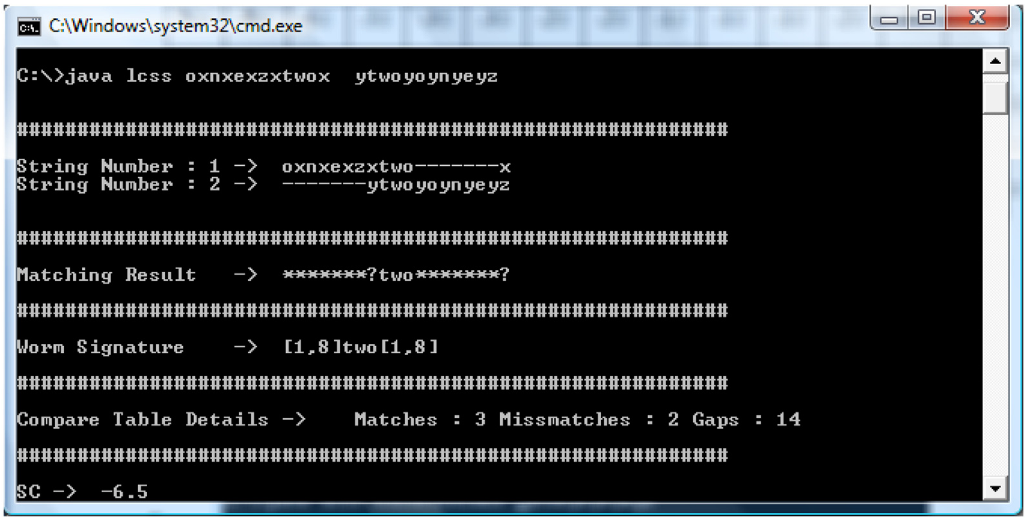
The CSR algorithm gathers the longest shared byte sequences across pairs of connections rather attempting to match the optimal score between two sequences. Tang et al. [5] stated that the CSR is extended from the Needleman–Wunsch algorithm. CSR exploits the design of the optimal alignment, which maximises the similarity score in formula (2). The maximum similarity score is performed through iteratively computing two matrices: the score matrix F with the traceback matrix PTR. Providing two sequences X and Y, assume $X\ (1\ldots i)$ and $Y\ (1\ldots j)$ are prefix subsequences of X and Y, Fi, j and supply the maximum similarity score for the two prefix subsequences. The PTR i, j records the traceback number to obtain the matching optimal alignment with the maximum score Fi, j. The optimal alignment of X along with Y will accumulate within FN, M and PTR N, M after that F and PTR are accomplished.

This algorithm includes three stages, in the same way as Needleman/Wunsch Algorithms. The first step is to initialise the iterative matrix F and PTR. The score matrix F along with the traceback matrix PTR is iteratively computed. From line 10 the CSR algorithm illustrated that Fi, j is calculated from $Fi-1, j-1$, $Fi-1, j$, and $Fi, j-1$ in three cases respectively. In a restricted sense, the optimal alignment of subsequence $X\ (1\ldots i)$ along with $Y\ (1\ldots j)$ is designed within three cases: Case 1, optimal alignment of subsequence $X\ (1\ldots i-1)$ and $Y\ (1\ldots j-1)$, $X\ (i)$ to $Y\ (j)$; Case



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\user>cd..

C:\Users>cd..

C:\>java lcss oxnxexzxtwox _ytwoyoynyeyz
```

Fig. 9. Run Java script of CSR algorithm.



```
C:\Windows\system32\cmd.exe
C:\>java lcss oxnxexzxtwox  ytwoyoynyeyz

#################################################################

String Number : 1 ->  oxnxexzxtwo-------x
String Number : 2 ->  -------ytwoyoynyeyz

#################################################################

Matching Result   ->  *******?two*******?

#################################################################

Worm Signature    ->  [1,8]two[1,8]

#################################################################

Compare Table Details ->    Matches : 3 Missmatches : 2 Gaps : 14

#################################################################

SC ->  -6.5
```

Fig. 10. Final results represent the polymorphic worm signature.

2, optimal alignment of subsequence $X\ (1\dots i-1)$ and $Y\ (1\dots j)$, $X\ (i)$ to a gap; Case 3, optimal alignment of subsequence $X\ (1..i)$ and $Y\ (1..j-1)$, $Y\ (j)$ to a gap. Tang et al. [14] have used the enc() in Case 2, utility to reward consecutive matches, which is the core range of their CSR algorithm from the Needleman–Wunsch algorithm. Finally, the alignment is reduced from the traceback matrix PTR.

In this paper the CSR algorithm was converted by using java script. This was done after typing the run command, as illustrated in Fig. 10.

This CSR is based on iterative LCS computed in its process until the result is obtained. To understand the steps in the CSR and java script with sequences, the following explanation is needed. Seq1 (string1) $\rightarrow$ oxnxexzxtwox, and Seq2 (string2) $\rightarrow$ ytwoyoynyeyz. The first step is to compute the longest common substring (LCS) which is LCS $\rightarrow$ "two". The second step will test the length of LSC is =0, the answer to which here is no. the third step is to divide the strings into two parts, left and right of LCS "two". Substring to Left of LCS in String 1 = "oxnxexzx" and on the right "*x*". Substring to left of LCS in String 2 = "*y*" and on the right "yoynyeyz". The fourth step is to compare the Left Substrings of both the strings for LCS i.e. "oxnxexzx" and "*y*", and to compare right substrings of both the strings for LCS i.e. "*x*" and "yoynyeyz". "oxnxexzx" and "*y*" there is no common substring. The length of "oxnxexzx" is 8 whereas length of "*y*" is 1. So to match the sequences add 8 "-" to the second string and repeats the same process with right string. Finally, the result is determined as:

###

String Number: 1 -> oxnxexzxtwo-------x

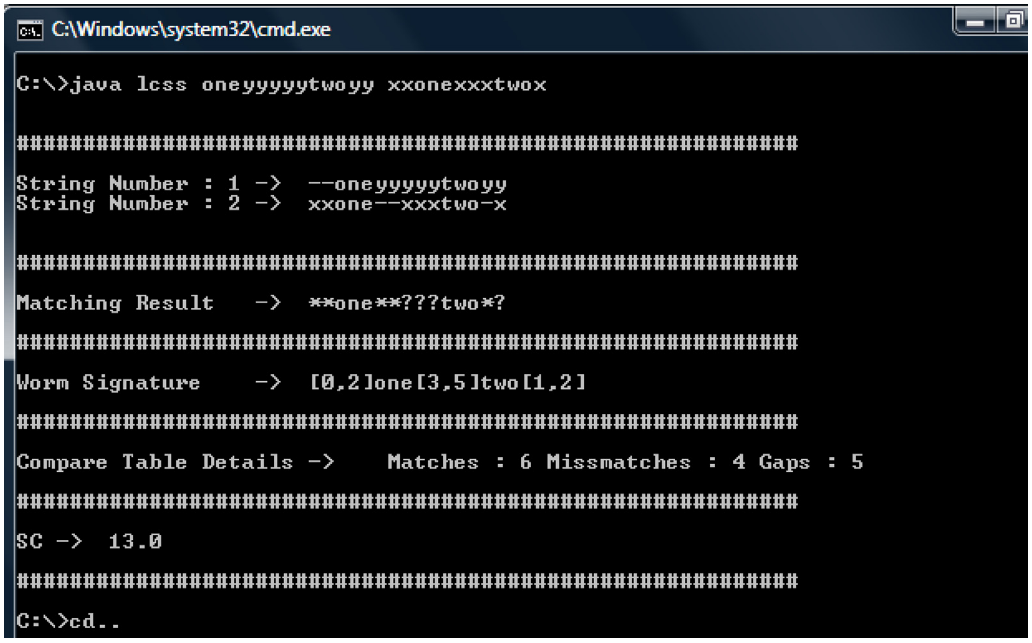
String Number: 2 -> -------ytwoyoynyeyz

###

Note if LCS is not zero in the left or right string this requires more computation of the LCS until it becomes zero. This requires repeating steps 2–4 until these two strings reach a length of LCS zero. Comparing strings character wise produces the following results: In case character at position (n) in string 1 and 2 are same then the same character appears at position (n) in the matching string. This is said to be a "match" i.e. at position 9 in string 1 and 2, we have "*t*" so the matching result will have "*t*" at position 2. If the character at position (n) in string 1 or string 2 is "-" then "*" appears at position (n) in the matching string. This is said to be a gap, e.g. at position 1 in string 1 and 2, they contain "*o*" and "-" so the matching result will have "-" at position 1. In the case where the character at position (n) in string 1 and 2 are not the same and neither is "-" then "?" character appears at position (n) in the matching string, this is said to be a "mismatch". Fig. 10 explains the final signature derived by RCS algorithm for polymorphic worm.

Tang et al. [14] modified the similarity score function of the Needleman–Wunsch algorithm from formula (1) to formula (2):

$$\mathrm{SC}(x,y) = k_m \times W_m + k_d \times W_d + k_g \times \delta + \sum_{\substack{s \text{ is substring} \\ \text{in alignment result}}} Enc(|s|). \tag{2}$$



```
C:\>java lcss oneyyyyytwoyy xxonexxxtwox

##############################################################

String Number : 1 ->  --oneyyyyytwoyy
String Number : 2 ->  xxone--xxxtwo-x

##############################################################

Matching Result   ->  **one**???two*?

##############################################################

Worm Signature    ->  [0,2]one[3,5]two[1,2]

##############################################################

Compare Table Details ->    Matches : 6 Missmatches : 4 Gaps : 5

##############################################################

SC ->  13.0

##############################################################

C:\>cd..
```

Fig. 11. Another signature example of sequence alignment by using CRS algorithm.

Tang et al. [15] recommend providing a score function enc() for rewarding contiguous substrings. In instance for sequences, they have defined as $\text{enc}(x) = 3(x-1)$ and set $W_m = 0.5$, $W_d = 0$, $\delta = -1$, the match score is $-6.5(0.5 \times 3 + 0 \times 2 + -1 \times 14 + 3 \times (3-1))$. Therefore, the CSR algorithm will result in the enhanced alignment for two sequences given with a maximum score -6.5. For the purpose of supporting the wildcard the CSR algorithm allows wildcard '?' and '*' in the sequence input and the sequence result will follow the comparison rules. For preserving distance restrictions this algorithm has a supposition for every character in sequences in a length area [Min, Max] where min is the low-bound and max is the up-bound. According to the alignment result from Fig. 10 the length range of each "*" is [0, 1], the length range of each "?" is [1, 1]. Combine the eight previous wildcards to substring "two" to one reiterating qualifier, and then calculate its low-bound length range to $0\times7+1 = 1$, up-bound to $1\times7+1 = 8$. Therefore, the reiterating qualifier previous to the substring 'two' is '[1, 8]'. Repeat this procedure, then convert the resulted to SRE signature "[1, 8] 'two' [1, 8]", as shown in Fig. 11.

To verify the results extract by the CSR algorithm, it has been used with other samples to generate accurate signatures. Additionally, all steps above have been repeated, showing that this algorithm preserves all the features of polymorphic worms because it grasps the length of the common sequence rather than being concerned with computing the optimal score between sequences. Fig. 11 reveals the signature match for two sequences. This algorithm is successful in deriving contiguous sequences and keeps all wildcards of the polymorphic worm.

5. Discussions

This paper investigated the architecture and evaluation for each method and demonstrated possible ways to overcome the problem raised from using these methods to generate accurate signatures. The signature should have high efficiency and reliability. The accuracy of the polymorphic worm signature is ensured by explaining all the possible features of the worm, as a result of which, the level of false positive results can be reduced. Additionally, the signature should not include any incorrect or useless features of the worm, which can also be helpful in avoiding false negative results. We have focused on the automatic methods used to generate accurate signatures. All signatures were based on filtering contents sequences of polymorphic worms.

Initially, the autograph method was used to infer the worm signature, which is based on matching single substring. The signature resulting from the autograph method was shown to be good for worm with invariant strings; however it was less useful for polymorphic worms since the signature contains a single contiguous string only. Additionally, the quality of the signature generated by the autograph method depended on the size of the content block. Small content sizes produced high levels of false positive results and large sizes increased the sensitivity of the performance of the autograph method. Various authors [5,16,19] argued that the main disadvantage of the autograph method is that it lacks the ability to detect large classes of worm as well as being relatively inefficient in detecting polymorphic worms as they do not contain enough common substrings. The autograph method also failed to preserve the distance restrictions between the invariant parts in the signature results. Furthermore, the single signature was not sufficiently qualified to match all worm patterns with low false negatives and low false positive results. Thus, the signature generated by using autograph is too inefficient, and failed to generate sufficiently accurate signatures for the polymorphic worm.

In contrast to the previous approach, the polygraph approach works effectively with polymorphic worms. The polygraph method is based on multiple invariant string matching and is considered to be more effective as it does not result in large quantities of false positives. However the signature generated by this approach is inaccurate because it does not take into its account all the invariant parts of the worm. This method can extract most invariant parts and distance restrictions, but ignores one byte invariant parts as "=" in the Code RedII worm. Tang et al. [14] explained that the polygraph approach relies on a well-classified worm flow pool and explicit protocol testing. The polygraph mechanism is also known to be complicated and may suffer from some false anomalous flow attacks. Additionally, it is based on extracting multiple invariants from the polymorphic worm payload, which changes appearance with each infection. This system captures the payload packet from a router; therefore in a worst case scenario may discover multiple polymorphic worms, each potentially exploiting different vulnerabilities. This could complicate the process of finding the invariant parts shared among these polymorphic worms. A hacker can send a pattern of the worm to the network and this worm change its payload automatically in each infection to produce other patterns. Security experts using this method therefore need to provide the polymorphic worm a chance to interact with hosts without any recital effect in order to capture all the polymorphic worm instances.

The final method, SRE, preserves the length of substrings with all lengths of distance restrictions. Thus, the SRE system has the ability to capture the polymorphic worm signatures more accurately and efficiently if compared with the two systems explained previously. While SRE extends polygraph's token subsequence signature model with length constraints, it still works well with all the sequences of the polymorphic worm. This system relies on a bioinformatics mechanism to detect the worm signature to extract the alignment sequences from the polymorphic worms by using Needleman–Wunsch algorithm. This algorithm was further improved by using CRS algorithm to capture contiguous invariant fixed parts in the worm patterns. Tang et al. [5,14] claimed that this system does not rely on a well-classified worm flow pool and explicitly protocol analysis. Overall, the SRE generates efficient and accurate signatures to match polymorphic worms.

The results show that the SER approach is successful as it preserved all the invariant parts and emphasised the order of distance restrictions between these invariants. However, these results require more understanding of complex algorithms and using other branches of science and methods in this field.

Furthermore, the results in the previous chapter demonstrated the transformation of a set of polymorphic worm patterns to a set of character sequences. A Needleman–Wunsch matrix was used to find the maximum alignment score that detects the common characters between two sequences. However, the significant limitation in this algorithm is that the data is misrepresented due to its computing misinterpretation of the optimal score. The goal was to match the similarities between the characters. A manual way was used to retrieve the maxim score, which was zero. This result was unsuitable for the polymorphic worm signature because the algorithm maximised the score without taking into account the alignment between the characters. To overcome this problem, a gap was inserted to align the matching character. Perl script was used to produce sequences between the characters revealed in Figs. 9 and 10.

The result proved that this algorithm is unsuitable to produce consecutive matches; in Fig. 9, the maximum similarity score was −6, while it was −11 in Fig. 10. However, the smallest value was observed to contain a contiguous substring 'two' which is semantically significant, and similar to the invariant parts of polymorphic worms. Despite this, it was still unsuitable for polymorphic worm detection because it does not take into account restriction distance. So the Needleman–Wunsch algorithm emphasised the overall maximum number of matches alternatively concerned with consecutive matches. The Needleman–Wunsch algorithm failed to produce contiguous substrings, which are important for worm traffic detection. The result was enhanced by preserving the invariant parts and keeping restriction distance using CRS algorithm converted to java script. The power of CRS comes from its ability to match strings for the longest common substring between two sequences. The CRS algorithm proved that the sequence result has an optimised alignment.

The results of this study verify the results obtained by Tang et al. [14]. Tang et al. [14] evaluated quality of the signature generated by SRE and proved that it is more accurate than the conjunction signature created by the polygraph method. SRE generates signatures and simultaneously keeps the distance restrictions for invariant content by the reiterating qualifiers. They verified that in all their experiments on the approaches of detecting polymorphic worm signatures, compared with present network-based signature generation systems approaches, the SRE mechanism produced more precise and effective signatures to match polymorphic worms. Polymorphic worm detection using SRE was shown to be successful automatic signature generation to find accurate signatures by detecting the longest substring common in a set of suspicious samples.

The previous approaches were shown to be less effective, whether autograph that generates single contiguous byte string signature, or polygraph that generates token-based signature (bytes sequence) occurring in significant numbers of suspicious flows pool. The single string signature technique is not effective at matching polymorphic worms and the signature generation signature depends on the token-based signature which can lose vital information associated with any sequential token such as restriction distances.

This paper demonstrates that the signature generation by SRE is more useful. As shown in Fig. 11, this method is successful in preserving the most important features of the worms, namely invariant parts and restriction distances. These are critical for the purpose of exploiting the vulnerability in victim devices. However, the SRE also has shortcomings, as it only generate signature for single polymorphic worm and may fail when attempting to generate signatures for a mixed number of various worms.

However, the overall limitations of these methods are that it is possible to generate polymorphic shellcode, which does not have invariant bytes. All these approaches may fail to generate signatures for polymorphic worms that do not have invariant bytes [23] and they would then be unable to explain any exploit-based signature because all methods are effective for a considerable portion of polymorphic worm. However, Newsome et al. [17] observed that polymorphic worm should contain invariant bytes, as they are significant to exploit vulnerable server. These approaches may issue limitations of all exploit-based signature generation approaches, sine generated signatures are not able to identify all prospective exploits of a vulnerability.

All approaches mentioned above tested their mechanisms offline on synthetically produced polymorphic worms, as they do not know polymorphic worms on the real Internet, so evaluation is not possible. Most of the techniques mentioned above rely on the classification of the network flow, hence if they are classified between the malicious and normal traffic this may affect the final accuracy of the signature generation because the content filtering could include noise.

The most critical recommendation that can be extracted from this study is that the dealing with the countermeasures against this type of worm by generating accurate signatures could be more complex and may need to work with various intricate algorithms or static algorithm forms in order to extract the accurate and efficient signature to be used with protection applications such as IDSs or firewalls. The accurate signature should describe all features (body) of the worms, including invariant parts and restricted distance.

The implementation accomplishes with several algorithms such as manual Needleman–Wunsch and after numerous experiments using Perl computer language script the result is that this is not efficient enough to represent the polymorphic worm. Furthermore, it was necessarily to develop this algorithm by extending it with the CRS algorithm. In order to preserve time and improve the result, this study converted the CRS algorithm to Java script. That mean deals with this type of mechanism may account complex. Also, as mentioned in Section 2 these types of worm exploit the vulnerability on the victim machine so, with their ability to change appearance in each attack. The patch works against its exploit and that by itself a time consuming process. Each automatic technique used to generate signatures should also be able to work on or off-line.

It should be noted that the developed algorithm could be applied in the Cloud services and systems with the same considerations and limitations. The conceptual similarity between the developed algorithm with the Cloud computing adoption framework (CCAF) security [24,25] are multilayer security approaches. For example, our security approach suitable for some levels such as Datacenter software security level and Hypervisor software security level.

6. Conclusions and future work

This aim was achieved by providing an overview of polymorphic worms and investigating existing automatic methods based on contents detection of polymorphic worm signatures in order to create accurate and efficient signatures. These methods are autograph, polygraph and SRE. Using SRE technique to generate efficiently polymorphic worm signatures carried out the actual development in this study. This exploited various algorithms, such as Needleman–Wunsch and CRS algorithm, and also used Perl and Java script to increase the accuracy of the results. Finally, the evaluation quality of the signatures resulted from automated methods of polymorphic worm signature detection.

Consequently, this study has discussed the seriousness of the threat that polymorphic worms pose to the security of Internet infrastructures. This is particularly problematic as this type of virus does not require human interaction and is able to spread across very large networks quickly, changing its appearance with every attack. In this study the automatic methods have been used to detect accurate signature for polymorphic worm core in NSG were successfully able to identify worm patterns on networks at early stages before they were able to quickly spread to other targets. To overcome the problem of the Needleman Wunsch algorithm generating a large number of useless characters, it proposes a novel pairwise sequence alignment algorithm CSR that extends from Needleman–Wunsch. The results indicated that the SRE approach is effective for automatically generating the signature of polymorphic worms, since it preserved all worm characteristics. Therefore, the proposed algorithm has investigated and the evaluation showed a significant accuracy and effect signatures.

As a part of future work, investigating the extraction of signatures online with rapid detection in the Cloud Environment. Furthermore, some research is being carried out in order to discover new approaches that provide additional powerful methods of accurately investigating the intrinsic similarities of worm patterns.

References

[1] A. Moftah, A. Maatuk, Plasmann, S. Aljawarneh, An Overview about the Polymorphic Worms Signatures, in: ICEMIS '15 September 24–26, 2015, Istanbul, Turkey©2015, ACM, 2015, ISBN 978-1-4503-3418-1/15/09, http://dx.doi.org/10.1145/2832987.2833031.

[2] S. Aljawarneh, A web engineering security methodology for e-learning systems, in: Network Security, Vol. 2011, Elsevier, 2011, pp. 12–15. http://dx.doi.org/10.1016/S1353-4858(11)70026-5.

[3] S. Aljawarneh, Cloud security engineering: avoiding security threats the right way, Int. J. Cloud Appl. Comput. 1 (2) (2011) 64–70. http://dx.doi.org/10.4018/ijcac.2011040105.

[4] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, G. Vigna, Polymorphic worm detection using structural information of executables, 2006 [Online]. Available from: http://www.springerlink.com/content/f1k63052u27p6438/ (accessed 6.07.2010).

[5] Y. Tang, X. Lu, B. Xiao, Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms, 2009 [Online]. Available from: http://www.springerlink.com/content/f1k63052u27p6438/ (accessed 6.07.2010).

[6] D. Brumley, J. Newsome, D. Song, H. Wang, S. Jha, Towards automatic generation of vulnerability-based signatures, in: the 2006 IEEE Symposium on Security and Privacy, 2006, pp. 2–16.

[7] I. Arce, Vulnerability, 2005 [Online]. Available from: http://archives.neohapsis.com/archives/sf/ids/2005-q2/0130.html (accessed 14.07.2010).

[8] M.M. Saudi, E.M. Tamil, S.A.M. Nor, M.Y.I. Idris, K. Seman, EDOWA worm classification, in: Proceedings of the World Congress on Engineering 2008 Vol I.WCE 2008, 2008, ISBN: 978-988-98671-9-5.

[9] J. Wang, I. Hamadeh, G. Kesidis, D. Miller, Polymorphic worm detection and defense: system design, experimental methodology, and data resources, 2006 [Online]. Available from: http://portal.acm.org/citation.cfm?id=1162666.1162676 (accessed 15.08.2010).

[10] M.V. Gundy, H. Chen, Z. Su, G. Vigna, Feature Omission Vulnerabilities: Thwarting Signature Generation for Polymorphic Worms, 2007 [Online]. Available from: http://www.cs.ucdavis.edu/~hchen/paper/acsac07.pdf (accessed 15.08.2010).

[11] H. Noh, J. Kim, C.Y. Yeun, K. Kim, New polymorphic worm detection based on instruction distribution and signature, 2008 [Online]. Available from: www.caislab.kaist.ac.kr/publication/paper_files/2008/SCIS08_Hanyoung.pdf (accessed 18.08.2010).

[12] B. Bayoglu, I. Sogukpinar, Polymorphic worm detection using token–pair signatures, 2008 [Online]. Available from: http://www.pdffact.com/worm-detection-using-local-networks.pdf (accessed 27.08.2010).

[13] B. Xiao, Y. Tang, J. Luo, G. Wei, Concept, characteristics and defending mechanism of worms, IEICE Trans. Inf. Syst. E92-D (5) (2009) 799–809.

[14] Y. Tang, B. Xiao, X. Lu, Generating simplified regular expression signatures for polymorphic worms, in: The 4th International Conference on Autonomic and Trusted Computing, ATC-07, 2007, pp. 478–88.

[15] H.A. Kim, B. Karp, Autograph: Toward automated, distributed worm signature detection, in: USENIX Security Symposium, 2004, pp. 271–286.

[16] H.A. Kim, Privacy-preserving distributed, automated signature-based detection of new internet worms, 2010 [Online]. Available from: http://reports-archive.adm.cs.cmu.edu/anon/2010/CMU-CS-10-122.pdf (accessed 12.10.2010).

[17] J. Newsome, B. Karp, D. Song, Polygraph: Automatically generating signatures for polymorphic worms, in: Proceedings of the 2005 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Washington, 2005, pp. 226–241.

[18] S.B. Needleman, C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, J. Mol. Biol. 48 (1970) 443–453.

[19] S.K. Singh, K.C. Roy, V. pathak, Channels reallocation in cognitive radio networks based on dna sequence alignment, 2010 [Online]. Available from: http://airccse.org/journal/ijngn/papers/0610jngn3.pdf (accessed 14.09.2010).

[20] Z. Du, F. Lin, Improvement of the Needleman–Wunsch algorithm, 2004 [Online]. Available from: http://www.springerlink.com/content/ln7af6d0bpvgqp14/ (accessed 15.09.2010).

[21] A.M. Lesk, Introduction to Bioinformatics, third ed., Oxford University Press, 2008.

[22] W.K. Sung, Algorithms in Bioinformatics: A Practical Introduction, Chapman & Hall/CRC, 2009.

[23] Y. Song, M.E. Locasto, A. Stavrou, A.D. Keromytis, S.J. Stolfo, On the infeasibility of modeling polymorphic shellcode, 2007.

[24] V. Chang, Y. Kuo, M. Ramachandran, Cloud computing adoption framework: A security framework for business clouds, Future Gener. Comput. Syst. 57 (2016) 24–41. http://dx.doi.org/10.1016/j.future.2015.09.031.

[25] V. Chang, M. Ramachandran, Towards achieving data security with the cloud computing adoption framework. services computing, IEEE Trans. Comput. Serv. 99 (2015).

Shadi A. Aljawarneh is an associate professor, Software Engineering, at Jordan University of Science and Technology, Jordan. He holds a BS.c. degree in Computer Science from Jordan Yarmouk University, a M.Sc. degree in Information Technology from Western Sydney University and a Ph.D. in Software Engineering from Northumbria University-England. He is currently an associate professor in faculty of IT in Isra University, Jordan, where he has worked since 2008. His research is centred in web and network security, e-learning, bioinformatics, Cloud Computing and ICT fields. Aljawarneh has presented at and been on the organising committees for a number of international conferences and is a board member of the International Community for ACM, Jordan ACM Chapter, ACS, and others. A number of his papers have been selected as "Best Papers" in conferences and journals.