Generating Malware Signature using Transcoding from Sequential Data to Amino Acid Sequence

Yue Zhao*, Yong Tang†, Yijie Wang‡, Shuhui Chen§
College of Computer
National University of Defense Technology, Changsha, China
*Email: zhaoyue.89@gmail.com
†Email: ytang@nudt.edu.cn
‡Email: wangyijie@nudt.edu.cn
§Email: csh999@263.com

***Abstract*—Signature generation is critical for malware defense. Since the manual operation of signature generation costs too much time and does not guarantee the accuracy, the automatic signature generation has raised great concerns. In this paper, we propose a novel approach for automatic signature generation of malware, which directly leverages bioinformatics algorithms and toolkits based on transcoding. Initially, we convert the malware sequential data, like propagation dataflow, system call sequences, malicious file content, etc. into amino acid sequences by transcoding. Then we leverage multiple sequence alignment software in bioinformatics, such as CLUSTAL, T-COFFEE and MUSCLE to align amino acid sequences. Finally, based on the alignment result of the amino acid sequences, the malware sequential signatures can be obtained through an inverse transcoding procedure. In our experiments, some multiple sequence alignment software based on different algorithms are evaluated and compared for the effect and efficiency of signature generation.**

***Keywords*—signature generation, transcoding, bioinformatics, multiple sequence alignment, malware**

I. INTRODUCTION

Most kinds of current malware defense systems, like IPS, anti-virus system and firewall are still mainly built on signature-based detection. Therefore, generating malware signature, especially for sequential data, like propagation dataflow, system call sequence, malicious file content, etc. is critical for malware defense so far. Due to the limitations in terms of time and quality for manual malware signature generation [17], a number of automatic signature generation approaches have been proposed in recent years.

Accuracy is the most important requirement for network-based signature generation (NSG) systems. Some recently proposed approaches such as SRE [18], [19] and PolyTree [20] based on multiple sequence alignment (MSA) techniques have shown the advantage in terms of accuracy. Unlike the re-implementation of one existing MSA algorithm in these works, in this paper we directly leverage MSA software in bioinformatics to generate signatures of malware sequential data by using the technology of transcoding, because there are some similarities between malware sequential data and gene sequences/amino acid sequences. We firstly convert malware sequential data, such as byte sequences in malware propagation, into amino acid sequences by transcoding. Then we use MSA software in bioinformatics, such as CLUSTALW [21], T-COFFEE [13] and MUSCLE [3] to align amino acid sequences. Finally, based on the alignment result of amino acid sequences, we obtain malware sequential signatures through an inverse transcoding procedure. The advantage of this approach is that a number of algorithms which are still evolving in MSA software can be more efficiently leveraged for malware signature generation. Compared to the re-implementation, the MSA software can align multiple sequential data more fast with just a little inaccuracy. Moreover, our experiments show the different advantages we get from different MSA software.

This paper is organized as follows. In Section II we introduce the related work about sequence alignment in bioinformatics and some methods of automatic signature generation based on sequence alignment in malware defence. In Section III we present the framework about automatic signature generation using MSA software through the transcoding. In Section IV we propose the algorithm of the transcoding which converts the malware sequential data into amino acid sequences. In Section V we conduct experiments that generate signatures of 13 groups of malware sequential data, and analyze the accuracy and time consumption. In Section VI we conclude this paper and prospect our future work.

II. RELATED WORK

A. Sequence Alignment in Bioinformatics

Sequence Alignment is a method that compares two or more character sequences to get the similarities and deviances among them. In 1970, it is first proposed by Needleman and Wunsch for the pairwise global alignment which evaluates amino acids by match/mismatch scores and gap penalties [10]. Then, the pairwise local alignment is proposed by Smith and

The work was partially supported by the National Natural Science Foundation of China under Grant No. 61003303. The work was partially supported by the National Natural Science Foundation of China under Grant No. 2011AA01A103.

978-1-4799-0838-7/13/$31.00 ©2013 IEEE

Waterman in 1981 [15]. Both of them are based on dynamic programming. Nowadays, the best known double sequence alignment software are FASTA [8] and BLAST [2]. FASTA format is accepted by most of sequence analysis software in bioinformatics, while BLAST is the most widely used database research algorithm for protein or nucleic sequences, which employs heuristics.

In recent years, MSA algorithms have attracted more concerns and been developed fast. Most of MSA progressive algorithms [5] analyze multiple sequences in these steps [21]. Initially, align all pairs of sequences to get distances between them. Then, build a guide tree based on the distances calculated. Finally, align all of sequences according to the guide tree. Based on the scoring scheme of the pairwise alignment, MSA algorithms can be distinguished between matrix- and consistency-based [12]. Both CLUSTALW and MUSCLE are popular MSA software with matrix-based algorithms. CLUSTALW[a] has made some improvements on gap penalty and weight matrix selection to increase the sensitivity [21]. MUSCLE[b] gets a comparable accuracy with CLUSTALW with less time consumption, because of some improvement in details [3]. Conversely, T-COFFEE and MAFFT are widely used MSA software with consistency-based algorithms. T-COFFEE[c] uses local and global pairwise algorithms to build a library and takes an advantage of accuracy than other commonly used MSA software. However, the time consumption of T-COFFEE is much more than other MSA software [13]. MAFFT[d] provides a considerable scalability of progressive alignment by a new tree-building algorithm, PartTree, and has an improved accuracy for noncoding RNAs [6]. Recently, some people have guided MSA algorithms into other fields, such as network traffic classification [4], [23] and malware signature generation which is introduced in Section II-B in detail.

B. Automatic Signature Generation

NSG system has been evolving quickly in recent years. This system can generate signatures automatically through the analysis of the suspicious network traffic. Initially, most of NSG systems employ their algorithms based on longest common substring (LCS) [7], [14], [22]. However, these methods can only extract the single longest signature fraction, which is not enough to indicate malware signatures in detail. Then, Autograph [1] and Earlybird [14] appear as the methods that partition the suspicious network traffic in fixed length. TANG Y, et al. [16] define the fixed length partition with several signature fractions as the significant region, but it is hard to select the appropriate length for significant region. In 2005, Newsome [11] first proposes the method called Token in the research of Polygraph. Token is the string that appears frequently in the suspicious network traffic, with the length longer than 1. Even though Token is an effective and efficient method, it can not distinguish the signature fractions with the length of 1 and indicate the relationship of the positions among the signatures.

[a]http://www.clustal.org/
[b]http://www.drive5.com/muscle/
[c]http://www.tcoffee.org/
[d]http://align.genome.jp/mafft/

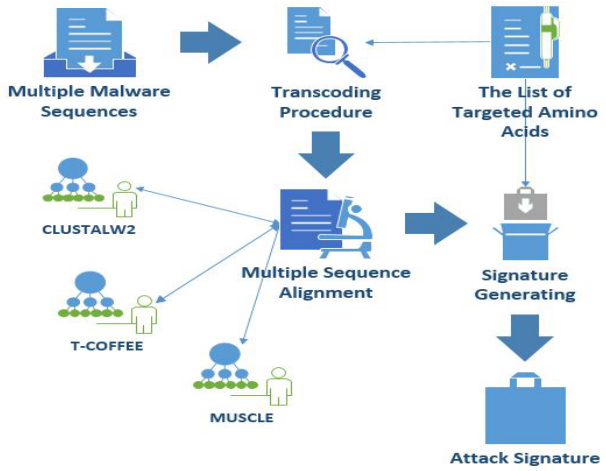

Figure 1. The Automatic Signature Generation Workflow

Considering the problems above, some methods appear to adapt the algorithms of MSA software in bioinformatics for malware sequential data [9], [18]–[20]. Especially, Tang and Xiao propose a new network-based signature generation (NSA) system – PolyTree [20] in 2011. PolyTree is the imrovement of the approach proposed in [19]. It re-implements the algorithms of CSR-Needleman-Wunsch for pairwise sequence alignment and T-COFFEE for multiple sequence alignment in its signature tree generator, and its signature selector generates signatures from the beginning traffic pool based on the current signature tree. Therefore, PolyTree has some advantages such as generating more accurate signatures compared with previous approaches and selecting signature set from multiple polymorphic worms with noise.

These approaches for network security above have made tremendous efforts to adapt MSA algorithms for malware sequential data. Unfortunately, re-implementation of these algorithms demands much time consumption. However, our proposal is to leverage MSA software directly, and our experiments show that this approach has a considerable advantage of time consumption. Moreover, our method of transcoding can also avoid getting meaningless fractions, which is indicated in Section IV-A.

III. FRAMEWORK

We propose the framework of automatic signature generation leveraging MSA software based on transcoding, and the workflow is shown in Fig. 1. This framework is composed of four parts, and the process of automatic signature generation using this framework can be also divided into four steps correspondingly.

Step 1 **Build a list of targeted amino acids which are selected to form the acid sequences which malware sequential data are converted into.** These targeted amino acids in the list do not repeat. Additionally, there should be at least 2 kinds of amino acids in the list so that each byte of malware sequential data can be indicated with corresponding amino acids. For example, if we select sixteen kinds of targeted amino acids, each byte of malware sequential data can be indicated by only 2 amino acids. Furthermore, it can be ensured that the amino acid sequences we get will not be too long to be analyzed by MSA software in bioinformatics.

Step 2 **Transcode malware sequential data into amino acid sequences which can be accepted directly by MSA software in bioinformatics.** In this step, we need to employ or build right mapping functions to indicate each byte of malware sequential data with target amino acids, and produce the result with right output format commonly accepted by most of MSA software in bioinformatics. For example, if we choose the way of using two amino acids to indicate one byte as mentioned in **Step 1**, a mapping relationship between the content of one byte in malware sequential data and the two targeted amino acids should be found or built by a mapping function $f : X \rightarrow Y^2$. Moreover, the mapping function should be a bijection so that inverse transcoding procedure can obtain the correct content of each byte indicated by targeted amino acids. This transcoding will not loss any information contained in malware sequential data, because of the bijection. However, what we concern about is that the transcoding approach may produce redundant information because we select more than one amino acid to indicate one byte, which will be discussed in the example in section IV-A. Additionally, we select FASTA as the input format since it is widely accepted by most of MSA software.

Step 3 **Leverage MSA software in bioinformatics to analyze amino acid sequences produced in Step 2, and generate a result in which the common sequences of amino acid sequences are marked observably.** In order to compare the accuracy of signature generation and prove the universality of the framework we propose, amino acid sequences are put into different kinds of MSA software in bioinformatics. Moreover, the common sequences should be marked observably and tidily in the output format. For instance, CLUSTAL, a kind of widely used output format, has an advantage of marking the common sequences, which is shown in Fig. 2. As we can see in Fig. 2, the left part is the names of selected sequences, while the right part is the contents of sequences corresponding to the names. And the contents of these selected sequences are filled with ”– ” to make the common parts of sequences in the same positions, which makes the result get the highest score from the MSA software. In addition, the last row marks the positions of common sequences with $*$.

Step 4 **Extract the common sequences out of the results generated by MSA software, and generate signatures of malware sequential data through the inverse transcoding procedure.** Initially, the common sequences should be extracted accurately from the result of **Step 3**. Then we should find the correct transcoding starting point and build the corresponding inverse mapping function $g : Y^2 \rightarrow X$ which converts two amino acids into one byte of malware sequential data. Finally, we can generate signatures by transcoding the common sequences.

Compared with existing MSA-based signature generation approaches, there are some advantages of our approach as follows:

(a) We can directly use existing MSA tools and algorithms without re-implementing them for malware sequential data particularly.
(b) So far the research on MSA algorithms is still very active in bioinformatics, so we can keep the advance of MSA algorithm research and easily use the state-of-the-art MSA algorithms in this framework.
(c) Based on a rich number of MSA software, we can choose suitable software considering the balance of accuracy and performance, which is introduced in detail in Section V.
(d) The amino acid sequences generated in **Step 2** are accepted not only by MSA software but also by other kinds of analysis software in bioinformatics. For example, BLAST can be used to build a database of amino acid sequences. Moreover, it is easy to get the degree of the similarity between any amino acid sequence and others in this database. Furthermore, it is possible to build a phylogenetic tree according to the degree of the similarity.

IV. ALGORITHM IMPLEMENTATION

In this section, we implement the algorithms of the framework according to our assumptions in Section III. These assumptions are summarized as follows:

(a) We build our list of targeted amino acids with 16 kinds of amino acids.
(b) The content of one byte in malware sequential data is represented by two targeted amino acids in the list.
(c) We select FASTA as the input format, and CLUSTAL as the output format, of MSA software.

A. Transcoding Algorithm

In Algorithm 1, our purpose is to convert all of malware sequential data into amino acid sequences with FASTA format. In line 3, it is the FASTA format that requires the ID (name and

```
MUSCLE (3.8) multiple sequence alignment


d:\data\codered.A.6      MSKPIMALYKP-----SRVCINPPRHKPTWIPSILCHYI----------LNGLTWINGHV
d:\data\codered.A.8      MSKPIMALYKH-----ILNHIKLCKGIPTWIPSILCHYI---------RVNPKNIMHLMP
d:\data\codered.A.12     MSKPIMALYKN--PILLNYCNTYCTW---WIPSILCHYI-------------WAWGGKHM
d:\data\codered.A.1      MSKPIMALYKG----------IPVGL---WIPSILCHYI-----INPTMNMPILYHINLP
d:\data\codered.A.13     MSKPIMALYKS----------VRSIL---WIPSILCHYI--------------------
d:\data\codered.A.18     MSKPIMALYKN------TYCKMKMSY---WIPSILCHYI--------------------
d:\data\codered.A.5      MSKPIMALYKG-----IHIRSKPPTKPINWIPSILCHYI-------------AHPVKNP
d:\data\codered.A.16     MSKPIMALYKK--------MRSVYLR---WIPSILCHYIYHNSLMPTLRKNSYTAGLILG
d:\data\codered.A.0      MSKPIMALYKG--------HHILMHI---WIPSILCHYI--------CKMRHKVYHLLNP
d:\data\codered.A.19     MSKPIMALYKG----------KKPKL---WIPSILCHYI--------------------
d:\data\codered.A.4      MSKPIMALYKN-----------------RWIPSILCHYI--------------------
d:\data\codered.A.7      MSKPIMALYKR---WKLNRILKNMNGILPWIPSILCHYI--------------NPNTLP
d:\data\codered.A.17     MSKPIMALYKH----------LKLKM---WIPSILCHYI----------ILKMVYKLMRP
d:\data\codered.A.10     MSKPIMALYKS-------YKLNPCKHMKPWIPSILCHYI--------------------
d:\data\codered.A.11     MSKPIMALYKPSGKINNSKLAHACCH---WIPSILCHYI--------PRKNHLTWHMKLS
d:\data\codered.A.3      MSKPIMALYKS------------YNS---WIPSILCHYI-------PRSVGILNWGMPKP
d:\data\codered.A.2      MSKPIMALYKT--------ANPPSHM---WIPSILCHYI-----MRCKHMINRSYHLMLM
d:\data\codered.A.15     MSKPIMALYKP----------SRSHM---WIPSILCHYI-------CKRVLRVYWAILNP
d:\data\codered.A.9      MSKPIMALYKI------NWARVACRV---WIPSILCHYI--------------------
d:\data\codered.A.14     MSKPIMALYKL--------MKLTAGH---WIPSILCHYIYCHLHINTKLYCPVILGLKLK
                         **********                  **********
```

Figure 2. An Example of CLUSTAL Format Output from MUSCLE

Algorithm 1 Transcoding Procedure

Require:

Malware sequential data $S_0, S_1, \ldots, S_i, \ldots, S_{(N-1)}$.

The list of targeted amino acids, $List$.

Ensure:

Amino acid sequences $A_0, A_1, \ldots, A_i, \ldots, A_{(N-1)}$ in FASTA format input of MSA software, $FASTA$.

1: Initialize an empty current amino acid sequence A_i for every current malware sequence S_i.
2: **for** each malware sequence S_i **do**
3: Set the ID of amino sequence A_i and write it into $FASTA$.
4: **while** current byte j in S_i is not current sequence terminator **do**
5: Transform j into an integer $x \in X$.
6: Use the mapping function $f : X \rightarrow Y^2$ to get corresponding integers $y_1, y_2 \in Y$.
7: Select amino acid characters $List[y_1]$ and $List[y_2]$, and write them in A_i in $FASTA$.
8: **end while**
9: **end for**
10: **return** $FASTA$

other information) of every amino acid sequence which should be unique, so that MSA software can correctly distinguish each of them. In line 5 we use integer x, which value range is from 0 to 255, to indicate the content of one byte. And in line 6, the value range of y_1, y_2 is 0 to 15. Moreover, the most important problem in Algorithm 1 is that the selection of the mapping function $f : X \rightarrow Y^2$. Bijection is the sufficient and necessary condition, and the functions we select are listed as follows.

$$y_1 = x\%16 \tag{1}$$

$$y_2 = (16 - x \div 16 + x\%16)\%16 \tag{2}$$

Proof. A bijection is a function which is both a surjection and an injection. First we will prove surjection. If we select x from 0 to 15, we can get the value ranges of y1 and y2 are both 0 to 15. Therefore, this mapping function is a surjection. For injection, we select a proof by contradiction. We assume that $f(x_1) = y_1y_2$, $f(x_2) = y_1y_2$ and $x_1 \neq x_2$. Because $y_1 = x_1\%16$ and $y_1 = x_2\%16$, we can get $y_2 = (16 - x_1 \div 16 + y_1)\%16$ and $y_2 = (16 - x_2 \div 16 + y_1)\%16$. Additionally, we can also get $x_1 \div 16 \neq x_2 \div 16$. Moreover, we get $0 \leq (16 - x_1 \div 16 + y_1) \leq 31$ and $0 \leq (16 - x_2 \div 16 + y_1) \leq 31$, because the value range of x is from 0 to 255. If we assume $x_1 > x_2$, we get $(16 - x_1 \div 16 + y_1) + 16 = (16 - x_2 \div 16 + y_1)$, which means $x_1 \div 16 = x_2 \div 16 + 16$. Therefore, we can get a contradiction between $x_1 \div 16 > 16$ and $0 \leq x_1 \leq 255$

The advantage of these equations we select is that y_1 and y_2 will change obviously and independently if x changes in a small range. In this way, MSA software can evaluate amino acid sequences accurately. For example, we assume that the matched character in two sequences will get 2 scores, and the mismatched character will get -1 score. The purpose is to get the longest common sequence of two sequence 12212112 and 12211221. If two amino acids AA indicates 1 and AB indicates 2, these two number sequences will be transcoded to $AAABABAAABAAAAAB$ and $AAABABAAAAABABAA$. Then the longest common sequence we get must be $AAABABAAA - A - A$ which gets 21 scores. However, if AB indicates 1 and CD indicates 2, the longest common sequence we get from the two sequences above is $ABCDCDAB$, which gets 16 scores. Actually, it is obvious that the common sequence of these two sequences is 1221, which is indicted by $AAABABAA$ or $ABCDCDAB$.

B. Signature Generation Algorithm

In Algorithm 2, we plan to generate the signature from the output of MSA software. The generation of signature can be

Algorithm 2 Signature Generation Procedure

Require:
CLUSTAL format output of MSA software, $CLUSTAL$.
The list of targeted amino acids, $List$.

Ensure:
Attack signature of malware sequential data $S_0, S_1, \ldots, S_i, \ldots, S_{(N-1)}$.

1: Initialize an empty intermediate sequence M, an empty signature sequence SGN.
2: Search the sequences with character $*$ as mark sequences in $CLUSTAL$.
3: Confirm the amino acid characters C which are marked by $*$ in $CLUSTAL$.
4: Write C into M with interval character δ_1 representing every amino acid character which is not marked by $*$.
5: **while** M is not empty and M has not been read out **do**
6: Read 2 characters in M.
7: **if** there is at least one δ_1 in these 2 characters **then**
8: Write an special interval character δ_2 into SGN.
9: **else**
10: Search these two amino acid characters in $List$ and record the array indexes $y_1, y_2 \in Y$.
11: Use the inverse mapping function $g : Y^2 \rightarrow X$ of the function $f : X \rightarrow Y^2$ to get the corresponding integer $x \in X$.
12: Transform x into one byte i.
13: Write i into SGN.
14: **end if**
15: **end while**
16: Format SGN to malware signatures.

divided into two steps. The first step is to extract the common sequences of amino acid sequences in $CLUSTAL$. We should extract the common sequence characters marked by $*$ and the interval size among the common sequence characters, which is depicted from line 2 to line 4. Then, from line 5 to line 15, we employ the inverse transcoding function to convert the amino acids of M into each byte of the signatures of malware sequential data.

There are two important problems to be solved. **How to determine the correct transcoding starting point?** An error starting point incurs a result which is transcoded incorrectly. Our solution is to choose one of the amino acid sequences as a reference sequence, and count from the beginning of the reference sequence until the first marked character is reached. If the counted result is an even number, it is the transcoding starting point. Conversely, if the counted result is an odd number, the transcoding starting point is just the next character no matter it is marked or not. **What is the specific inverse mapping function $g : Y^2 \rightarrow X$ of the function $f : X \rightarrow Y^2$?** It is an algebra problem. Because of bijection, we build the function $g : Y^2 \rightarrow X$ as follows:

$$x = ((y_1 - y_2)\%16) \times 16 + y_1 \quad (3)$$

TABLE II
THE LENGTH OF SIGNATURES GENERATED IN OUR EXPERIMENTS

	The Length of Signatures Generated (Byte)			
Data Set	CLU2	MUS	T-CO	PolyTree
apache_host	36	40	40	40
athttps	16	16	16	18
athttps.A	32	31	34	34
athttps.B	29	29	31	31
athttps.C	32	32	34	34
bindTSIG	0	0	3	6
codered.A	34	34	35	36
codered.B	34	34	36	36
codered.C	34	34	36	36
codered.D	33	34	36	36
codered.E	33	34	36	36
codered.F	34	34	36	36
IISprinter	25	27	29	29

V. EXPERIMENTS AND ANALYSIS

The platform of our framework is a 2-CPU x86 machine with 4GB memory using a Windows 7 operating system. We select CLUSTALW2, T-COFFEE and MUSCLE as MSA software in our framework and PolyTree as the software of reference. The list of targeted amino acids consists of alanine (A), aspartate or asparagines (B), cystine (C), glycine (G), histidine (H), isoleucine (I), lysine (K), leucine (L), methionine (M), asparagines (N), proline (P), glutamine (Q), arginine (R), serine (S), threonine (T), valine (V). Additionally, the reason why we select these amino acids is that each of them can get a relatively high score in the scoring matrices if it matches with itself. We run our experiments on 13 malware sequential data sets, and get the features and execution times of these data sets.

The time complexity of the algorithm in CLUSTALW2, T-COFFEE, and MUSCLE is $O(N^2L^2)$ [3], $O(N^2L^2) + O(N^3L) + O(N^3) + O(NL^2)$ [13], and $O(NL^2 + N^3L)$ [3] respectively. In addition, the time complexity of the algorithm of PolyTree is $O(N^2L^2)+O(N^3L)+O(N^3)+O(NL^2)$ [20]. It is obviously shown in Table I that the time consumptions of CLUSTALW2, T-COFFEE and PolyTree have close relationships with the average length and the number of sequences in data sets. Even though MUSCLE does not present a result of time consumption which conforms to the time complexity, the advantage it presents is that the time consumption for most of data sets is relatively low.

In Table II, it is shown that the results of CLUSTALW2 and MUSCLE have a small bit of signatures lost in accuracy, but the framework which applies CLUSTALW2 or MUSCLE can obviously take the advantage of time consumption. Moreover, the results of T-COFFEE can nearly reach the accuracy of PolyTree with much lower time consumption, especially when the average length of data sets becomes longer.

We analyze the signatures we have got in detail, and find that the lost parts of signatures are very short sequences, and these lost sequences have long distances with other sequences

TABLE I
THE FEATURE AND EXECUTION TIME IN OUR EXPERIMENTS

Data Set	Number	Avg Length (Byte)	Execution Time (Sec)			
			CLUSTALW2	MUSCLE	T-COFFEE	PolyTree
apache_host	40	892	206	110	2527	4037
athttps	20	830	52	29	398	629
athttps.A	20	830	51	42	389	678
athttps.B	20	830	51	59	397	653
athttps.C	20	830	51	38	386	658
bindTSIG	40	561	79	127	1232	1436
codered.A	20	303	8	7	68	73
codered.B	20	303	8	12	67	70
codered.C	20	303	8	11	67	72
codered.D	20	305	10	11	65	76
codered.E	20	304	8	6	64	74
codered.F	20	305	8	6	62	70
IISprinter	40	466	58	38	1003	985

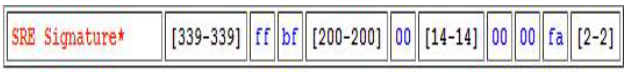

Figure 3. The Signature of bindTSIG

in signatures. The main reason is that the short sequences matched cannot get scores high enough to set them apart from sequences which are not matched in MSA software. Especially, the results of the data set $bindTSIG$ are obviously caused by short sequences in signature, and the detailed signature is depicted in Fig. 3. The signature of $bindTSIG$ consists of three short sequences, which have long distances from each other. It results in some of short sequences lost in signature generated by MSA software. Therefore, making changes of parameters selected in MSA software to reduce the loss of signatures will be the next step of our work in the future.

Summarize: the time consumption of MUSCLE is relatively low, but the execution time is hard to predict. Because the relevance of the time consumption to the number and the average length of sequences is weaker than T-COFFEE, MUSCLE and PolyTree. T-COFFEE is more accurate because it could get some short sequences in signatures generated, while CLUSTALW2 and MUSCLE will lose some short sequences which have one or two characters. Therefore, the users are suggested to apply different MSA software for different demands, such as MUSCLE for reducing time consumption and T-COFFEE for improving accuracy.

VI. CONCLUSIONS

Through the analogy from the signature generation of malware sequential data, to the extraction of the common sequences in multiple amino acid sequences, we propose an approach of automatic signature generation for malware sequential data, using MSA software in bioinformatics based on transcoding. The framework of this approach is updatable, economical and extendable. Moreover, the transcoding algorithm is implemented preliminarily. In the experiments, we select three kinds of MSA software in bioinformatics such as CLUSTALW2, T-COFFEE and MUSCLE to generate signatures. Signatures generated by PolyTree were used as reference data. We compare time consumption and accuracy among them, and make suggestions on how to choose them for different demands.

The further work will include but not limited to: the improvement of details such as the selection of targeted amino acids, the determination of the correct transcode starting point, the structure of the mapping function and the variety of accepted formats, the optimization of the selection on parameters in bioinformatics MSA software and the other kinds of analysis software of amino acid sequences generated by transcoding malware sequential data.

REFERENCES

[1] H. ah Kim and B. Karp. Autograph: Toward automated, distributed worm signature detection. In *USENIX Security Symposium*, pages 271–286, 2004.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, Oct 1990.

[3] R. C. Edgar. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5:113, Aug 2004.

[4] K. Fabjański and T. Kruk. Network traffic classification by common subsequence finding. *Computational Science–ICCS 2008*, pages 499–508, 2008.

[5] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: an integrated method. *J Mol Evol*, 20(2):175–186, 1984.

[6] K. Katoh, K.-i. Kuma, H. Toh, and T. Miyata. Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Res*, 33(2):511–518, 2005.

[7] C. Kreibich and J. Crowcroft. Honeycomb - creating intrusion detection signatures using honeypots. In *USENIX Technical Conference*, 1990.

[8] D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, Mar 1985.

[9] L. Nan, X. Chunhe, Y. Yi, and W. Haiquan. An algorithm for generation of attack signatures based on sequences alignment. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 3, pages 964–969. IEEE, 2008.

[10] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol*, 48(3):443–453, Mar 1970.

[11] J. Newsome, B. Karp, and D. Song. Polygraph: Automatically generating signatures for polymorphic worms. In *Security and Privacy, 2005 IEEE Symposium on*, pages 226–241. IEEE, 2005.

[12] C. Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput Biol*, 3(8):e123, Aug 2007.

[13] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–217, Sep 2000.

[14] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In *Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, volume 99, 2004.

[15] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J Mol Biol*, 147(1):195–197, Mar 1981.

[16] Y. Tang and S. Chen. Defending against internet worms: a signature-based approach. In *IEEE INFOCOM*, volume 2, pages 1384–1394, 2005.

[17] Y. Tang, X. Lu, and Y. Wang. Survey of automatic attack signature generation. *Journal on Communications*, 30 (2):96–105, 2 2009.

[18] Y. Tang, X. Lu, and B. Xiao. Generating simplified regular expression signatures for polymorphic worms. *Autonomic and Trusted Computing*, pages 478–488, 2007.

[19] Y. Tang, B. Xiao, and X. Lu. Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms. *Computers & Security*, 28:827–842, 2009.

[20] Y. Tang, B. Xiao, and X. Lu. Signature tree generation for polymorphic worms. *IEEE Transactions on Computers*, 60:565–579, 2011.

[21] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res*, 22(22):4673–4680, Nov 1994.

[22] K. Wang, G. F. Cretu, and S. J. Stolfo. Anomalous payload-based worm detection and signature generation. In *Recent Advances in Intrusion Detection*, pages 227–246, 2005.

[23] X. Wang and D. Xu. An efficient sequence alignment algorithm of network traffic. In *International Conference on Networking, Sensing and Control*, pages 1743–1746, 2008.