Scientific Research Publishing

Journal of Information Security, 2017, 8, 296-327
http://www.scirp.org/journal/jis
ISSN Online: 2153-1242
ISSN Print: 2153-1234

Exploring the Effects of Gap-Penalties in Sequence-Alignment Approach to Polymorphic Virus Detection

Vijay Naidu*, Jacqueline Whalley, Ajit Narayanan

School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand
Email: *vijay.naidu@aut.ac.nz, jacqueline.whalley@aut.ac.nz, ajit.narayanan@aut.ac.nz

How to cite this paper: Naidu, V., Whalley, J. and Narayanan, A. (2017) Exploring the Effects of Gap-Penalties in Sequence-Alignment Approach to Polymorphic Virus Detection. *Journal of Information Security*, **8**, 296-327.
https://doi.org/10.4236/jis.2017.84020

Received: August 1, 2017
Accepted: October 16, 2017
Published: October 19, 2017

Abstract

Antiviral software systems (AVSs) have problems in identifying polymorphic variants of viruses without explicit signatures for such variants. Alignment-based techniques from bioinformatics may provide a novel way to generate signatures from consensuses found in polymorphic variant code. We demonstrate how multiple sequence alignment supplemented with gap penalties leads to viral code signatures that generalize successfully to previously known polymorphic variants of JS. Cassandra virus and previously unknown polymorphic variants of W32.CTX/W32.Cholera and W32.Kitti viruses. The implications are that future smart AVSs may be able to generate effective signatures automatically from actual viral code by varying gap penalties to cover for both known and unknown polymorphic variants.

Keywords

Polymorphic Malware Variants, Gap Penalties, Syntactic Approach, Pairwise Sequence Alignment, Multiple Sequence Alignment, Automatic Signature Generation, Smith-Waterman Algorithm, JS. Cassandra Virus, W32.CTX/W32.Cholera Virus, W32.Kitti Virus

1. Introduction

The automatic extraction of virus and other malware signatures for use in antiviral software systems (AVSs) is of paramount importance due to the need to find effective solutions to defend systems against the increasing number and severity of attacks [1]. It is generally accepted that these attacks now pose a global risk [2]. Early work on automatic signature extraction focused on simulating the way that human experts analyzed viruses and generated signatures for use in

AVSs [3].

Typically, suspicious code is identified due to anomalous behavior of a computer system. Human experts then manually analyze the suspicious code to identify invariant code portions (syntactic analysis) or code portions that are regularly executed (semantic analysis). Such analysis leads to the generation of unique signatures for use by AVSs when scanning network packets, user files or memory. Before such signatures can be released, they must be checked against non-malware to ensure that the number of false positives is kept acceptably low. For instance, signatures based only on malware encryption/decryption information are likely to lead to unacceptably high false positives due to the large proportion of normal Internet traffic that also carries encryption/decryption information for integrity (e.g. hash algorithms) and authentication (e.g. certified public keys). But relying on human expertise alone to provide manually extracted signatures is becoming increasingly difficult with the growing volume of malware. As a result, interest continues to grow in methods to improve automatic signature extraction. Semantic approaches [4] [5], in addition to standard dynamic and execution behavior analysis [6] [7], now include methods such as control flow analysis [8] [9], behavior model checking [10] [11], executable graph mining [12] and formal semantic models of analysis [13]. The main problem with a semantic approach is that an infection must occur to produce anomalous behavior. Several execution traces may be required before signatures can be extracted manually, and there is always the risk that such signatures may not be effective for different execution paths of the same viral code. Syntactic or static approaches [14] [15] [16] on the other hand, while possibly preferable because of their ability to extract signatures that may apply to different variants of the same malware family and to generate signatures irrespective of differences in execution paths, have not managed to keep pace with the latest polymorphic and metamorphic techniques used by virus writers to obfuscate their malware [17] [18]. Static signature extraction methods must also disassemble or reverse engineer executable code so that structural analysis of the source code is possible. Such analysis includes: statistical analysis of parameter values and searching for repeating strings [19] [20]; code feature selection [21]; feature extraction [22]; and n-grams analysis [23] [24] [25]. The mapping of executable code to a suitable level of program representation that allows such structural analysis is problematic, however, due to such code being deliberately constructed to hide its functionality, such as through the use of redundant control instructions and variable assignments.

Predicting future metamorphic and polymorphic viral forms to prepare AVSs for as yet unknown variants has remained a distant research goal for both semantic and syntactic techniques. The key to a successful syntactic approach would appear to lie in analyzing malware code directly and without execution, and so removing the need for reverse engineering. By comparing different structural variants of the same virus, a successful structural/static approach may

be able to identify common code patterns despite attempts to obfuscate through polymorphism because, if the virus is to perform its designated payload or function and remain a variant of a virus family, a common code must be present even if it is deliberately obscured. A purely syntactic approach, such as the one proposed in this paper, should detect new polymorphic viral variants independently of semantic knowledge based on execution traces, command and control channels, deduplication and propagation vectors. That is, a purely syntactic approach to new variants should not require prior infection by those variants.

In this study, we focus on a sequence-based automatic signature extraction method for identifying polymorphic malware using syntactic analysis of hex code. Theoretically, malware with polymorphism changes its code and keeps the functions intact, whereas malware with metamorphism changes sub-functionality and code while preserving overall functionality [26]. The implications of this theoretical division are unknown for automatic signature extraction. It is not even known if any metamorphic malware actually exists [27]. For that reason, we confine our approach to polymorphic malware capable of mutating into a potentially infinite number of functionally equivalent but structurally different variants (details below).

Previous work in syntactic signature extraction [28] introduced the idea of using basic pairwise sequence alignment techniques from bioinformatics to identify "consensuses" (common occurrences of hex code) in pairs of variants, which was a signature for that pair. These consensuses were in turn multiply aligned with each other to generate a common consensus (*i.e.* a meta-signature) for all variants [29] [30]. A by-product of alignment is that variable-length viral sequences become of fixed length and longer through the introduction of gaps. Gaps are the segments that are generated when aligning amino acid or nucleotide sequences so that similar and analogous residues in two or more sequences are paired with each other in the same column. These could also get deposited at areas where one or more sequences have some additional residues (produced by an insertion) or have missed some residues (produced by a deletion). Gaps are generally substituted with gap symbols such as blanks, asterisks or hyphens to make it pair up with sequences that have no gaps. If insertions and deletions never occurred, then sequences could simply be paired by shifting them along each other and only considering the alignment that best paired the existing residues. In previous work, the evaluation of these consensus-based techniques was restricted to all known, already identified, polymorphic variants. The signatures extracted were therefore "variant-fit" rather than "variant-predictive". The aim of this paper is to examine whether string searching algorithms of greater sophistication than those investigated previously by Naidu and Narayanan [29], such as the Smith-Waterman algorithm which unlike previous work [29] includes different combinations of gap open and gap extend penalties, can lead to the automatic generation of signatures not just for known variants but also for unknown (future), or newly generated, variants. In order to

train and test these novel approaches to automatic signature detection we use three well-known viruses with their known (for JS. Cassandra virus) and unknown (for W32.CTX/W32.Cholera and W32.Kitti viruses) polymorphic variants (more details in Subsection 5.2).

A significant number of known variants exist for JS. Cassandra; thus, this virus is considered useful for testing the hypothesis that relatively sophisticated gap open and extend facilities do indeed capture known variants that have already been shown to be captured using consensus identification without gap penalties [29]. W32.CTX/W32.Cholera virus and W32.Kitti virus, on the other hand, are used to generate new or unknown variants for testing the effects of the more sophisticated gap open and extend facilities on their newly corresponding viral syntactic signatures that are generated in this research. Well-established viruses are chosen because their structure and behavior are well understood. Virus generation, even for experimental purposes in academic computer laboratories, is illegal in many countries. We state explicitly that the intention of our research is to aid the global fight against cybercrime through understanding the mechanisms leading to new polymorphic variants so that appropriate automatic signature extraction techniques can be developed to help reduce their impact in future, smarter AVS technologies.

In Section 2 and Section 3, we discuss the background of syntactic techniques and previous related work. In Section 4, we describe the problem statement. We then demonstrate our systems and methods in Section 5. Section 6 compares the results against state-of-the-art AVS products. Section 7 contains the conclusion.

2. Background

Because the same viral function can appear in many different physical code forms it has been posited that only semantic analysis will reveal commonalities among variants of the same virus for effective signature generation. As a result, syntactic techniques for signature extraction based on structural detection of malware are relatively unexplored in comparison to semantic techniques, and so there is very little in the way of related literature. What literature there is discussed in Section 3. In order to understand syntactic-based polymorphism detection techniques it is useful to consider a simple example of linguistic signature extraction. Consider the following structurally-related sentences, where the first sentence is the original sentence, and the other three are polymorphic versions of it:

The cat saw the mouse
The mouse was seen by the cat
We see that the cat saw the mouse
We see that the mouse was seen by the cat

Signature extraction is similar to finding the two patterns “cat saw mouse” and “mouse seen cat” that will help to detect all four sentences as variants despite the variable length of the sentences, the movement of tokens within the

sentences and introduction of extra material. If options and alternatives are allowed, "{we see} [cat|mouse] [saw|seen] [cat|mouse]" is an approximate regular expression (rule-based signature) for all four sentences that also allows for derivations of new structural variants not so far encountered (e.g. "that the cat was seen by the mouse was seen by us"). These signature examples are of course simplistic when compared to the real task of automatic signature extraction. Viral signatures must also take into account dependencies between non-adjacent code in order to deal with specific polymorphic features as well as possible rearrangements of code that alter the left-to-right order of signatures. In reality, the first four sentences above would be in hex (machine code) format and require accurate disassembly to a language amenable to structural analysis, and the signature then converted back to hex for real-time scanning of network packets and cached files. Signatures must also be checked for their uniqueness. That is, before the generated signature can be released it must be able to distinguish its source malware from all other malware as well as be consistent with as many variants of that malware as possible. It is generally believed that in 2017 a contemporary AVS may contain between a quarter of a million to half a million signatures due to the increasing rate of release of new malware. Updates to AVSs may require removing old and no longer effective signatures as well as adding new signatures, and this can be expected to become more time-consuming with the growth in occurrences of new malware.

A sequence-based approach to signature extraction was previously proposed and demonstrated using the Smith-Waterman algorithm (SWA) without gap penalties [29]. SWA is used extensively in bioinformatics for sequence alignment (finding common subsequences or consensuses among a set of variable length sequences), and previous work demonstrated the feasibility of using such consensuses in viral hex code as signatures. The approach was further refined [30] by adopting SWA with six different substitution matrices. Results showed that it was possible to extract signatures/meta-signatures after applying data mining rule-extraction techniques to the extracted signatures. Such signatures/meta-signatures can, in turn, be employed as rule-based string templates for creating more specific, variant-oriented polymorphic malware signatures for detecting known variants belonging to the same virus family. In other words, previous work has shown how to progress syntactically (*i.e.* without execution traces) from viral code consensus identification for a set of variants of the same virus family (training set) to generation of signatures in either a regular expression or rule format for identification of other known variants of the same virus family (test set).

Another related advancement in a syntactic approach was also recently reported [31]. Two different dynamic programming methods, namely, Needleman-Wunsch and SWA were investigated. However, this work was limited to a single polymorphic malware family (JS. Cassandra) and used fixed parameters which were not tuned [31]. It was found that SWA gave the best results with

100% of known variants being identified.

What has improved considerably since the historical view that only semantic analysis will reveal viral signatures is the growth in our knowledge of sequence-based syntactic and structural search algorithms in bioinformatics. Such algorithms do not just search for the presence or absence of characters in certain positions but also use pre-loaded substitution matrices that give substitution probabilities and/or allow such substitution matrices to be generated using probabilistic techniques. Of greater importance to this paper is that such algorithms manipulate (shift) the strings/sequences to allow for insertion and deletion of characters to maximize the number of matching characters. Previous work [32] showed that such string manipulation algorithms from bioinformatics work best with biologically represented strings (amino acids, nucleotide bases) rather than arbitrary character sets. This is due to the possible inclusion of heuristic biological information in the algorithms that determines to some extent the matching process (e.g. built-in information concerning mutation rates between amino acids or nucleotide bases). The implications of rewriting already well-understood and publicly available sequence-based bioinformatics algorithms to work on hex code (numeric data) are not known. For these reasons and to allow comparison with previous work, conversion of hex code to an appropriate biological representation is required before sequence matching, with conversion back to hex code for signature generation. We used a simple identity (ID) substitution matrix for our alignment experiments instead of other well-known biological substitution/mutation matrices, such as BLOSUM (Block Substitution Matrix) and PAM (Point Accepted Mutation). ID provides the most parsimonious method in that no assumptions are made as to how symbols may be related to each other. Also, the use of ID allows the effects of gap opening and closing to be accurately assessed without being compromised by probabilistic substitution matrices.

3. Related Work

Previous research related to this work has primarily focused on worms. Syntactic approaches include Autograph [33], Honeycomb [34] and Early Bird [35], all of which generate signatures that constitute individual adjoining byte strings (tokens). Another syntactic approach is Polygraph [36], which identifies an array of tokens, a subsequence of tokens and Bayes signatures based on probabilistic methods to detect polymorphic worms. Semantic approaches include PAYL [37], which produces subsequence signature tokens that associate ingress/egress payload notifications to detect the initial replication of worms. Other semantic approaches include: Nemean [38], which focuses on identifying signatures that defend against worms; Hamsa [39], which produces a set of signature tokens that can deal with polymorphic worms by investigating their invariant activity; and Botzilla [40], which produces signatures for the malicious activities (traffic) created by a malware binary executed several times within a controlled domain.

Nearly all previous approaches deal with only one malware family and it is currently not known how generalizable these methods are for capturing variants belonging to different families. A future, "smart" AVS needs to generate multiple signatures with very low false positives that can fully capture variants emanating from many different corresponding polymorphic viral families with multiple malicious activities (polymorphic engines). In our approach, new structural variants were generated by us in the laboratory using the information included in documents concerning the corresponding polymorphic viral family (more details in Subsection 5.2). This use of newly generated novel variants differentiates our approach from all previous research that exclusively uses existing malware samples from an online repository.

Other semantic-based research exists for different types of malware, including: ShieldGen [41], which generates network signatures for unseen vulnerabilities that are protocol-aware (for instance, the protocol mode with which an invasive message can be posted); AutoRE [42], which produces a spam signature creation architecture from spam emails that use botnets to detect them; and Wurzinger *et al.*'s [43] approach, which identifies botnets that are under the influence of botmaster (malicious body) using network signatures by examining the response from a compromised host to a received command and by generating detection models. ProVex [44] is also a semantic-based approach which generates signatures to identify botnets that use encrypted command and control (C&C) systems after being given the keys and decryption routine employed by the malware using binary code reuse strategy, and is based on the research proposed by Caballero *et al.*'s approach [45]. FIRMA [46], also a semantic-based approach, can be employed to detect similar C&C systems but does not produce signatures for these. A number of syntactic and semantic-based strategies were proposed by Scheirer *et al.*'s approach [47] for the identification of many polymorphic worms and use intrusion detection techniques such as sliding window schemes and instruction semantics, with further refinements by Scheirer *et al.* In comparison to these semantic-based approaches, we propose a purely syntactic approach which generates variable-length syntactic viral signatures that identify known and unknown variants belonging to a polymorphic viral family, independently of execution traces, and, critically for a syntactic approach, without needing numerous infections for the purpose of malware association.

There has also been some related research on sequence alignment approaches using a semantic approach in other security areas. For instance, sequence alignment was used to identify masquerade detection by comparing "audit data" (actual examples of attempted malicious activity via command line entry using authenticated accounts) with legitimate user signatures extracted from their actual command line entries [48]. Another example is intrusion detection [49], where variable length patterns from training data consisting of system call traces of commands under normal execution were analyzed by a sequence-based algorithm called Teiresias. Other sequence alignment approaches that are based on

semantics include Zhao *et al.*'s [50] approach, which generates signatures through an inverse transcoding method by converting the malware sequential information, such as system call sequences, propagation dataflow, etc., into amino acid sequences and then aligning them using multiple sequence alignment tool. Ki *et al.*'s [51] approach generates sequences that are typical API call sequence motifs of malicious activities belonging to several malware samples and employed multiple sequence alignment tool to align those malware samples to extract signatures. They then used data mining and machine learning algorithms to calculate statistical measures, such as accuracy, precision, etc., to test the extracted signatures but did not test the signatures against new variants. MalGene [52] uses sequence alignment techniques on two sequences of system call events belonging to two different analysis environments: one environment in which the malware evades the AVS, and the other in which it exhibits the malicious activities. These events are used to construct an "evasion signature" using sequence alignment. However, this semantic approach requires system call sequences from both analysis environments which in turn requires the use of system monitoring, which adds an overhead. In contrast, our syntactic approach is independent of any prior semantic knowledge. The syntactic approach most closely related to ours [53] adds nothing new to what was reported by Chen *et al.*'s approach in 2012 [28], and repeats the structural sequence alignment and data mining approaches adopted in that paper and subsequently enhanced by [29] [30] [31].

To conclude this section, previous use of sequence alignment has for the most part been semantic in nature, relying on system behavior patterns rather than code or structural patterns for the identification of malware or fraudulent activity. Also, because of their semantic nature, the generalizability of the results to new variants created through polymorphism is unknown, as is the generalizability, if any, of signatures to malware of different families. Our syntactic-driven approach, on the other hand, is based on the intuition that most new (polymorphic) variants are simple syntactic alterations of existing malware. The "expressive power" of signatures can be evaluated by identifying how well these signatures generalize to new and unseen variants of the same family, all derived through polymorphic (structural) changes to the code, as well as across different families. The advantage of a syntactic approach is that no semantics is required. That is, there is no need for an infection before a signature is generated. Finally, most semantic approaches in the literature do not address the problem of false positive rates. This is because there are many different ways that a program can run and false positive rates may be impossible to quantify for signatures extracted from a limited number of execution traces on one variant of malware. With a syntactic approach, on the other hand, signatures can be checked against static code and objects, including files, without needing to execute any code. For instance, one method of distributing malware is to generate new polymorphic variants and store them undetected in user files until triggered, and syntactic

signatures may be effective in catching such variants before execution. The advantages of a syntactic approach are obvious for future smart AVS technology, but so far there has been very little attempt to analyze the effectiveness of a purely syntactic approach systematically and across different malware families. For instance, the signatures generated from our approach are able to satisfy the false positive rate requisite of 0.1%. More importantly, as will be shown below, the number of malware training examples needed to extract a signature for use against unseen test examples is surprisingly small given the sequence alignment approach adopted in our experiments.

4. Problem Statement

Our previous work [29] [30] [31] has shown that sequence alignment techniques supplemented with Smith-Waterman algorithm lead to signatures that generalized successfully to unseen but previously known variants of polymorphic viruses. This prior work adopted a fixed combination of gap open and gap extend penalties for the automatic generation of virus signatures. However, it is not known how well this method generalizes to new, unknown variants or what the effect of gap penalties is. In this paper, we use ten different combinations of gap open and gap extend penalties to determine whether changes in these penalty parameters can help to identify signatures for known as well as unknown polymorphic variants which we generate in the laboratory, thereby extending the ability of future AVSs to identify variants not previously encountered.

5. Systems and Methods

5.1. Technical Safeguards

Hex (Hexadecimal) dump extraction (Step-1) and testing (Step-8) were undertaken on a stand-alone system to prevent possible unintended infection of other systems. Downloading of polymorphic malware (and known variants) as well as the generation of unknown variants was performed using "Oracle VM VirtualBox" [54] (an x86 software package with virtualization capability) with a pre-installed Linux-based (Ubuntu) operating system image. Due to possible security sensitivity, some of the methods below (Step-1 and Step-8) are not described in detail, especially details concerning generating hex dumps from polymorphic malware, which are omitted. Interested readers are requested to contact the corresponding author, using their academic email addresses, for further information. Our method consists of eight steps (see **Figure 1** below).

5.2. Hex Dump Extraction

JS. Cassandra virus was written in 2003 by a virus author known as 'Second Part To Hell/SPTH' in Austria. Unlike any other JavaScript virus, JS. Cassandra is comprised of four distinct polymorphic engines: polymorphic engine I, which includes Garbage or Junk codes; polymorphic engine II, which modifies its Body (Body Changing); polymorphic engine III, which modifies its Variables (Variable

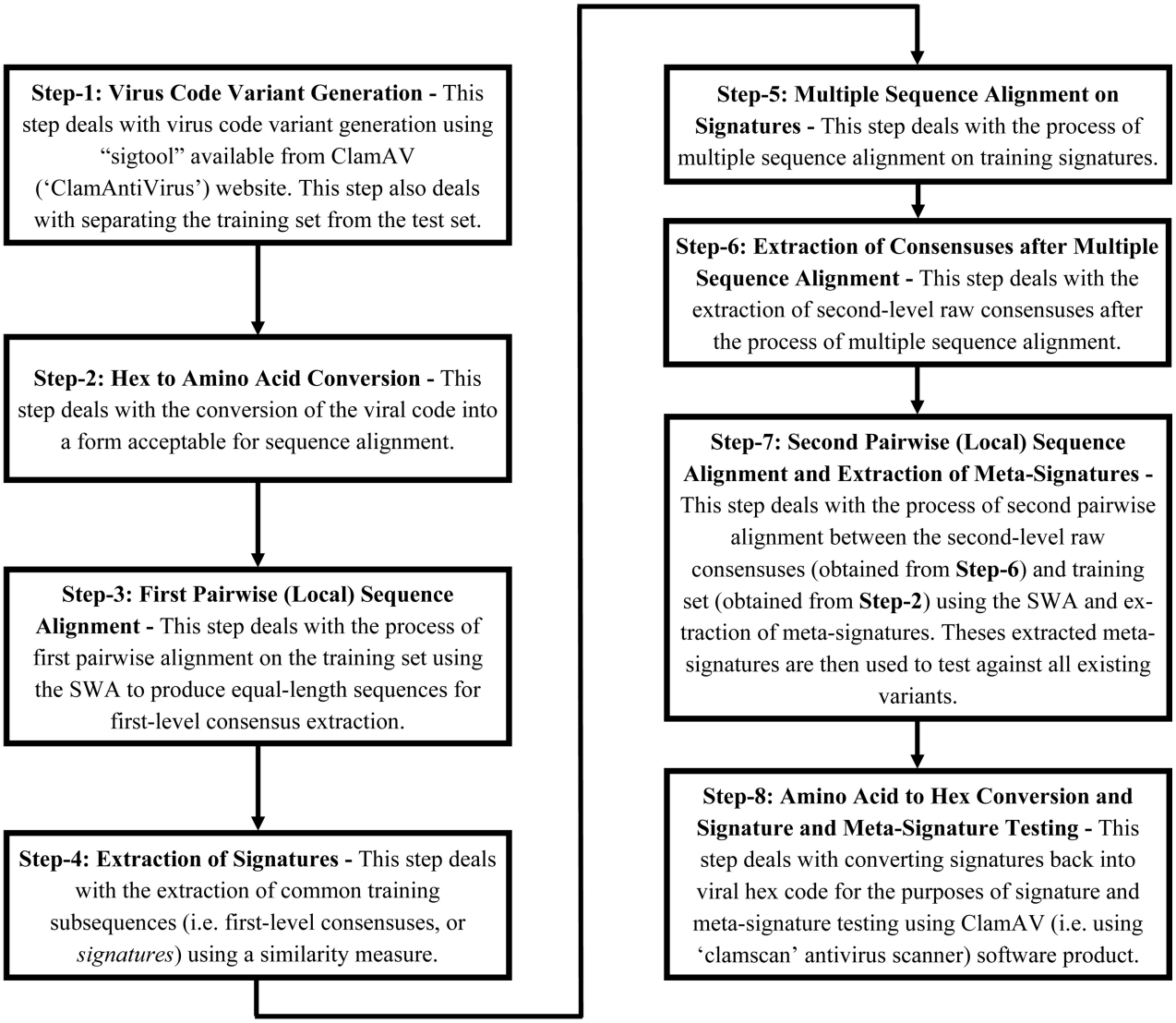

Figure 1. Biosequence analysis method comprising of eight steps.

Changing); and polymorphic engine IV, which modifies its Numbers (Number Changing) [55] [56]. The original JS. Cassandra virus with its source code was downloaded from the virus author's (Second Part to Hell) website [55]. All 351 known (existing) polymorphic variants of the JS. Cassandra virus were successfully retrieved [57].

Win32.Cholera/W32.Cholera/W32.CTX is a polymorphic virus which attacks executable PE (Portable Executable) files and was first identified in 2010. This virus is programmed in assembly language, and it employs an EPO (Entry Point Obfuscation) approach, which makes its identification difficult [58] [59]. The original source files were downloaded from "VX Heaven" [60] website. 198 new polymorphic variants of "W32.Cholera" virus were generated by executing one of the original virus files (in this case, a file named "Virus.Win32.CTX.10853").

Win32.Kitti/W32.Kitti is a polymorphic virus which works with the help of an overlapping code as an obfuscation technique and was first identified in 2011.

This virus modifies its instructions to create new instructions with the same semantics but a different structure using an overlapping code process [61] [62] [63]. The original virus file along with its source code in assembly language was downloaded from the "Second Part to Hell" [64] website. 1105 new polymorphic variants of "W32.Kitti" virus were generated by executing the original virus file (in this case a file named 'oc.exe').

The method consists of 8 steps, summarized as follows. Step-1 deals with virus code variant generation and separating the training set from the test set. Step-2 deals with converting the hex code into a form acceptable for sequence alignment. Because variant generation leads to variable length code, Step-3 deals with the process of first pairwise (local) sequence alignment on the training set using the SWA to produce equal-length sequences for consensus extraction. Gap open and gap close penalties are introduced in this step. Step-4 deals with the extraction of common training subsequences (*i.e.* consensuses, or *signatures*) using a similarity measure. Step-5 deals with the process of multiple sequence alignment on these training signatures. Step-6 deals with the extraction of consensuses after the process of multiple sequence alignment. Step-7 deals with the process of second pairwise (local) sequence alignment between the consensuses (obtained from Step-6) and training set (obtained from Step-2) using the SWA and extraction of meta-signatures. Lastly, Step-8 deals with converting signatures back into viral hex code for the purposes of signature and meta-signature testing. More details concerning each step are provided below.

Summarizing our method, sequence alignment works on variable length viral hex strings to produce equal length hex strings through opening and closing gaps. These equal length strings can be analyzed to produce first-level consensuses (signatures), which represent common subsequences at specific locations for the pairwise alignments. These consensuses/signatures can themselves be analyzed using multiple sequence alignment to produce second-level raw consensuses that can be further analyzed to identify similarities with each other to produce meta-signatures for the six variants in that test family. These meta-signatures are then used to test against all existing variants.

Step-1 (Virus code variant generation): The JS. Cassandra virus and all its known variants were written in the JavaScript programming language, and their source code was readily available. Five variants out of the 351 known variants were taken for our training purposes plus the original "JS. Cassandra.js" virus (a total of six variants). In the case of the W32.CTX virus, five variants out of 198 newly generated polymorphic variants were taken for our training purposes plus the original "Virus.Win32.CTX.10853" virus (a total of six variants). In the case of the W32.Kitti virus, five variants of the 1105 newly generated polymorphic variants were taken for our training purpose as well as the original "oc.exe" virus (a total of six variants). New variant generation was achieved by using information obtained from various sources concerning polymorphic versions (details available on request). The percentage of training to test ratio of training variants

for JS. Cassandra virus is 1.7% (6:352), for W32.CTX virus is 3.01% (6:199), and for W32.Kitti virus is 0.54% (6:1106). A CRC32b hash value was generated for each of these 18 training variants and no duplicates were found, indicating that they were unique. Only six variants (the original plus five variants in each case) are chosen for training (*i.e.* for generating signatures and meta-signatures) in line with previous work [29] [30] [31].

All 18 training variants were checked using the "VirusTotal" [65] (a free online scanner for malware) website to confirm that malicious functionality was preserved in the 18 variants. "VirusTotal" employs 55 well-known AVS products and so provides good assurance that our variant generation for the W32.CTX and W32.Kitti viruses was effective. **Table 1** gives the detection ratio based on the 55 state-of-the-art AVS products obtained from the "VirusTotal" website for the 18 training variants, indicating that on average only 53.69% and 73.33% of the 55 AVS products successfully detected the 15 variants and three original polymorphic viruses, respectively. Hex dumps were then extracted from the 18 variants using "sigtool" (available on the ClamAV ("Clam AntiVirus") [66] website). A severely reduced proportion of training to test samples was used to reflect the current difficulty in identifying signatures that generalize from a small,

Table 1. Detection ratio based on the 55 state-of-the-art AVS products obtained from the VirusTotal website for the 18 malicious variants.

Polymorphic Malware 1	Filename	Detection Ratio
JS. Cassandra Virus	JS. Cassandra.js (Original Virus)	39/55
	v_000.js (Variant 1)	19/55
	v_002.js (Variant 2)	21/55
	v_003.js (Variant 3)	15/55
	v_004.js (Variant 4)	17/55
	v_005.js (Variant 5)	17/55
Polymorphic Malware 2	**Filename**	**Detection Ratio**
W32.CTX/W32.Cholera Virus	W32.CTX.Cholera.Virus.10853 (Original Virus)	38/55
	actmovie.exe (Variant 1)	41/55
	cisvc.exe (Variant 2)	42/55
	dcomcnfg.exe (Variant 3)	37/55
	forcedos.exe (Variant 4)	39/55
	MRT.exe (Variant 5)	39/55
Polymorphic Malware 3	**Filename**	**Detection Ratio**
W32.Kitti Virus	OC.exe (Original Virus)	44/55
	absdmfcj.exe (Variant 1)	41/55
	adehsjud.exe (Variant 2)	41/55
	crilunah.exe (Variant 3)	44/55
	nafybgho.exe (Variant 4)	12/55
	nalgjahg.exe (Variant 5)	18/55

previously encountered set of known variants to a potentially infinite set of new variants.

5.3. Hex to Amino Acid Conversion

Step-2 (Converting the viral code into a form acceptable for sequence alignment): In this step, the extracted 18 hex dump sequences belonging to the three polymorphic malware families were converted into amino acid sequences. Conversion of hexadecimal into amino acid sequences for input to JAligner [67] was performed using the rules shown in Table 2. A short example of the conversion of hexadecimal code into 16 amino acid characters is shown below:

4*d*5*a*800001000000 (16-bit hexadecimal code)

KDLAQGGGGHGGGGGG (16 amino acid characters)

5.4. First Pairwise (Local) Sequence Alignment and Signature Extraction

The string matching SWA was used to perform pairwise local alignment and to extract the most common substring/pattern from the three different families of polymorphic variants. Signature and meta-signature in this section are defined as follows. A signature is a single string (or a common substring/pattern) that can identify a single or (in some cases) a few known and unknown variants, whereas a meta-signature is a string (or a common substring/pattern) that can identify most or all known variants as well as some or all unknown (or new) variants.

Step-3 (First pairwise (local) sequence alignment using the SWA): In this step, a pairwise (local) alignment was performed on all six training strings for each family using the SWA with an ID substitution matrix (*i.e.* alignment was performed through matching in particular positions rather than preloaded biologically informed mutation rates) between two sequential converted amino acid sequences using JAligner [67]. Ten different combinations of gap open and gap extend penalties were used while conducting the pairwise local alignments. A gap penalty of zero means no penalty for any gaps introduced in the alignment

Table 2. Rules for converting hexadecimal into amino acid.

Hexadecimal	Amino Acid	Hexadecimal	Amino Acid
0	G	8	Q
1	H	9	P
2	I	a	A
3	R	b	B
4	K	c	C
5	L	d	D
6	M	e	E
7	N	f	F

[68] [69]. In our case, we have six variants, *i.e.* V1, V2, V3, V4, V5 and V6 (where V1 is the original virus and V2-V6 are its polymorphic variants). For instance, between V1 and V2, ten different combinations of gap open and gap extend penalties were applied, which led to ten different pairwise local alignments. We applied a similar procedure on the remaining four pairs *i.e.* on V2 and V3, V3 and V4, V4 and V5, and V5 and V6, respectively. In total, 150 pairwise local alignments were carried out in this step, 50 for each of the three viruses. In the case of the W32.Kitti virus, only the first 46,000 amino acid characters (*i.e.* around 18.5%) were aligned due to the significantly longer lengths of amino acid sequences belonging to its six variants. In the case of amino acid sequences, JAligner [67] allows pairwise alignment of two sequences of maximum combined sequence length of up to 92,000, only after dedicating the initial Java memory size of 13,312 MB and maximum heap memory size of 15,360 MB to JAligner.

Step-4 (Extraction of signatures): After the local alignment process, common substrings, or signatures, from the pairwise local alignments which had the highest percentage of identities and similarities were extracted (*i.e.* a threshold of 85% and over), resulting in 57 common substrings from the 61 pairwise local alignments. Ten common substrings were extracted from the 26 pairwise local alignments for the JS. Cassandra virus, 17 from the ten pairwise local alignments for W32.CTX/W32.Cholera virus and 30 from the 25 pairwise local alignments for W32.Kitti virus. The minimum and maximum sequence lengths of signatures obtained for JS. Cassandra virus were 53 and 198, respectively, with a mean (sum, median and standard deviation of 1064, 107 and 45.563, respectively) of 106.4 for ten signatures in their amino acid representation. The minimum and maximum sequence lengths of signatures obtained for W32.CTX virus were 30 and 1069, respectively, with a mean (sum, median and standard deviation of 7410, 276 and 397.665, respectively) of 436 for 17 signatures in their amino acid representation. The minimum and maximum sequence lengths of signatures obtained for W32.Kitti virus were 790 and 1868, respectively, with a mean (sum, median and standard deviation of 50,662, 1868 and 407.706, respectively) of 1689 for 30 signatures in their amino acid representation.

5.5. Multiple Sequence Alignment and Consensus Extraction

Step-5 (Multiple sequence alignment on signatures): In this step, a multiple alignment was performed on the signatures (*i.e.* common substrings) obtained in Step-4 using T-Coffee [70] available on the EMBL-EBI website, with alignment being constrained to the ID matrix. In total, three separate multiple alignments were performed (*i.e.* on 10, 17 and 30 signatures, respectively), one for each of the three polymorphic malware types. The main purpose of alignment here is to produce second-level consensuses (more details in Step-6).

Step-6 (Extraction of consensuses after multiple sequence alignment): T-Coffee [70], similar to other alignment tools, produces a consensus sequence that represents the most common residues (amino acid representations) in each

position of multiple sequences after alignment. In this step, the consensus was stored and the process was repeated three times, once for each polymorphic malware. Three consensuses were extracted in this step. One of these consensuses with a sequence length of 203 for the JS. Cassandra virus is shown below in hex representation:

4a4e4f49da598f4a09cad3585d1a0b9c9bdd5b990a13585d1a0b9c985b991bdb4a0a4a8e4e4e4e4a4ac9cf4f49cad3585d1a0b9c9bdd5b990a13585d1a0b9c985b991bdb4a0a4a8e4e4e4e4a4ac9ca49cad4dd1c9a5b99cb999c9bdb50da185c90dbd9194a0

5.6. Second Pairwise (Local) Sequence Alignment and Meta-Signature Extraction

Step-7 (Second pairwise (local) sequence alignment using the SWA and Extraction of meta-signatures): In this step, a pairwise (local) alignment between the consensus and the sequence of the original virus/variant was performed using the SWA with an ID matrix using JAligner [67]. In total, three separate pairwise local alignments were performed, one for each type of polymorphic malware. The fixed combination of gap open (*i.e.* 10) and gap extend (*i.e.* 1) penalty (as used in [29] [30] [31]) was used in this step. The outcome of this alignment is a common substring, or meta-signature, that will be used to detect all the known (and the unknown/new) polymorphic variants of that family. In total, three meta-signatures for JS. Cassandra virus, three meta-signatures for W32.CTX/Cholera virus and five meta-signatures for W32.Kitti virus were extracted in this step. One of the eleven common substrings (*i.e.* the meta-signatures) of sequence length 56 obtained from this step for the JS. Cassandra virus is shown below in hex representation:

28272b4d6174682e726f756e64284d6174682e72616e646f6d28292a

5.7. Amino Acid to Hex Conversion and Meta-Signature (and Signature) Testing

Step-8 (Converting the sequences back into viral hex code and signature testing): In this final step, the eleven meta-signatures from Step-7 (and the 57 signatures obtained in Step-4) in their amino acid sequence representation were converted back to hexadecimal format for testing purposes. The eleven hex meta-signatures and the 57 signatures obtained in Step-4 were tested against the three polymorphic malware types along with their known and unknown variants using ClamAV (*i.e.* Clamscan antivirus scanner) software. One of the eleven hex meta-signatures, with a sequence length 76, obtained from this step for the JS. Cassandra virus is shown below:

393939292b273d3d272b4d6174682e726f756e64284d6174682e72616e646f6d28292a393939

5.8. Summary

By downloading the JS. Cassandra polymorphic virus and its known variants in

its original JavaScript coding as well as generating new (unknown) variants of the other two viruses, the authenticity of the variants has been assured. By checking all 18 training variants against a number of AVS systems, we have provided assurance that these variants are genuinely malicious. The first pairwise alignment was conducted using ten different combinations of gap open and gap extend penalties, and the second pairwise alignment was conducted using a fixed combination of gap open (*i.e.* 10) and gap extend penalty (*i.e.* 1). There were no gap open and gap extend penalty options available for the process of multiple sequence alignment. After signature extraction, all biologically-represented signatures and meta-signatures were converted back to hex code for evaluation (details below). All the signature/meta-signature testing against the polymorphic variants was conducted using the latest version of the Clamscan antivirus scanner [66].

6. Results and Evaluation of State-of-the-Art AVS Products

Table 3 provides the results of the pairwise local alignments that were performed in Step-3. Only the desired pairwise local alignment results with the highest percentage of identities and similarities are shown in Table 3.

From Table 3, it can be seen that the percentages of identities and similarities were higher than 85%, indicating that there were high percentages of the code conserved in the sequences. In the case of W32.Kitti virus, the percentage of identities and similarities was 100%. In the case of W32.CTX virus, the percentages of identities and similarities were over 94% and in some cases 100%. As expected, Table 3 indicates that the amount of gap increases with lower gap open penalties (see Columns "Gap Open Penalty" and "Gaps Percentage"), indicating that the amount of insertions or deletions to maximize the amount of matches was also lower. In previously adopted methods [29] [30] [31] a fixed combination of gap open (*i.e.* 10) and gap extend (*i.e.* 1) penalty was used. The work reported here has instead explored various combinations of gap open and gap extend penalties (conducted in Step-3) to explore the effect of these penalties on variant detection. It can be seen from the results in Table 3 that the percentages of identities and similarities were higher (*i.e.* over 97%) when the gap open and gap extend penalties were higher, indicating that the (pairwise local) alignments were compact, thereby restricting the amount of gaps (with lower gap percentages) and increasing their importance (see Columns "Gap Open Penalty", "Gap Extend Penalty" and "Gaps Percentage" in Table 3).

Tables 4-6 provide the detection rate results for the three malware types along with their known and unknown variants. The detection was carried out using Clamscan and the most effective signatures obtained in Step-4. The most effective signatures were determined to be the signatures that detected over 90% of the variants. These signatures were placed inside our own generated (.ndb) database [29], which is used by Clamscan as a recommended database file format for signature testing purposes. Detection performance for each of the three viruses

Table 3. Results of the pairwise local alignments that were performed in Step-3.

Polymorphic Malware 1	Pairwise Alignment	Gap Open Penalty	Gap Extend Penalty	Identity Percentage	Similarity Percentage	Gaps Percentage	Alignment Length	Alignment Score
JS. Cassandra Virus	Original JS. Cassandra virus and Variant 1	15	1	98.51%	98.51%	1.49%	134	116.00
		20	0.5	98.51%	98.51%	1.49%	134	111.50
		20	1	98.51%	98.51%	1.49%	134	111.00
		25	0.5	100.00%	100.00%	0.00%	108	108.00
		25	1	100.00%	100.00%	0.00%	108	108.00
	Variant 1 and Variant 2	15	1	88.84%	88.84%	11.16%	215	139.00
		20	0.5	88.84%	88.84%	11.16%	215	140.00
		20	1	100.00%	100.00%	0.00%	138	138.00
		25	0.5	100.00%	100.00%	0.00%	138	138.00
		25	1	100.00%	100.00%	0.00%	138	138.00
	Variant 2 and Variant 3	15	1	100.00%	100.00%	0.00%	106	106.00
		20	1	100.00%	100.00%	0.00%	106	106.00
		25	0.5	100.00%	100.00%	0.00%	106	106.00
		25	1	100.00%	100.00%	0.00%	106	106.00
	Variant 3 and Variant 4	10	1	95.19%	95.19%	4.81%	208	170.00
		15	0.5	95.19%	95.19%	4.81%	208	164.00
		15	1	95.19%	95.19%	4.81%	208	160.00
		20	0.5	95.19%	95.19%	4.81%	208	154.00
		20	1	95.19%	95.19%	4.81%	208	150.00
		25	0.5	95.19%	95.19%	4.81%	208	144.00
		25	1	95.19%	95.19%	4.81%	208	140.00
	Variant 4 and Variant 5	10	1	100.00%	100.00%	0.00%	198	198.00
		15	1	100.00%	100.00%	0.00%	198	198.00
		20	1	100.00%	100.00%	0.00%	198	198.00
		25	0.5	100.00%	100.00%	0.00%	198	198.00
		25	1	100.00%	100.00%	0.00%	198	198.00
Polymorphic Malware 2	**Pairwise Alignment**	**Gap Open Penalty**	**Gap Extend Penalty**	**Identity Percentage**	**Similarity Percentage**	**Gaps Percentage**	**Alignment Length**	**Alignment Score**
W32.CTX/W32. Cholera Virus	Original W32.CTX virus and Variant 1	25	1	99.29%	99.29%	0.71%	1553	1507.00
	Variant 1 and Variant 2	5	1	96.15%	96.15%	3.85%	2309	2015.00
	Variant 2 and Variant 3	10	1	96.41%	96.41%	3.59%	2060	1804.00
	Variant 3 and Variant 4	5	1	94.40%	94.40%	5.60%	2017	1707.00
		10	1	100.00%	100.00%	0.00%	736	736.00
	Variant 4 and Variant 5	15	1	100.00%	100.00%	0.00%	736	736.00
		20	0.5	100.00%	100.00%	0.00%	736	736.00

Continued

		20	1	100.00%	100.00%	0.00%	736	736.00
		25	0.5	100.00%	100.00%	0.00%	736	736.00
		25	1	100.00%	100.00%	0.00%	736	736.00
Polymorphic Malware 3	**Pairwise Alignment**	**Gap Open Penalty**	**Gap Extend Penalty**	**Identity Percentage**	**Similarity Percentage**	**Gaps Percentage**	**Alignment Length**	**Alignment Score**
W32.Kitti Virus	Original W32.Kitti virus and Variant 1	5	1	86.35%	86.35%	13.65%	3297	2061.00
		10	1	100.00%	100.00%	0.00%	1868	1868.00
		15	1	100.00%	100.00%	0.00%	1868	1868.00
		20	1	100.00%	100.00%	0.00%	1868	1868.00
		25	1	100.00%	100.00%	0.00%	1868	1868.00
	Variant 1 and Variant 2	10	1	100.00%	100.00%	0.00%	1868	1868.00
		15	1	100.00%	100.00%	0.00%	1868	1868.00
		20	1	100.00%	100.00%	0.00%	1868	1868.00
		25	1	100.00%	100.00%	0.00%	1868	1868.00
	Variant 2 and Variant 3	5	1	88.12%	88.12%	11.88%	3266	2130.00
		10	1	100.00%	100.00%	0.00%	1868	1868.00
		15	1	100.00%	100.00%	0.00%	1868	1868.00
		20	1	100.00%	100.00%	0.00%	1868	1868.00
		25	1	100.00%	100.00%	0.00%	1868	1868.00
	Variant 3 and Variant 4	5	1	88.18%	88.18%	11.82%	3265	2129.00
		10	1	100.00%	100.00%	0.00%	1868	1868.00
		15	1	100.00%	100.00%	0.00%	1868	1868.00
		20	1	100.00%	100.00%	0.00%	1868	1868.00
		25	1	100.00%	100.00%	0.00%	1868	1868.00
	Variant 4 and Variant 5	5	0.5	87.03%	87.03%	12.97%	3285	2349.00
		5	1	90.51%	90.51%	9.49%	3225	2217.00
		10	1	100.00%	100.00%	0.00%	1868	1868.00
		15	1	100.00%	100.00%	0.00%	1868	1868.00
		20	1	100.00%	100.00%	0.00%	1868	1868.00
		25	1	100.00%	100.00%	0.00%	1868	1868.00

was measured using the following metrics: true positive rate (sensitivity), true negative rate (specificity), positive predictive value (precision), detection ratio (accuracy) and F1 score (the harmonic mean of the positive predictive value and true positive rate) and are presented in Tables 4-6. In total, 57 signatures were tested, but only the results using the most effective signatures are shown in Tables 4-6.

The performance of our virus detection method was compared with that of the top commercial products available at the time of the research in 2016 as

Table 4. Detection rates for detection of JS. Cassandra Polymorphic Malware and its known variants by testing the Top Five 2016 [71] state-of-the-art AVS products and our top signatures obtained in Step-4 using Clamscan.

Polymorphic Malware	Pairwise Alignment/AVS Product	Top Five State-of-the-art AVS Products and Our Top Signatures (S)	Detection Ratio (with Accuracy) and Statistical Measures	
JS. Cassandra Virus	AntiVirus Ranked No. 1	Bitdefender Antivirus	Detection Ratio (Accuracy)	1/352 (0.2841%)
			Sensitivity/Recall	0.2841%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	0.5666%
	AntiVirus Ranked No. 2	Kaspersky Anti-Virus	Detection Ratio (Accuracy)	1/352 (0.2841%)
			Sensitivity/Recall	0.2841%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	0.5666%
	AntiVirus Ranked No. 3	McAfee AntiVirus	Detection Ratio (Accuracy)	152/352 (43.18%)
			Sensitivity/Recall	43.18%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	60.31%
	AntiVirus Ranked No. 4	Norton Security	Detection Ratio (Accuracy)	5/352 (1.42%)
			Sensitivity/Recall	1.42%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	2.80%
	AntiVirus Ranked No. 5	F-Secure Anti-Virus	Detection Ratio (Accuracy)	1/352 (0.2841%)
			Sensitivity/Recall	0.2841%)
			Specificity	0.00%
			Precision	100.00%
			F1 Score	0.5666%
	Original JS. Cassandra virus and Variant 1	S1	Detection Ratio (Accuracy)	340/352 (96.59%)
			Sensitivity/Recall	96.59%
			Specificity	0.00%
			Precision	100%
			F1 Score	98.26%
	Variant 1 and Variant 2	S4	Detection Ratio (Accuracy)	339/352 (96.31%)
			Sensitivity/Recall	96.31%
			Specificity	0.00%
			Precision	100%
			F1 Score	98.12%

Continued

		Detection Ratio (Accuracy)	339/352 (96.31%)
		Sensitivity/Recall	96.31%
	S5, S8	Specificity	0.00%
		Precision	100%
		F1 Score	98.12%
		Detection Ratio (Accuracy)	325/352 (92.33%)
		Sensitivity/Recall	92.33%
Variant 2 and Variant 3	S6	Specificity	0.00%
		Precision	100%
		F1 Score	96.01%
		Detection Ratio (Accuracy)	340/352 (96.59%)
		Sensitivity/Recall	96.59%
	S7	Specificity	0.00%
		Precision	100%
		F1 Score	98.26%
		Detection Ratio (Accuracy)	325/352 (92.33%)
		Sensitivity/Recall	92.33%
Variant 3 and Variant 4	S10	Specificity	0.00%
		Precision	100%
		F1 Score	96.01%

reported by the "TopTenReviews" [71] website. The top five AVS products in this listing were tested using the same three viruses along with their known and unknown variants, and the results are presented in **Tables 4-6**.

From **Tables 4-6**, it can be seen that most of our signatures obtained in Step-4 detected the polymorphic variants, except for two of the 57 signatures that detected none of the variants (not shown in **Tables 4-6**). In the case of W32.Kitti virus, for 26 out of the 28 most effective signatures the detection rates were 100% and for the remaining two, the detection rates were over 99% (**Table 6**). In the case of W32.CTX virus, for four out of the eight most effective signatures the detection rates were 100% and for the remaining four, the detection rates were over 91% (**Table 5**). For the JS. Cassandra virus, the detection rates were above 92% using seven of the 12 signatures (**Table 4**). From **Tables 4-6** (based on the detection ratio, accuracy and statistical measures, such as sensitivity, specificity, etc., needed for malware detection), it can also be seen that none of the top five AVS products fully detected the polymorphic variants except for the Kaspersky Anti-Virus, which successfully detected all of the new polymorphic variants of the W32.Kitti virus. In some cases, the top five AVS products could only successfully detect the original virus and none of its variants (either known or unknown).

Table 5. Detection rates for detection of W32.CTX/W32.Cholera Polymorphic Malware and its new/unknown variants by testing the Top Five 2016 [71] state-of-the-art AVS products and our top signatures obtained in Step-4 using Clamscan.

Polymorphic Malware	Pairwise Alignment/AVS Product	Top Five State-of-the-art AVS Products and Our Top Signatures (S)	Detection Ratio (with Accuracy) and Statistical Measures	
W32.CTX/W32. Cholera Virus	AntiVirus Ranked No. 1	Bitdefender Antivirus	Detection Ratio (Accuracy)	176/200 (88.00%)
			Sensitivity/Recall	88.00%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	93.62%
	AntiVirus Ranked No. 2	Kaspersky Anti-Virus	Detection Ratio (Accuracy)	86/200 (43.00%)
			Sensitivity/Recall	43.00%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	60.14%
	AntiVirus Ranked No. 3	McAfee AntiVirus	Detection Ratio (Accuracy)	27/200 (13.50%)
			Sensitivity/Recall	13.50%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	23.79%
	AntiVirus Ranked No. 4	Norton Security	Detection Ratio (Accuracy)	177/200 (88.50%)
			Sensitivity/Recall	88.50%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	93.89%
	AntiVirus Ranked No. 5	F-Secure Anti-Virus	Detection Ratio (Accuracy)	191/200 (95.50%)
			Sensitivity/Recall	95.50%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	97.69%
	Variant 1 and Variant 2	S4	Detection Ratio (Accuracy)	183/200 (91.50%)
			Sensitivity/Recall	91.50%
			Specificity	0.00%
			Precision	100%
			F1 Score	95.56%
	Variant 2 and Variant 3	S7	Detection Ratio (Accuracy)	189/200 (94.50%)
			Sensitivity/Recall	94.50%
			Specificity	0.00%
			Precision	100%
			F1 Score	97.17%

Continued

		Detection Ratio (Accuracy)	189/200 (94.50%)
		Sensitivity/Recall	94.50%
	S12	Specificity	0.00%
		Precision	100%
		F1 Score	97.17%
		Detection Ratio (Accuracy)	200/200 (100.00%)
		Sensitivity/Recall	100.00%
Variant 3 and Variant 4	S13, S15-S16	Specificity	0.00%
		Precision	100.00%
		F1 Score	100.00%
		Detection Ratio (Accuracy)	192/200 (96.00%)
		Sensitivity/Recall	96.00%
	S14	Specificity	0.00%
		Precision	100%
		F1 Score	97.96%
		Detection Ratio (Accuracy)	200/200 (100.00%)
		Sensitivity/Recall	100.00%
Variant 4 and Variant 5	S17	Specificity	0.00%
		Precision	100.00%
		F1 Score	100.00%

The eleven meta-signatures, obtained from Step-7, were tested on the three viruses along with their known and unknown variants using Clamscan by placing these meta-signatures inside our own generated (.ndb) database [29]. **Figure 2** shows that all 352 (accuracy of 100%) JS. Cassandra variants (including the original virus) were successfully detected by the Clamscan antivirus scanner using our .ndb database. One of the three meta-signatures obtained for JS. Cassandra in Step-7 detected all 352 JS. Cassandra variants (output is shown in **Figure 2**). Two of the other three meta-signatures detected 340 out of 352 (with an accuracy of 96.59%) and 15 out of 352 (with an accuracy of 4.26%) JS. Cassandra variants, respectively. **Figure 3** shows that all 200 of the W32.CTX variants (including the two original viruses) were successfully detected by the Clamscan antivirus scanner. **Figure 4** shows that all 1106 of the W32.Kitti variants (including the original virus) were successfully detected by one of the three successful (with 100% accuracy) meta-signatures. The remaining two out of the overall five meta-signatures detected none of the 1106 variants. One of the three meta-signatures obtained for the W32.CTX virus in Step-7 detected all 200 W32.CTX variants (as shown in **Figure 3**) while another detected 189 of the 200 variants (94.5% accuracy). However, the final meta-signature detected only 19 of the 200 W32.CTX variants (9.5% accuracy). None of the scans (as shown in **Figures 2-4**) took

Table 6. Detection rates for detection of W32.Kitti Polymorphic Malware and its new/unknown variants by testing the Top Five 2016 [71] state-of-the-art AVS products and our top signatures obtained in Step-4 using Clamscan.

Polymorphic Malware	Pairwise Alignment/AVS Product	Top Five State-of-the-art AVS Products and Our Top Signatures (S)	Detection Ratio (with Accuracy) and Statistical Measures	
W32.Kitti Virus	AntiVirus Ranked No. 1	Bitdefender Antivirus	Detection Ratio (Accuracy)	324/1106 (29.29%)
			Sensitivity/Recall	29.29%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	45.31%
	AntiVirus Ranked No. 2	Kaspersky Anti-Virus	Detection Ratio (Accuracy)	1106/1106 (100.00%)
			Sensitivity/Recall	100.00%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	100.00%
	AntiVirus Ranked No. 3	McAfee AntiVirus	Detection Ratio (Accuracy)	293/1106 (26.49%)
			Sensitivity/Recall	26.49%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	41.88%
	AntiVirus Ranked No. 4	Norton Security	Detection Ratio (Accuracy)	450/1106 (40.69%)
			Sensitivity/Recall	40.69%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	57.84%
	AntiVirus Ranked No. 5	F-Secure Anti-Virus	Detection Ratio (Accuracy)	333/1106 (30.11%)
			Sensitivity/Recall	30.11%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	46.28%
	Original W32.Kitti virus and Variant 1	S1-S6	Detection Ratio (Accuracy)	1106/1106 (100.00%)
			Sensitivity/Recall	100.00%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	100.00%
	Variant 1 and Variant 2	S7-S10	Detection Ratio (Accuracy)	1106/1106 (100.00%)
			Sensitivity/Recall	100.00%
			Specificity	0.00%
			Precision	100.00%
			F1 Score	100.00%

Continued

Variant 2 and Variant 3	S11, S13-S16	Detection Ratio (Accuracy)	1106/1106 (100.00%)
		Sensitivity/Recall	100.00%
		Specificity	0.00%
		Precision	100.00%
		F1 Score	100.00%
	S12	Detection Ratio (Accuracy)	1105/1106 (99.91%)
		Sensitivity/Recall	99.91%
		Specificity	0.00%
		Precision	100.00%
		F1 Score	99.95%
Variant 3 and Variant 4	S17, S19-S22	Detection Ratio (Accuracy)	1106/1106 (100.00%)
		Sensitivity/Recall	100.00%
		Specificity	0.00%
		Precision	100.00%
		F1 Score	100.00%
	S18	Detection Ratio (Accuracy)	1105/1106 (99.91%)
		Sensitivity/Recall	99.91%
		Specificity	0.00%
		Precision	100.00%
		F1 Score	99.95%
Variant 4 and Variant 5	S23, S25, S27-S30	Detection Ratio (Accuracy)	1106/1106 (100.00%)
		Sensitivity/Recall	100.00%
		Specificity	0.00%
		Precision and F1 Score	100.00%

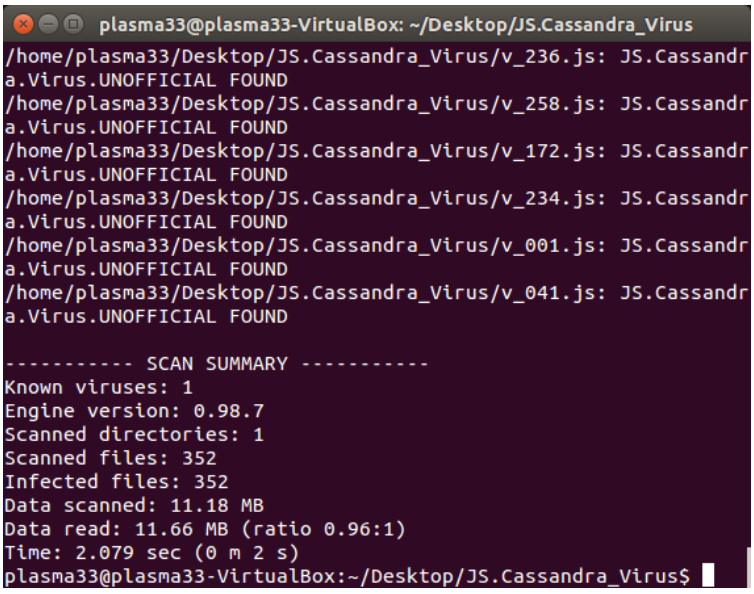

Figure 2. Screenshot of the scan result obtained from Clamscan antivirus scanner for 352 JS. Cassandra viral variants using the meta-signature.

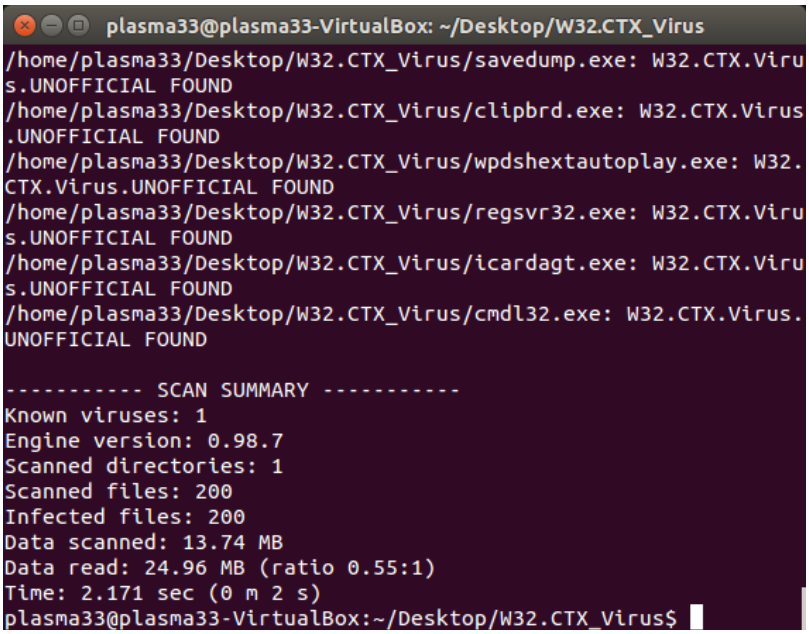

Figure 3. Screenshot of the scan result obtained from Clamscan antivirus scanner for 200 W32.CTX viral variants using the meta-signature.

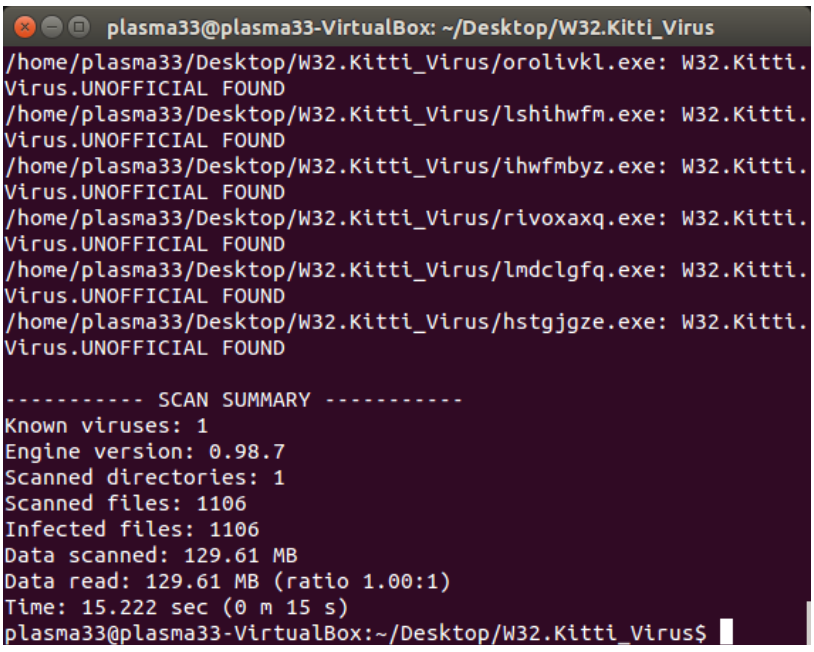

Figure 4. Screenshot of the scan result obtained from Clamscan antivirus scanner for 1106 W32.Kitti viral variants using the meta-signature.

longer than 15 seconds, with most taking just a couple of seconds. Six signatures (*i.e.* three signatures and three meta-signatures) were checked for false positives on 8173 Windows system files: one signature and two meta-signatures obtained for JS. Cassandra virus, one signature and one meta-signature obtained for W32.Kitti virus, and one signature obtained for W32.CTX virus. Figure 5 shows that only two of the 8173 Windows system files were detected as false positives

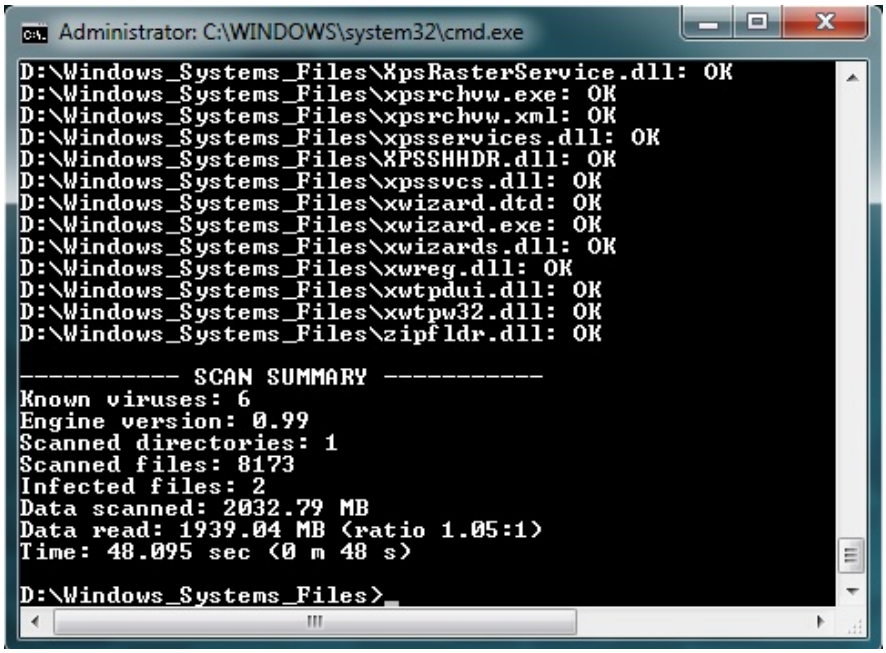

Figure 5. Screenshot of the scan result obtained from Clamscan antivirus scanner for 8173 Windows system files using the six signatures.

(0.024% false positive rate) using the six signatures, satisfying the false positive rate requisite of 0.1%.

7. Conclusions

The aim of our research was to test whether increasingly sophisticated gap open and extend penalties help to produce signatures capable of capturing new polymorphic variants. The results indicate that relatively sophisticated gap penalties captured known variants (training set) of JS. Cassandra virus (see Figure 2). Furthermore, the increasingly sophisticated gap penalties captured unknown variants (test set) of W32.CTX and W32.Kitti viruses, respectively, indicating the feasibility of more sophisticated gap open and gap extend facilities (see Figure 3 and Figure 4). Remarkably, our research demonstrated that it is possible to detect known (training set) as well as unknown (test sets) variants using the training signatures obtained from a very small proportion (typically 3% and below) of training variants of that test family. Detection of test variants using the training signatures could revolutionize our understanding on the detection and generation of polymorphic variants. The three virus families selected are 5 - 11 years old. But as our analysis shows, current AVS products still cannot successfully and consistently identify all their known variants (see Table 1, Table 4, Table 5 and Table 6).

As can be seen from our research, significant concerns exist as to whether modern AVS software systems can or will identify new/unknown (future) variants of polymorphic malware. The ultimate goal for any future, smart AVS would be to identify all potential new/unknown (future) polymorphic variants

utilizing a syntactic method to detect variants both within a virus family as well as across virus families. Our findings show that increasing the gap open and gap extend penalties decreases the number of gaps (in some cases to the point where no gaps exist) in the final alignment (see Columns "Gap Open Penalty", "Gap Extend Penalty" and "Gaps Percentage" in Table 3). Moreover, the signatures obtained from the alignment with few or no gaps have proven to be more effective and successful in detecting known and unknown polymorphic variants than alignment with many gaps. From the results provided in Tables: Tables 3-5, it can be concluded that some of the final alignments, *i.e.*, those with gap percentages of 0.5 or higher, have moderately effective signatures (an accuracy of less than 100%). From the results presented in Table 1 and Table 6, it can be concluded that the final alignments with no gap percentages (*i.e.* 0.00%) have highly effective signatures (*i.e.* with an accuracy of 100%). Most importantly, the results from Table 3 indicate that the conversion of malware code into biological representations has served the task of identifying common code subsequences.

Future work: While gap extend and gap open penalties are used in Step-3 to extract first-level signatures, the effect of such penalties on meta-signature extraction also requires investigation. The meta-signatures generated are currently linear. Conversion of these linear signatures to rule-based templates will need to be undertaken to compress their representation.

Limitations of our study: Our focus on well-known and historic viruses does not take into account the rapid evolution of other forms of malware, such as ransomware and DDoS attacks that involve external manipulation. Furthermore, we do not take into account the unknown (new) variants generated from polymorphic virus construction kits. Building such a library of unknown polymorphic variants will allow us to investigate the impact of a new polymorphic malware detection system in relation to old and existing malware variants. However, nearly all malware has a self-replicating component irrespective of its function. On the assumption that our signatures and meta-signatures are capturing essential aspects of malware replication, the results described here may be applicable to other malware types (not just viruses or worms) that also involve a replication step.

References

[1] Symantec Internet Security Threat Report (2014) Symantec Corporation. http://www.symantec.com

[2] Global Risks 2012: Insight Report (2012) World Economic Forum. http://reports.weforum.org/global-risks-2012/

[3] Kephart, J. and Arnold, W. (1994) Automatic Extraction of Computer Virus Signatures. *Proceedings of the* 4th *Virus Bulletin International Conference*, Abingdon, England, 178-184. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Automatic+extraction+of+computer+virus+signatures&btnG=

[4] Christodorescu, M., Jha, S., Seshia, S.A., Song, D. and Bryant, R.E. (2005) Seman-

tics-Aware Malware Detection. *Proceedings of the IEEE Symposium on Security and Privacy SP '05*, California, 8-11 May 2005, 32-46. https://doi.org/10.1109/sp.2005.20

[5] Sathyanarayanan, V.S., Kohli, P. and Bruhadesgwar, B. (2008) Signature Generation and Detection of Malware Families. In: Mu, Y., Susilo, W. and Seberry, J., Eds., *Information Security and Privacy*, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 336-349. https://doi.org/10.1007/978-3-540-70500-0_25

[6] Ellis, D., Aiken, J.G., Attwood, K.S. and Tenaglia, S.D. (2004) A Behavioral Approach to Worm Detection, *Proceedings of the ACM Workshop on Rapid Malcode* (*WORM*04), Washington, DC, 29 October 2004, 43-53. https://doi.org/10.1145/1029618.1029625

[7] Gao, D., Reiter, M.K. and Song, D. (2006) Behavioral Distance for Intrusion Detection. In: Valdes, A. and Zamboni, D., Eds., *Recent Advances in Intrusion DetectionRAID* 2005, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 63-81. https://doi.org/10.1007/11663812_4

[8] Cesare, S. and Xiang, Y. (2010) Classification of Malware Using Structured Control Flow. *Proceedings of the* 8th *Australasian Symposium on Parallel and Distributed Computing* (*AusPDC*), Brisbane, 1 January 2010, 61-70.

[9] Tang, H., Zhu, B. andRen, K. (2009) A New Approach to Malware Detection. In: Park, J.H., Chen, HH., Atiquzzaman, M., Lee, C., Kim, T. and Yeo, SS., Eds., *Advances in Information Security and Assurance ISA* 2009, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 229-238. https://doi.org/10.1007/978-3-642-02617-1_24

[10] Kinder, J., Katzenbeisser, S., Schallhart, C. and Veith, H. (2005) Detecting Malicious Code by Model Checking. In: Julisch, K. and Kruegel, C., Eds., *Detection of Intrusions and Malware, and Vulnerability Assessment DIMVA* 2005, Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 174-187. https://doi.org/10.1007/11506881_11

[11] Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F. and Nazario, J. (2007) Automated Classification and Analysis of Internet Malware. In: Kruegel, C., Lippmann, R. and Clark, A., Eds., *Recent Advances in Intrusion Detection RAID* 2007. Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 178-197. https://doi.org/10.1007/978-3-540-74320-0_10

[12] Eskandari, M. and Hashemi, S. (2012) A Graph Mining Approach for Detecting Unknown Malware. *Journal of Visual Languages and Computing*, **23**, 154-162.

[13] Chaumette, S., Ly, O. and Tabary, R. (2011) Automated Extraction of Polymorphic Signatures Using Abstract Interpretation. *Proceedings of the* 5th *International Conference on Network and Systems Security* (*NSS*), Milan, 6-8 September 2011, 41-48. https://doi.org/10.1109/icnss.2011.6059958

[14] Zhang, Q. and Reeves, D.S. (2007) MetaAware: Identifying Metamorphic Malware. *Proceedings of the IEEE* 23rd *Annual Computer Security Applications Conference*, Florida, 10-14 December 2007, 411-420. https://doi.org/10.1109/acsac.2007.9

[15] Steinbock, B. and Martini, P. (2009) Classification and Detection of Metamorphic Malware Using Value Set Analysis. *Proceedings of the* 4th *International Conference on Malicious and Unwanted Software*, Quebec, 13-14 October 2009, 39-46.

[16] Griffin, K., Schneider, S., Hu, X. and Chiueh, T. (2009) Automatic Generation of String Signatures for Malware Detection. In: Kirda, E., Jha, S. and Balzarotti, D., Eds., *Recent Advances in Intrusion Detection RAID* 2009. Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 101-120.

https://doi.org/10.1007/978-3-642-04342-0_6

[17] Moser, A., Kruegel, C. and Kirda, E. (2007) Limits of Static Analysis for Malware Detection. *Proceedings of the IEEE* 23rd *Annual Computer Security Applications Conference*, Florida, 10-14 December 2007, 421-430. https://doi.org/10.1109/acsac.2007.21

[18] Rastogi, V., Chen, Y. and Jiang, X. (2014) Catch Me If You Can: Evaluating Android Anti-Malware against Transformation Attacks. *IEEE Transactions on Information Forensics and Security*, **9**, 99-108. https://doi.org/10.1109/TIFS.2013.2290431

[19] Schultz, M.G., Eskin, E., Zadok, E. and Stolfo, S.J. (2001) Data Mining Methods for Detection of New Malicious Executables. *Proceedings of the IEEE Symposium on Security* & *Privacy*, California, 14-16 May 2001, 38-49. https://doi.org/10.1109/secpri.2001.924286

[20] Baldangombo, U., Jambaljav, N. and Horng, S-J. (2013) A Static Malware Detection System Using Data Mining Methods. https://arxiv.org/abs/1308.2831 https://doi.org/10.5121/ijaia.2013.4411

[21] Komashinskiy, D. and Kotenko, I. (2010) Malware Detection by Data Mining Techniques Based on Positionally Dependent Features. *Proceedings of the* 18th *Euromicro Conferences on Parallel, Distributed and Network-based Processing*, Pisa, 17-19 February 2010, 617-623. https://doi.org/10.1109/pdp.2010.30

[22] Tabish, S.M., Shafiq, M.Z. and Farooq, M. (2009) Malware Detection Using Statistical Analysis of Byte-Level File Content. *Proceedings of the* 15th *ACM SIGKDD Workshop on Cybersecurity and Intelligence Informatics*, Paris, 28 June - 1 July 2009, 23-31. https://doi.org/10.1145/1599272.1599278

[23] Abou-Assaleh, T., Cercone, N. and Sweidan, R. (2004) Detection of New Malicious Code Using N-Grams Signatures. *Proceedings of the* 2nd *Annual Conference on Privacy, Security and Trust*, New Brunswick, 13-15 October 2004, 13-15.

[24] Kolter, J.Z. and Maloof, M.A. (2006) Learning to Detect and Classify Malicious Executables in the Wild. *Journal of Machine Learning Research*, **7**, 2721-2744.

[25] Shafiq, M.Z., Tabish, S.M. and Farooq, M. (2008) Embedded Malware Detection Using Markov N-Grams. *Proceedings of the* 5th *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Paris, 10-11 July 2008, 88-107. https://doi.org/10.1007/978-3-540-70542-0_5

[26] Skoudis, E. and Zeltser, L. (2004) Malware: Fighting Malicious Code. United States: Prentice Hall Professional, New Jersey.

[27] Ferrie, P. and Ször, P. (2001) Hunting for Metamorphic. *Virus*, 123-143. https://pdfs.semanticscholar.org/4e07/1925c789610a6c4db9d460b5d45b4f1ec861.pdf

[28] Chen, Y., Narayanan, A., Pang, S. and Tao, B. (2012) Malicioius Software Detection Using Multiple Sequence Alignment and Data Mining. *Proceedings of the* 2012 *IEEE* 26th *International Conference on Advanced Information Networking and Applications* (*AINA*), 26-29 March 2012, 8-14. https://doi.org/10.1109/AINA.2012.62

[29] Naidu, V. and Narayanan, A. (2016) A Syntactic Approach for Detecting Viral Polymorphic Malware Variants. In: Chau, M., Wang, G. and Chen, H., Eds., *Intelligence and Security Informatics PAISI* 2016. Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 146-165. https://doi.org/10.1007/978-3-319-31863-9_11

[30] Naidu, V. and Narayanan, A. (2016) Using Different Substitution Matrices in a String-Matching Technique for Identifying Viral Polymorphic Malware Variants. *Proceedings of the* 2016 *IEEE Congress on Evolutionary Computation* (*WCCI-IEEE*

CEC), Vancouver, 24-29 July 2016, 2903-2910.
https://doi.org/10.1109/cec.2016.7744156

[31] Naidu, V. and Narayanan, A. (2016) Needleman-Wunsch and Smith-Waterman Algorithms for Identifying Viral Polymorphic Malware Variants. *Proceedings of the* 14th *IEEE International Conference on Dependable, Autonomic and Secure Computing* (*DASC*), Auckland, 8-12 August 2016, 326-333.
https://doi.org/10.1109/dasc-picom-datacom-cyberscitec.2016.73

[32] Naidu, V. and Narayanan, A. (2014) Further Experiments in Biocomputational Structural Analysis of Malware. *Proceedings of the* 10th *IEEE International Conference on Natural Computation* (*ICNC*), 19-21 August 2014, 605-610.
https://doi.org/10.1109/icnc.2014.6975904

[33] Kim, H.-A. and Karp, B. (2004) Autograph: Toward Automated, Distributed Worm Signature Detection. *SSYM*'04 *Proceedings of the* 13th *conference on USENIX Security Symposium*, San Diego, CA, 9-13 August, **13**, 19-19.

[34] Kreibich, C. and Crowcroft, J. (2004) Honeycomb: Creating Intrusion Detection Signatures Using Honeypots. *ACM SIGCOMM Computer Communication Review*, **34**, 51-56. https://doi.org/10.1145/972374.972384

[35] Singh, S., Estan, C., Varghese, G. and Savage, S. (2004) Automated Worm Fingerprinting. *OSDI*'04 *Proceedings of the* 6th *conference on Symposium on Operating Systems Design* & *Implementation*, San Francisco, CA, 06-08 December, **6**, 4.

[36] Newsome, J., Karp, B. and Song, D. (2005) Polygraph: Automatically Generating Signatures for Polymorphic Worms. *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 8-11 May, 226-241. https://doi.org/10.1109/SP.2005.15

[37] Wang, K., Cretu, G. andStolfo, S.J. (2005) Anomalous Payload-Based Worm Detection and Signature Generation. *RAID*'05 *Proceedings of the* 8th *international conference on Recent Advances in Intrusion Detection*, Seattle, WA, 7-9 September, 227-246. https://doi.org/10.1007/11663812_12

[38] Yegneswaran, V., Giffon, J. T., Barford, P. andJha, S. (2005) An Architecture for Generating Semantic Aware Signatures. *SSYM*'05 *Proceedings of the* 14th *Conference on USENIX Security Symposium*, Baltimore, MD, 31 July-05 August, **14**, 34-43.

[39] Li, Z., Sanghi, M., Chen, Y., Kao, M.-Y. and Chavez, B. (2006) Hamsa: Fast Signature Generation for Zero-Day Polymorphic Worms with Provable Attack Resilience. *IEEE Symposium on Security and Privacy*, 21-24 May, 32-47.
https://doi.org/10.1109/SP.2006.18

[40] Rieck, K., Schwenk, G., Limmer, T., Holz, T. and Laskov, P. (2010) Botzilla: Detecting the Phoning Home of Malicious Software. *SAC*'10 *Proceedings of the* 2010 *ACM Symposium on Applied Computing*,Sierre, Switzerland, 22-26 March, 1978-1984.

[41] Cui, W., Peinado, M., Wang, H. J. andLocasto, M. E. (2007) Shieldgen: Automatic-data Patch Generation for Unknown Vulnerabilities with Informed Probing. *IEEE Symposium on Security and Privacy*, Berkeley, CA, USA, 20-23 May, 252-266.
https://doi.org/10.1109/SP.2007.34

[42] Xie, Y., Yu, F., Achan, K., Panigrahy, R., Hulten, G. andOsipkov, I. (2008) Spamming Botnets: Signatures and Characteristics. *ACM SIGCOMM Computer Communication Review*, **38**, 171-182. http://doi.acm.org/10.1145/1402946.1402979

[43] Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C. andKirda, E. (2009) Automatically Generating Models for Botnet Detection. *ESORICS*'09 *Proceedings of the* 14th *European Conference on Research in Computer Security*, Saint-Malo, France,

21-23 September, 232-249.

[44] Rossow, C. and Dietrich, C. J. (2013) Provex: Detecting Botnets with Encrypted Command and Control Channels. *DIMVA'13 Proceedings of the 10th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Berlin, Germany, 18-19 July, 21-40. https://doi.org/10.1007/978-3-642-39235-1_2

[45] Caballero, J., Johnson, N.M., McCamant, S. and Song, D. (2009) Binary Code Extraction and Interface Identification for Security Applications. *Technical Report, DTIC Document*, No. UCB/EECS-2009-133.

[46] Rafique, M. Z. and Caballero, J. (2013) Firma: Malware Clustering and Network Signature Generation with Mixed Network Behaviors. *RAID* 2013 *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions, and Defenses*, Rodney Bay, St. Lucia, 23-25 October, **8145**, 144-163. https://doi.org/10.1007/978-3-642-41284-4_8

[47] Scheirer, W. and Chuah, M.C. (2008) Syntax vs. Semantics: Competing Approaches to Dynamic Network Intrusion Detection. *International Journal of Security and Networks*, **3**, 24-35. https://doi.org/10.1504/IJSN.2008.016199

[48] Coull, S.E. and Szymanski, B.K. (2008) Sequence Alignment for Masquerade Detection. *Computational Statistics & Data Analysis*, **52**, 4116-4131.

[49] Wespi, A., Dacier, M. and Debar, H. (1999) An Intrusion-Detection System Based on the Teiresias Pattern-Discovery Algorithm. *IBM Thomas J. Watson Research Division*.

[50] Zhao, Y., Tang, Y., Wang, Y. and Chen, S. (2013) Generating Malware Signature Using Transcoding from Sequential Data to Amino Acid Sequence. *International Conference on High Performance Computing and Simulation* (*HPCS*), Helsinki, Finland, 1-5 July, 266-272. https://doi.org/10.1109/HPCSim.2013.6641425

[51] Ki, Y., Kim, E. and Kim, H.K. (2015) A Novel Approach to Detect Malware Based on API Call Sequence Analysis. *International Journal of Distributed Sensor Networks*, **2015**, Article No. 4. https://doi.org/10.1155/2015/659101

[52] Kirat, D. and Vigna, G. (2015) Malgene: Automatic Extraction of Malware Analysis Evasion Signature. *CCS'15 Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, Denver, Colorado, USA, 12-16 October, 769-780. https://doi.org/10.1145/2810103.2813642

[53] Kumar, V. and Mishra, S.K. (2013) Detection of Malware by Using Sequence Alignment Strategy and Data Mining Techniques. *International Journal of Computer Applications*, **62**. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Detection+of+Malware+by+Using+Sequence+Alignment+Strategy+and+Data+Mining+Techniques&btnG=

[54] Oracle VM VirtualBox (2016) VirtualBox. https://www.virtualbox.org/

[55] JS.Cassandra by Second Part To Hell (2014) rRlF#4 (Redemption). http://spth.virii.lu/rrlf4/rRlf.13.html

[56] Tutorials-Win32 Polymorphism (2014) VX Heavens. http://vxer.org/

[57] Viruses (2004) Second Part To Hell's Artworks-VIRUSES. http://spth.virii.lu/

[58] Win32/CTX.6889.A (2004) NOD21 Website. http://www.nod21.com

[59] W32/CTX-A (2015) SOPHOS Website. https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/W32~CTX-A/detailed-analysis.aspx

[60] Computer Virus Collection/Virus.Win32.CTX (2 Files)—VX Heaven (2009) Virus

Collection (VX Heaven). http://vxer.org/vl.php?dir=Virus.Win32.CTX

[61] Second Part to Hell's Artworks-VIRUSES (2013) Second Part To Hell's Artworks. http://spth.virii.lu/virii.htm

[62] Second Part to Hell's Artworks-INDEX (2014) Second Part To Hell's Artworks. http://spth.virii.lu/main.htm

[63] Valhalla 4 Announcement (2013) VX Heaven Forum. http://vxer.org/

[64] Viruses: w32.kitti.rar (2013) Second Part To Hell's Artworks. http://spth.virii.lu/w32.kitti.rar

[65] VirusTotal (2016) Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/

[66] ClamavNet (2015) ClamAV® is an Open Source Antivirus Engine for Detecting Trojans, Viruses, Malware & Other Malicious Threats. https://www.clamav.net/

[67] Moustafa, A. (2010) *JAligner: Java Implementation of the Smith-Waterman Algorithm for Biological Sequence Alignment*. Retrieved from SourceForge: http://jaligner.sourceforge.net/

[68] Clustal (2012) *Clustal: Multiple Sequence Alignment*. Retrieved from Clustal: http://www.clustal.org/download/clustalw_help.txt

[69] Yan, R., Wang, X., Huang, L., Lin, J., Cai, W. and Zhang, Z. (2014) GPCRserver: An Accurate and Novel G Protein-Coupled Receptor Predictor. *Molecular BioSystems*, **10**, 2495-2504. https://doi.org/10.1039/C4MB00272E

[70] Notredame, C., Higgins, D. G., & Heringa, J. (2000) T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. *Journal of Molecular Biology*, **302**, 205-217. Retrieved from EMBL-EBI: http://www.ebi.ac.uk/Tools/msa/tcoffee/ https://doi.org/10.1006/jmbi.2000.4042

[71] TopTenReviews (2016) Top 10 Best Antivirus Software for 2016—Top Ten Reviews. http://www.toptenreviews.com/