Characterizing Virus Replication

Jose Andre Morales, Peter J. Clarke, Yi Deng
School of Computing and Information Sciences
Florida International University
Miami, Fl 33199
{jmora009, clarkep, deng} @cis.fiu.edu

Abstract

In this paper we present a formal characterization of virus replication. Two detection models for virus replication are developed, one using operation sequence matching and the other using frequency measures. The paper shows that virus replication can be identified and used to detect known and unknown viruses. An overview of testing for both models are given along with preliminary results confirming that virus replication can be used to detect known and unknown viruses.

1 Introduction

In 2006, an FBI survey reported computer viruses as the number one cause of financial loss for American companies [9]. At over \$32 million dollars, viruses accounted for over 70% of all financial losses for the corporations surveyed. For the same year Kaspersky Labs reported a strong rise in the number of new viruses and more momentum in the second half of the year with email worms topping the list [1]. Kaspersky also forecasts that viruses will increasingly appear, helping to spread other forms of malware and use more sophisticated techniques to avoid detection. Despite this growing problem antivirus companies continue to use signature databases as their primary tool for virus detection. In 2006 Kaspersky labs averaged 10,000 new record updates to its signature database per month and 200 new malware samples per day [6]. Antivirus companies still average 6 hours to release a solution to newly discovered viruses [2, 19]. It is clear that antivirus companies will continue to not properly and quickly handle the ever growing virus problem using signature databases as the centerpiece of detection. A possible solution to this is behavior based detection [14].

Behavior based detection monitors process execution. A process is flagged as a possible virus if the behavior of the process is similar to known viruses. This approach has the advantage of being able to detect new undiscovered viruses. The main drawback to this approach is it can also flag benign processes as being a possible virus which result in false positives. Several detection methods have been proposed where each one uses a specific aspect of viruses to base their detection methodology. These methodologies usually rely on virus characteristics that are not consistently found in all viruses. The result is the methodologies are limited to detecting specific classes of viruses or detection under specific conditions. In this paper we characterize virus replication, which is the fundamental characteristic present in all viruses, with a finite state machine. A malware classified as a virus under the accepted definitions implies the virus has the ability to replicate. We assume the process used by a virus to replicate itself is relatively similar across all viruses. We present two virus replication detection models. The first model uses strings representing an ordered sequence of execution of operations from the replication process of known viruses. These sequences are searched in other processes to determine if the process is a possible virus or benign. The second model uses metrics representing the frequency of use of operations in the replication process of known viruses. Frequency measurements are recorded for other processes and compared to these metrics to determine if the process is a possible virus or benign.

The contributions of this research are:

1. **Characterizing virus replication.** Presenting an FSA to characterize virus replication creates a foundation for future research in this area. The FSA can be used to build new detection models and develop novel theoretical results.
2. **Showing that virus replication can be detected.** The ability to detect virus replication allows for multiple detection models to be proposed. Detecting virus replication becomes a new detection vector that can be used in current antivirus solutions.
3. **Using replication to detect unknown viruses.** Detecting virus replication to stop unknown viruses may prevent major virus epidemics. Just like other behavior based approaches, the ability to stop unknown viruses relieves some pressure from antivirus companies to release effective solutions to newly discovered viruses as quickly as possible.

The remainder of this paper is organized as follows: Sections 2 and 3 present background and related work. In Section 4 a formal characterization of virus replication is built using a finite state machine (FSA) and based on Cohen's formal definition [8] of a virus. Section 5 presents two detection models for virus replication using the characterization. Section 6 is an overview of testing currently being conducted along with some preliminary results. Conclusions and future work are presented in Section 7.

2 Background

Formal models of viruses have been presented in [8, 11, 10]. Each of these seminal works describe virus replication in some form. Cohen provides the foundational results using Turning Machines to illustrate the replication process of a virus as symbols on a tape transferred from one segment of the tape to another segment of the same tape. Adleman defined infection as the replication aspect of a virus using recursive functions. Von Neumann created a self reproducing automata showing that replication can be defined formally with computational models. Many detection models, both signature and behavior based have been created using these seminal papers, an excellent summary of these is found in [5, 14]. Using sequences of execution of operations to detect malware has been proposed in [16, 18, 12, 17, 4, 21, 15, 3]. In each research, the detection is based on anomalous detection or misuse detection [12]. The two approaches record the complete behavior via sequence of operations of a benign or malicious process. Detection is achieved by identifying differences in executing processes and the recorded sequences.

3 Related Work

In [16], anomalous intrusion detection was performed via system call monitoring. A database was trained to recognize the normal behavior of benign processes in a system. The system calls made by a process were compared to the database, if the process made system calls not matching the database, the process was marked as anomalous. The main drawback to this approach is determining how much exhaustive training must be performed to cover all cases of a benign process. If a sequence is left out, this can lead to false positives by identifying normal behavior as anomalous. My research takes a similar approach of monitoring sequences of executed operations to detect viruses. The difference of my approach is the training is done on sequences of executions of operations representing the replication of the virus. Since replication is the fundamental characteristic of a virus, retraining of the database is reduced since the behavior may not vary. This is an improvement on the approach in [16] where training must be repeated for each new program.

In [7], Ellis presented a method to detect worms in a computer network using behavioral signatures. The approach taken to detect worms relied on behavioral patterns of worms reflective of the network communications typical of worms. These behaviors were developed from the definition of a worm. The method was shown capable of detecting classes of worms without a priori knowledge of any specific worm. My research is similar in using the definition of a virus for detection. I use the fundamental characteristic of replication for detecting a virus. My research uses a singular characteristic of a virus as opposed to Ellis which uses multiple characteristics of a worm. Using a singular characteristic results a more focused detection which may be faster than a detection using multiple characteristics.

4 Characterizing Replication

The strict definition of a virus is a program that infects other programs by modifying them to include a possibly evolved version of itself [8]. A less strict version defines a virus as a program that recursively and explicitly copies a possibly evolved version of itself [14]. Both of these definitions express replication as the qualifying fundamental characteristic of a virus. Under these definitions, a malware program is classified as a virus if and only if it has the ability to replicate. It can be inferred that replication is the only characteristic of a virus consistently present in all viruses. Cohen's formal definition of a viral set using a Turing Machine illustrates virus replication as symbols on a tape being read and written to another area further down the tape [8, pg.164]. The symbols being read and written represent the virus. The definition shows read, search and write operations as essential for virus replication to occur. In addition the strict definition of a virus implies the use of open and close operations. To infect a program a virus may have to first gain access to it which may require an open operation. Once infection is completed the newly infected file may have to be closed to become usable by a system. Cohen states that any element of a viral set can produce any number of other elements of the set depending on the remaining tape [8, pg. 165]. This implies that a virus may attempt to replicate several times during one execution which results in a high frequency use of certain operations.

We can now state that virus replication consists of a sequence of execution of some combinations of the following general operations: *open*, *read*, *write*, *search* and *close.* The frequency of execution of this sequence can be one or more times during one execution of a virus. A virus may also execute several different sequences of execution one or more times to attempt replication. Each general operation causes the virus to transition into a new state during the replication process. Each virus executes specific operations during replication that can be classified under one of the general operations stated above. A characterization of virus replication can be stated as a sequence of executions E known as a *replication sequence* with a set of specific operations $p \in P$ and a set of replication states $Q = \{o, r, w, s, c\}$. The members of Q are the replication states a virus transitions into when one of the above general operations are executed. The replication states are defined as follows: o = *opened*, r = *read*, w = *written*, s = *searched*, c = *closed.* It is trivial to see which general operation transitions into each replication state. The members of P, the replication set, are the specific operations executed by the virus that produce a transition into a replication state in Q. A replication sequence E_i is written as $e_1, e_2, e_3, ..., e_n$ with n being the total number of executed operations. The subscript i initialized to 1 uniquely identifies each replication sequence. Each $e \in E$ is one execution of a specific operation p resulting in a new replication state $q \in Q$. This characterization is formally defined with a *finite state automata (FSA)* in Figure 1.

FSA E is a 5-tuple $(\sum, Q, s, f, \delta)$ where:

- $\sum$ is the *alphabet* of E. Elements of $\sum$ are specific operations p belonging to the *replication set* P.
- Q is the finite set of *replication states* $\{o, r, w, s, c\}$
- $s \in Q$ is the *start state* of E
- $f \in Q$ is the *final state* of E
- $\delta : Q \text{ x } \sum \rightarrow Q$

Figure 1: Virus Replication Sequence FSA

The members of $\sum$ must solely consist of specific operations that when executed individually, transition to a replication state $q \in Q$. There is no specific start and final replication states for E. When a virus attempts replication it may start in any replication state $q \in Q$. A virus may also exit replication from any state in Q. No specific $p \in P$ needs to be executed first to start replication and executed last to exit, it can commence and finish with the execution of any operation p belonging to replication set P. Thus all the replication states of Q are valid for the start and final state of E. A virus executes many operations during one complete execution of itself. Only a subset of these operations belong to the replication set P, namely those that transition the virus to a replication state q. FSA E is meant to capture only the ordered sequence of executions of operations $p \in P$. It does not capture other executed operations not belonging to P even if those operations were executed in between operations that do belong to the replication set P. FSA E captures strictly the replication process of a virus and nothing else. A replication sequence E_i comes from a process that is already known to be a virus or from a process that is detected as exhibiting virus replication behavior. If FSA E produces a replication sequence belonging to a benign process it is not labeled and discarded as invalid. Figure 2 is a sample production of FSA E abstractly showing a replication sequence of specific operations $p \in P$ and the corresponding replication state $q \in Q$.

$$E_i = \text{start} \xrightarrow{p_1} q_1 \xrightarrow{p_2} q_2 \xrightarrow{p_3} q_3 \xrightarrow{p_4} q_4 \xrightarrow{p_5} q_5$$

Figure 2: Abstract Replication Sequence E_i

The subscripts assigned to the operations p and the states q in Figure 2 were added only to show correspondence with the operations p. They are not part of FSA E and are not required.

5 Replication Detection Models

Two models for detecting virus replication are presented in this section. Both are based on the characterization and FSA presented in Section 4. The characterization defines virus replication as a replication sequence consisting of operations belonging to the replication set P and cause a transition to a replication state $q \in Q$. A replication sequence can occur many times in one execution of a virus. A virus can also execute many different replication sequences. Detecting frequency of use and replication sequences may be used to identify virus replication and differentiate from benign replication. Recall from Section 4 that FSA E captures only the replication process of a virus. In the detection models presented here FSA E is used to capture the replication sequence of a process. When the detection starts it is not known if a process is a virus or benign. If the process is detected as a virus or exhibiting virus replication behavior it is a valid replication sequence and labeled E_i. If the process is not detected as a virus it is assumed benign, not labeled and not used any further in the detection.

5.1 Operation Sequence Detection Model

This model searches for an encoded string of the operations p of a virus replication sequence. The alphabet $\sum$ is composed of all specific operations executed by a virus during the replication process. As an example, assume $\sum = \{$openfilex, readfilex, copytofile, createfilenew, finddir, getfileattrib, closefile, writefilex, setfilepointer$\}$. All the members of $\sum$ result in a transition to a replication state $q \in Q$. This is illustrated with the mapping of operations to states in Table 1.

Table 1: Operation to State Mapping

Replication State	Operation Name
opened	`openfilex`
read	`readfilex` `setfilepointer`

written	`copytofile`
	`createfilenew`
	`writefilex`

searched	`finddir` `getfileattrib`
closed	`closefile`

We encode the string by assigning one unique character for each specific operation. The character can be a single letter or a single digit. The encoding for our example alphabet is in Table 2.

Table 2: Operation Encoding

Operation Name	Encoded Character
`openfilex`	`O`
`readfilex`	`R`
`setfilepointer`	`S`
`copytofile`	`C`
`createfilenew`	`T`
`writefilex`	`W`
`finddir`	`F`
`getfileattrib`	`G`
`closefile`	`L`

A complete replication sequence for a specific virus could be: openfilex, readfilex, setfilepointer, writefilex, writefilex, readfilex, finddir, copytofile. The encoded string would be ORSWWRFC. Based on Table 1, FSA E would produce the complete replication sequence in Figure 3.

$$E_1 = \text{start} \xrightarrow{O_1} \textit{opened} \xrightarrow{R_2} \textit{read} \xrightarrow{S_3} \textit{read} \xrightarrow{W_4} \textit{written} \xrightarrow{W_5} \textit{written} \xrightarrow{R_6} \textit{read} \xrightarrow{F_7} \textit{searched} \xrightarrow{C_8} \textit{closed}$$

Figure 3: Complete Replication Sequence E_1

This model is implemented in four steps:

1. Build a training set of random virus samples
2. Record the complete replication sequence of each virus
3. Extract replication subsequences
4. Match replication subsequences in a process to detect virus replication behavior

Steps 1-3 is the training session, step 4 is the detection session.

Build a training set of random virus samples. A set of virus binaries of an arbitrary sample size needs to be built to train the detection model. The set should meet any established criteria such as detecting a specific class or family of virus. If your detecting a specific class of virus, for example Peer-to-Peer worms, then your training set should consist of random samples of only Peer-to-Peer worms.

Record the complete replication sequence of each virus. Each virus is run once and the specific operations used in the replication process are recorded in the order of execution. Analyzing the execution of a virus can be done at varying degrees of granularity. Each degree can reveal more or less detailed information about the operations being used. A choice must be made of the desired level of granularity for analyzing and recording the replication process of a virus. With the granularity chosen a comprehensive review of the operations used must be done to populate the replication set P. This involves possibly reviewing specifications on the operating system, hardware and applications to derive a complete list of operations. The specific operations $p \in P$ are those used at the chosen granularity that transition the virus into a replication state q. The replication set P must be complete with only all those specific operations that cause a transition to a replication state. This will ensure correct recording of the replication process and avoid recording operations not belonging to the replication set P. FSA E will run with $\sum$ = replication set P and Q = replication states $\{o, r, w, s, c\}$. As the virus executes, E will transition on each executed operation in $\sum$ to a replication state in Q. When the virus is done executing, E will have produced a complete replication sequence E_i for the specific virus, an example of which is in Figure 3. The sequence of operations $p_1 ... p_n$ are extracted from E_i, encoded, converted to a string and recorded.

Extract replication subsequences. This step identifies replication subsequences that occur in multiple viruses of the training set. The goal of this step is to create a set of replication subsequences that are found in more than one training set virus. A replication subsequence found in more than one virus in the training set may indicate a high probability of being found in other viruses outside of the training set. Subsequences not found in more than one virus of the training set are discarded. For each replication sequence, create all possible subsequences between an arbitrary minimum and maximum size and attempt to match them in at least one other training set virus. If a match is made the subsequence is recorded. The resulting set contains replication sequences $E_1 ... E_i$ where i is the last subsequence and the total number recorded subsequences. If a sequence E_j is a subsequence of a sequence E_i, it is labeled E_{ji}, where j is its unique identifier and i is the unique identifier of the parent sequence. Recall that a replication sequence E_i is an ordered sequence of executions where each $e \in E$ is an execution of a specific operation p. Subsequences containing the last executed operation p_n of a sequence E_i should always have p_n as the last operation in the subsequence. As an example, Figure 4 has two valid and invalid subsequences of the replication sequence E_1 shown in Figure 3.

Valid Subsequences
$E_{21} = \overset{R_2}{\rightarrow} \mathit{read} \overset{S_3}{\rightarrow} \mathit{read} \overset{W_4}{\rightarrow} \mathit{written}$
$E_{31} = \overset{S_3}{\rightarrow} \mathit{read} \overset{W_4}{\rightarrow} \mathit{written} \overset{W_5}{\rightarrow} \mathit{written} \overset{R_6}{\rightarrow} \mathit{read}$

Invalid Subsequences
$E_{41} = \overset{F_7}{\rightarrow} \mathit{searched} \overset{C_8}{\rightarrow} \mathit{closed} \overset{O_1}{\rightarrow} \mathit{opened} \overset{R_2}{\rightarrow} \mathit{read}$
$E_{51} = \overset{W_5}{\rightarrow} \mathit{written} \overset{R_6}{\rightarrow} \mathit{read} \overset{F_7}{\rightarrow} \mathit{searched} \overset{C_8}{\rightarrow} \mathit{closed} \overset{O_1}{\rightarrow} \mathit{opened} \overset{R_2}{\rightarrow} \mathit{read}$

Figure 4: Valid & Invalid Subsequences of E_1

Both subsequences E_{41} and E_{51} are invalid for placing operations O_1, R_2 after operation C_8. Since p_8 was the last executed operation, it is not correct to build subsequences with operations appearing after it that never executed after it.

Match replication subsequences in a process to detect virus replication behavior. The set of replication sequences created in the training session is used to detect virus replication behavior in other processes by replication sequence matching. Processes are monitored at the same degree of granularity and with the same replication set P used during the training session. When a process starts execution an FSA E is initialized. As the process executes operations $p \in P$, E transitions to a new replication state. After each transition the current replication sequence is extracted as a string and compared for a match with the replication sequences set. If a match occurs the process is stopped or suspended and flagged as suspicious for exhibiting virus replication behavior.

5.2 Replication State Frequency Model

This model uses the percentage of replication states occurring in known viruses to detect replication behavior. The basis of the model is a high frequency of execution of replication sequences during one execution of a virus. We expect this would lead to a significantly higher percentage of replication states in a viral process than in benign processes. This model is implemented in three steps:

1. Build a training set of random virus samples
2. Calculate percentage of occurrence for each replication state
3. Match occurrence percentage in a process to detect virus replication behavior

Steps 1-2 is the training session, step 3 is the detection session. Note that the training session uses only complete replication sequences E_i and not subsequences E_{ji}. Step 1 is the same as in Section 5.1 and will be omitted here.

Calculate percentage of occurrence for each replication state. The alphabet $\sum$ and the level of granularity is used the same here as in step 2 of Section 5.1. Each training set virus is executed once and as the transitions occur each distinct replication state is counted. This process is done for each virus in the training set. One counter TSC is used to count the total number of replication states occurring in the viruses of the training set. Once all viruses have been executed, the count for each replication state is divided by TSC, the result is the percentage of occurrence for the specific state. FSA E will run with $\sum$ = replication set P and Q = replication states $\{o, r, w, s, c\}$. As each training set virus executes, E will transition on each executed operation $p \in P$ to a replication state $q \in Q$. The counter for q and TSC are incremented by one. The counter for each replication state is: To, Tr, Tw, Ts, Tc where the subscript represents a replication state. Each counter is initialized to zero. At the end of executing all training set viruses, the five replication state counters are each divided by TSC and the results recorded. These results are the occurrence percentage of each replication state compared to all replication states for the training set.

Match occurrence percentage in a process to detect virus replication behavior. The occurrence percentages calculated in the training session are assigned to the following variables: Po, Pr, Pw, Ps, Pc where each subscript represents a replication state. Processes are monitored with the same replication set P and at the same degree of granularity used in the training session. When a process starts execution an FSA E is initialized. As the process executes operations $p \in P$, E transitions to a new replication state. After each transition, the occurrence percentage for the current replication state q is calculated for E and compared to Pq. If the occurrence percentage of the current replication state q surpasses or equals Pq the process should be suspended or terminated and flagged as suspicious for exhibiting virus replication behavior. This comparison can be extended to two or more replication states. In this extended case, the process is flagged suspicious when two or more occurrence percentages for E equal or surpass their respective Pq. Note that a specific amount of replication states used for detection should be predetermined through testing and analysis to avoid high amounts of false positive and false negative production.

As an example of using this model, consider the two sets of replication sequences in Figure 5. These sequences are built using the mapping of Table 1 and the encoding of Table 2. The first set contains complete replication sequences from a training session. The second set contains replication sequences being monitored during the detection session for viral replication behavior.

Training Session Complete Replication Sequences
$E_1 = \overset{O_1}{\rightarrow} \mathit{opened} \overset{R_2}{\rightarrow} \mathit{read} \overset{R_3}{\rightarrow} \mathit{read} \overset{W_4}{\rightarrow} \mathit{written} \overset{W_5}{\rightarrow} \mathit{written} \overset{W_6}{\rightarrow} \mathit{written} \overset{F_7}{\rightarrow} \mathit{searched}$
$E_2 = \overset{S_1}{\rightarrow} \mathit{read} \overset{W_2}{\rightarrow} \mathit{written} \overset{W_3}{\rightarrow} \mathit{written} \overset{R_4}{\rightarrow} \mathit{read} \overset{L_5}{\rightarrow} \mathit{closed} \overset{F_2}{\rightarrow} \mathit{searched}$
$E_3 = \overset{F_1}{\rightarrow} \mathit{searched} \overset{L_2}{\rightarrow} \mathit{closed} \overset{O_3}{\rightarrow} \mathit{opened} \overset{R_4}{\rightarrow} \mathit{read} \overset{C_5}{\rightarrow} \mathit{written} \overset{S_6}{\rightarrow} \mathit{read} \overset{C_7}{\rightarrow} \mathit{written}$
$E_4 = \overset{W_1}{\rightarrow} \mathit{written} \overset{R_2}{\rightarrow} \mathit{read} \overset{F_3}{\rightarrow} \mathit{searched} \overset{L_4}{\rightarrow} \mathit{closed} \overset{O_5}{\rightarrow} \mathit{opened} \overset{R_6}{\rightarrow} \mathit{read} \overset{T_7}{\rightarrow} \mathit{written} \overset{W_8}{\rightarrow} \mathit{written} \overset{L_9}{\rightarrow} \mathit{closed}$

Detection Session Complete Replication Sequences
$E_1 = \overset{O_1}{\rightarrow} \mathit{opened} \overset{T_2}{\rightarrow} \mathit{written} \overset{W_3}{\rightarrow} \mathit{written} \overset{L_4}{\rightarrow} \mathit{closed}$
$E_2 = \overset{F_1}{\rightarrow} \mathit{searched} \overset{O_2}{\rightarrow} \mathit{opened} \overset{G_3}{\rightarrow} \mathit{searched} \overset{C_4}{\rightarrow} \mathit{written} \overset{S_5}{\rightarrow} \mathit{read} \overset{W_6}{\rightarrow} \mathit{written} \overset{W_7}{\rightarrow} \mathit{written} \overset{L_8}{\rightarrow} \mathit{closed}$
$E_3 = \overset{O_1}{\rightarrow} \mathit{opened} \overset{R_2}{\rightarrow} \mathit{read} \overset{W_3}{\rightarrow} \mathit{written} \overset{L_4}{\rightarrow} \mathit{closed}$

Figure 5: Replication Sequences of Testing and Detection Sessions

The occurrence percentage for Figure 5 are listed in Table 3. The table has two sections: one for the training session and one for the detection session. For both sections, the number of occurrences of each state is listed along with its occurence percentage. Note in the training session the percentages are calculated based on all the complete replication sequences $E_1..E_4$. In the detection session the occurence percentage is calculated for each individual replication sequence E_1, E_2, E_3. Also note the replication sequences in the detection session are not necessarily complete replication sequences. Since comparisons are made after each transition, these sequences could represent any point during the replication process. Assume we are using two replication states: *opened* and *written* to detect virus replication behavior in the detection session. From Table 3 we see the occurence percentage in the training session for *opened* was 10% and for *written* 34%. In the detction session E_1 and E_2 would be flagged as suspicious but not E_3.

Table 3: Occurence Percentage Results

	Number of Occurrences	**Occurence Percentage**
Training Session		
$E_1 ... E_4$, $TSC = 29$		
	$opened = 3$	10%
	$read = 8$	27%
	$written = 10$	34%
	$searched = 4$	14%
	$closed = 4$	14%
Detection Session		
E_1, $TSC = 4$		
	$opened = 1$	25%
	$read = 0$	0%
	$written = 2$	50%
	$searched = 0$	0%
	$closed = 1$	25%
E_2, $TSC = 8$		
	$opened = 1$	12%
	$read = 1$	12%
	$written = 3$	38%
	$searched = 2$	25%
	$closed = 1$	12%
E_3, $TSC = 4$		
	$opened = 1$	25%
	$read = 1$	25%
	$written = 1$	25%
	$searched = 0$	0%
	$closed = 1$	25%

6 Testing and Preliminary Results

We are currently performing tests and analysis on the two detection models in Section 5. We will present here a brief overview of the testing process and report some preliminary results. A sample set of 112 virus binaries was created from malware repositories on the Internet [20, 13]. The set consisted of 4 groups of 28 viruses each. The four groups were of the following four types of viruses: email worms, peer to peer worms, network worms and Win32 viruses. This sample set was used to create 4 test sets of the following sizes: 28, 56, 84 and 112. These 4 test sets also had equal number of viruses of each group stated above. The members of each test set were randomly chosen from the 4 groups. The virtual machine software VMware Workstation with Windows XP-SP2 installed was used to executed the viruses. The replication sequences were recorded using the Process Monitor utility from SysInternals. Testing the replication state frequency model has not yet been completed. Some preliminary results from the operation sequence detection model testing are as follows: for the test set of 28 viruses, the total number of subsequences recorded from the training session was 154,659 of these 77,677 were unique subsequences appearing in 1 or more of the other viruses in the set. The smallest subsequence size with a match was 29 and the largest subsequence size with a match was 231. Several of the subsequences were matched in multiple viruses with some being found in upto 13 viruses. Further analysis revealed a set of 8 subsequences that toghether identified all 28 viruses. These preliminary results are very promising in showing viral replication as a plausible approach to detecting known and unknown viruses.

7 Conclusion and Future Work

We have presented a formal characterization of virus replication. The characterization is based on Cohen's formal model of computer viruses. Two detection models were developed using the characterization. The first model searches for matches of sequence of executed operations to detect a process as viral. The second uses a frequency percentage of replication state occurence to detect if a process is possibly a virus. The preliminary results of our testing are very promising showing with a set of 28 viruses over 77,000 unique subsequences were built that matched in other viruses. Many of these subsequence were foudn in multiple viruses upto 13 for some. A set of 8 subsequences was enough to identify all 28 viruses. This clearly shows that matching replication subsequences can be plausibly used to detect known and unknown viruses. Our future work includes completing the testing and formal analysis of both models. Creating new detection models using data mining and machine learning approaches. Extensive false negative and false positive testing are needed to show the practicality of the approach. A tool that implements these models as dynamic analysis will be built for real time detection testing.

8 Acknowledgments

This was supported in part by the National Science Foundation under Grant No. HRD-0317692. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied by the above agencies.

References

[1] Gostev A. Kaspersky security bulletin 2006: Malware evolution. *Viruslist.com*, February 2007. http://www.viruslist.com/en/analysis?pubid=204791924.

[2] Livingston B. How long must you wait for an anti-virus fix? *Datamation*, February 2004. http://itmanagement.earthweb.com/.

[3] Warrender C., Forrest S., and Pearlmutter B.A. Detecting intrusions using system calls: Alternative data models. In *IEEE Symposium on Security and Privacy*, pages 133–145, 1999.

[4] Eskin E. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the 17th International Conference on Machine Learning*, pages 255–262. Morgan Kaufmann, San Francisco, CA, 2000.

[5] Filiol E. *Computer Viruses: from Theory to Applications.* IRIS International series, Springer Verlag, 2005. ISBN 2-287-23939-1.

[6] Kaspersky E. Problems for av vendors: some thoughts. *Virus Bulletin*, April 2006. http://www.virusbtn.com/virusbulletin/archive/2006/04/vb200604-comment.

[7] Daniel R. Ellis, John G. Aiken, Kira S. Attwood, and Scott D. Tenaglia. A behavioral approach to worm detection. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 43–53, New York, NY, USA, 2004. ACM Press.

[8] Cohen F. *A Short Course on Computer Viruses.* Wiley Professional Computing, 1994. ISBN 0-471-00769-2.

[9] Evers J. Computer crimes cost 67 billion, fbi says. *cnet News.com*, January 2006.

[10] Von Neumann J. Theory of self-reproducing automata. Technical report, University of Illinois, 1966.

[11] Adleman L.M. An abstract theory of computer viruses. In *CRYPTO '88: Advances in Cryptology*, pages 354–374. Springer, 1988.

[12] Darren Mutz, Fredrik Valeur, Giovanni Vigna, and Christopher Kruegel. Anomalous system call detection. *ACM Trans. Inf. Syst. Secur.*, 9(1):61–93, 2006.

[13] Offensive computing. http://www.offensivecomputing.net/.

[14] Szor P. *The Art of Computer Virus Research and Defense*. Symantec Press and Addison-Wesley, 2005. ISBN 9-780321-304544.

[15] Sekar R., Bendre M., Dhurjati D., and Bollineni P. A fast automaton-based method for detecting anomalous program behaviors. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 144, Washington, DC, USA, 2001. IEEE Computer Society.

[16] Forrest S., Hofmeyr S.A., Somayaji A., and Longstaff T.A. A sense of self for unix processes. In *Proceedings of 1996 IEEE Symposium on Computer Security and Privacy*, 1996.

[17] Hofmeyr S., Forrest S., and Somayaji A. Intrusion detection using sequences of system calls. *Journal of Computer Security*, 6:151–180, 1998.

[18] Stolfo S., Apap F., Heller K. Eskin E., Hershkop S., Honig A, and Svore K. A comparative evaluation of two algorithms for windows registry anomaly detection. *Journal of Computer Security*, 13(4), 2005. http://www1.cs.columbia.edu/ids/publications/WindowsRegistry2005.pdf.

[19] Bradley T. The new virus fighters. *Datamation*, January 2006. http://www.pcworld.com/article/id,124163-page,4/article.html.

[20] Vx heavens. http://vx.netlux.org/.

[21] Lee W., Stolfo S., and Chan P. Learning patterns from unix process execution traces for intrusion detection. In *Proceedings of the AAAI97 workshop on AI Approaches to Fraud Detection and Risk Management*, pages 50–56. AAAI Press, 1997.