J Comput Virol
DOI 10.1007/s11416-007-0077-6

ORIGINAL PAPER

Characterization of virus replication

Jose Andre Morales · Peter J. Clarke · Yi Deng · B. M. Golam Kibria

Received: 6 June 2007 / Revised: 27 October 2007 / Accepted: 27 November 2007

Abstract New viruses spread faster than ever and current signature based detection do not protect against these unknown viruses. Behavior based detection is the currently preferred defense against unknown viruses. The drawback of behavior based detection is the ability only to detect specific classes of viruses or have successful detection under certain conditions plus false positives. This paper presents a characterization of virus replication which is the only virus characteristic guaranteed to be consistently present in all viruses. Two detection models based on virus replication are developed, one using operation sequence matching and the other using frequency measures. Regression analysis was generated for both models. A safe list is used to minimize false positives. In our testing using operation sequence matching, over 250 viruses were detected with 43 subsequences. There were minimal false negatives. The replication sequence of just one virus detected 130 viruses, 45% of all tested viruses. Our testing using frequency measures detected all test viruses with no false negatives. The paper shows that virus replication can be identified and used to detect known and unknown viruses.

J. A. Morales (✉) · P. J. Clarke · Y. Deng
School of Computing and Information Sciences,
University Park, Miami, FL 33199, USA
e-mail: jmora009@cis.fiu.edu

P. J. Clarke
e-mail: clarkep@cis.fiu.edu

Y. Deng
e-mail: deng@cis.fiu.edu

B. M. G. Kibria
Department of Statistics, Florida International University,
University Park, Miami, FL 33199, USA
e-mail: kibriag@fiu.edu

1 Introduction

In 2006, an FBI survey reported computer viruses as the number one cause of financial loss for American companies [8]. 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 [11]. 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 [14]. Antivirus companies still average 6 h to release a solution to newly discovered viruses [2,15]. Using signature databases results in slow responses to protect systems from these new viruses. A possible solution to this is behavior based detection [23].

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 (FSM). 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. A fitted linear regression model was also generated. The second model uses frequency metrics of operations used 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. Fitted regression models are also given for both detection models to predict if the processes of the test sets are a virus or benign.

The contributions of this research are:

1. **Showing that virus replication can be detected.** The ability to detect virus replication leads to stopping virus distribution under multiple virus classes and in several different conditions.
2. **Using replication to detect unknown viruses.** An unknown virus can be detected when it attempts to replicate. Detecting replication is not specific to any virus. This allows unknown viruses to be detected without the need for any prior information about the specific virus.

The remainder of this paper is organized as follows: Sect. 2 present review of the current literature. In Sect. 3 a formal characterization of virus replication is built using an FSM and based on Cohen's formal definition [4] of a virus. Section 4 presents two detection models for virus replication using the characterization. Section 5 details our experiments and statistical analysis conducted on the detection models with a discussion of the results. Conclusions and future work are presented in Sect. 6.

2 Literature review

An early formal model of abstract replication is presented in [24]. Formal models of viruses have been presented in [4,1]. Each of these seminal works describe virus replication in some form. Von Neumann created a self reproducing automata showing that replication can be defined formally with computational models. Cohen provides the foundational results using Turing 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. Many detection models, both signature and behavior based have been created using these seminal papers, an excellent summary of these is found in [9,23]. Using sequences of execution of operations to detect malware has been proposed in [7,10,12,16,18,20,22,26]. In each research, the detection is based on anomalous detection or misuse detection [18]. 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.

In [10], 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. Our research takes a similar approach of monitoring sequences of executed operations to detect viruses. The difference between the work presented in [10] and our 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 [10] where training must be repeated for each new program.

In [5], 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. Our research is similar since it uses the definition of a virus for detection. We use the fundamental characteristic of replication for detecting a virus. Our 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.

3 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 [4]. A less strict definition of a virus is a program that recursively and explicitly copies a possibly evolved version of itself [23]. 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 (TM) illustrates virus replication as symbols on a tape being read and written to another area further down the tape [4, p.164]. Cohen's definition is explained with the following steps:

1. For all M and V, the pair (M,V) $\in V$ iff
2. V is a set of TM sequences and M is a TM where
3. M's tape head is at a cell j at time t and the tape cells starting at j hold the virus v
4. At a time t' $>$ t tape cells starting at cell j', far enough away from v hold the virus v' such that
5. At time t $<$ t" $<$ t', v' is written by M to tape cells starting at j'

Steps 1 and 2 setup the Turing machine M for processing the tape with the virus. Steps 3 through 5 illustrate the read and write operations of the virus. Replication starts with the interpretation of the virus beginning at cell j to cause M to read and write the virus to a sequence of cells further down the tape beginning at cell j'. The symbols defining the virus starting at cell j are read and written one at a time to a new location further down the tape starting at cell j'. Between each read and write, M's tape head has to search up and down the tape for the next symbol of the virus to read and the next cell to write. 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 the program, which may require an open operation. To read and write the symbols of the virus, the tape head must first move to it and be allowed to access the individual symbol, this implies an open operation. When the tape head moves away from a cell, the symbol in that cell is left in an idle state ready for use at some future point. The act of the tape head moving away from a symbol and leaving it ready for future use is implicitly a close operation. Once infection is completed, a newly infected program may have to be closed to become usable by a system. Cohen's definition also shows that a virus will have M read and write the virus itself to some other location of the tape by instructing M to read the virus starting at j and write one symbol at a time starting at j'. The entire replication process in Cohen's definition illustrates the virus instructing M to read and write the virus itself. The virus refers to itself as the source of the whole replication process. Referring to itself in this manner is essential for the virus to successfully replicate. Cohen states that any element of a viral set can produce any number of other elements of the set depending on the remaining tape [4, p. 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.

When replication of a virus occurs, some of the operations contain references to the virus itself. For the virus to successfully replicate, it will refer to itself in some of the open, close, read, write, search operations. For example a replication sequence of a virus referring to itself could be: search(a) open(a) read(a) write(a,b) close(a) close(b), with virus a and infection target b. When a virus has completed replication a subset of the operations performed will have referred to the virus itself. It is these operations with a self reference to the virus that distinguish a virus from a benign process. 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*. We assume some of the operations in this sequence will self reference the virus itself indicating the occurrence of virus replication. 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 is executed. The replication states are defined as follows: $o = opened$, $r = read$, $w = written$, $s = searched$, $c = closed$. 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 $e_1, e_2, e_3, \ldots, e_n$ with n being the total number of executed operations. The function $length(E_i)$ returns the total number of events e for the sequence E_i, therefore $length(E_i) = n$. The subscript i initialized to 1 uniquely identifies each replication sequence, the ith 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 machine (FSM)* in Fig. 1.

The members of $\sum$ must solely consist of the specific operations p which when executed individually, transition E to a replication state q. There is no specific start and final replication states for s and f. When a virus attempts replication it may start in any replication state q. A virus may also exit replication from any state q. No specific 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. Thus all the replication states of Q are valid for s and f 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

Fig. 1 Virus replication sequence FSM

- $\sum$ 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
- $\delta : Q \text{ x } \sum \rightarrow Q$

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

Fig. 2 Abstract complete replication sequence E_i

transition the virus to a replication state q. FSM E is meant to capture only the ordered sequence of executions of operations 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. FSM 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 being examined for possible exhibition of virus replication behavior. If FSM E produces a replication sequence belonging to a benign process it is not labeled and discarded as invalid. FSM E only accepts replication sequences belonging to a virus. A replication sequence E_i can hold either 1. a complete replication sequence, which is the sequence of all the operations p executed by a virus in one complete execution, or 2. a subsequence of a complete replication sequence which is defined in Fig. 4. A sample production of FSM E abstractly showing a complete replication sequence of specific operations p and the corresponding replication state q is in Figure 2.

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

4 Replication detection models

Two models for detecting virus replication are presented in this section. Both are based on the characterization and FSM presented in Sect. 3. The characterization defines virus replication as a replication sequence consisting of operations p and cause a transition to a replication state 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. In the detection models presented here FSM E records the replication sequence of a process. When the detection starts it is not known if a process is a virus or benign. If the sequence is detected as a virus or exhibiting virus replication behavior it is accepted by E as a valid replication sequence and labeled E_i. If the sequence is not detected as a virus it is assumed benign, not labeled and not used any further in the detection.

Table 1 Operation to state mapping

Replication state	Operation name
opened	`openfilex`
read	`readfilex`
	`setfilepointer`
written	`copytofile`
	`createfilenew`
	`writefilex`
searched	`finddir`
	`getfileattrib`
closed	`closefile`

4.1 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. This is illustrated with the mapping of operations to states in Table 1.

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.

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, FSM E would produce the complete replication sequence in Fig. 3.

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

Fig. 3 Complete replication sequence E_1

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

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`

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 you are detecting a specific class of virus, for example Win32 viruses, then your training set should consist of random samples of only Win32 viruses.

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 very abstract granularity level may reveal less information about the execution. It may only provide high level calls made by the virus. These calls may group more specific calls together making it difficult to classify them as part of a replication process. This could result in false negative production due to poorly recorded replication sequences. A very detailed granularity level may provide too much information such as system call arguments and hardware level information, which is not used in our recording. This extra information could slow down the recording due to the overhead of parsing out the unneeded information. 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, an expert may be required to ensure a correct analysis. The specific operations p are those used at the chosen granularity that transition the virus into a replication state q. The replication set P must contain only 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. FSM 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, which is shown in Fig. 3. The sequence of operations $p_1 \ldots p_n$ are extracted from E_i, encoded, converted to a string and recorded.

$$E_{ji} = \overset{p_k}{\rightarrow} q_k \ldots \overset{p_{k+l}}{\rightarrow} q_{k+l}$$

$$\text{with } k \geq 1,\ l \geq 0 \text{ and } k + l \leq length(E_i)$$

Fig. 4 Abstract replication subsequence E_{ji}

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 $E_1 \ldots E_i$ 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 jth subsequence of i and i is the ith complete replication sequence. An abstract subsequence is defined in Fig. 4. 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, Fig. 5 has two valid and invalid subsequences of the replication sequence E_1 shown in Fig. 3.

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

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

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

Fig. 5 Valid and invalid subsequences of E_1

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 FSM E is initialized. As the process executes operations 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.

4.2 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 Sect. 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 Sect. 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. FSM 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 to a replication state 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 FSM E is initialized. As the process executes operations 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.

As an example of using this model, consider the two sets of replication sequences in Fig. 6. 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.

The occurrence percentage for Fig. 6 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 occurrence percentage. Note in the training session the percentages are calculated based on all the complete replication sequences $E_1 \ldots E_4$. In the detection session the occurrence 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

Fig. 6 Replication sequences of testing and detection sessions

Training Session Complete Replication Sequences

$E_1 = \stackrel{O_1}{\rightarrow} opened \stackrel{R_2}{\rightarrow} read \stackrel{R_3}{\rightarrow} read \stackrel{W_4}{\rightarrow} written \stackrel{W_5}{\rightarrow} written \stackrel{W_6}{\rightarrow} written \stackrel{F_7}{\rightarrow} searched$

$E_2 = \stackrel{S_1}{\rightarrow} read \stackrel{W_2}{\rightarrow} written \stackrel{W_3}{\rightarrow} written \stackrel{R_4}{\rightarrow} read \stackrel{L_5}{\rightarrow} closed \stackrel{F_2}{\rightarrow} searched$

$E_3 = \stackrel{F_1}{\rightarrow} searched \stackrel{L_2}{\rightarrow} closed \stackrel{O_3}{\rightarrow} opened \stackrel{R_4}{\rightarrow} read \stackrel{C_5}{\rightarrow} written \stackrel{S_6}{\rightarrow} read \stackrel{C_7}{\rightarrow} written$

$E_4 = \stackrel{W_1}{\rightarrow} written \stackrel{R_2}{\rightarrow} read \stackrel{F_3}{\rightarrow} searched \stackrel{L_4}{\rightarrow} closed \stackrel{O_5}{\rightarrow} opened \stackrel{R_6}{\rightarrow} read \stackrel{T_7}{\rightarrow} written \stackrel{W_8}{\rightarrow} written \stackrel{L_9}{\rightarrow} closed$

Detection Session Complete Replication Sequences

$E_1 = \stackrel{O_1}{\rightarrow} opened \stackrel{T_2}{\rightarrow} written \stackrel{W_3}{\rightarrow} written \stackrel{L_4}{\rightarrow} closed$

$E_2 = \stackrel{F_1}{\rightarrow} searched \stackrel{O_2}{\rightarrow} opened \stackrel{G_3}{\rightarrow} searched \stackrel{C_4}{\rightarrow} written \stackrel{S_5}{\rightarrow} read \stackrel{W_6}{\rightarrow} written \stackrel{W_7}{\rightarrow} written \stackrel{L_8}{\rightarrow} closed$

$E_3 = \stackrel{O_1}{\rightarrow} opened \stackrel{R_2}{\rightarrow} read \stackrel{W_3}{\rightarrow} written \stackrel{L_4}{\rightarrow} closed$

Table 3 Occurrence percentage results

	Number of occurrences	Occurrence percentage
Training session $E_1 \ldots E_4, TSC = 29$		
	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

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 occurrence percentage in the training session for *opened* was 10% and for *written* 34%. In the detection session E_1 and E_2 would be flagged as suspicious but not E_3.

5 Experiments

In this section we present two sets of experiments each for the models described in Sect. 4. We first describe the setup and environment in which the experiments were performed identifying the type of hardware and software used to support the experiments. The virus set used in the training, and evaluation are also described. Each detection model is tested first by following the model as described in Sect. 4 and is then tested using regression analysis.

5.1 Experiment setup

A sample set of 84 virus binaries was created from malware repositories on the Internet [19,25]. The names of the 84 viruses belonging to the sample set are listed in Appendix Table 12. The set consisted of 4 groups of 21 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 3 test sets of the following sizes: 28, 56 and 84. These 3 test sets also had equal number of viruses of each group stated above. The members of each test set were randomly chosen from the four 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 Microsoft Windows Sysinternals. The level of granularity was at the I/O request packet (IRP) function code level [13]. Table 4 lists the IRP operations used in the testing. Next to each operation name is the encoded character used and the replication state resulting from the operationć6s execution. The encoded characters were assigned first in numerical order starting with 0 then in alphabetical order. The operation names are the actual IRP function code names. The grouping of operation name to replication state was done based on each functions description in [13].

5.2 Operation sequence testing

For the training session, three test sets were executed in the virtual machine. The complete replication sequence of each virus was recorded, encoded and saved to a database and subsequences were created. The minimum size was 30 and the maximum size 1000 respectively. If a subsequence matched in any of the other recorded operation sequences of the test

Table 4 IRP operations

Operation name	Encoded character	Replication state
IRP_MJ_CREATE	0	
IRP_MJ_FILE_SYSTEM_CONTROL	a	*opened*
FASTIO_NETWORK_QUERY_OPEN	g	
IRP_MJ_READ	1	
FASTIO_CHECK_IF_POSSIBLE	e	*read*
FASTIO_READ	i	
IRP_MJ_WRITE	2	
IRP_MJ_FLUSH_BUFFERS	5	
IRP_MJ_SET_INFORMATION	7	
FASTIO_ACQUIRE _FOR_CC_FLUSH	b	*written*
FASTIO_WRITE	k	
FASTIO_ACQUIRE _FOR_MOD_WRITE	m	
IRP_MJ_DIRECTORY_CONTROL	3	
IRP_MJ_QUERY _VOLUME_INFORMATION	4	
IRP_MJ_QUERY_INFORMATION	6	*searched*
FASTIO_QUERY_INFORMATION	h	
IRP_MJ_CLEANUP	8	
IRP_MJ_CLOSE	9	*closed*

Table 5 Operation sequence testing results

Training session			
	28 vx	56 vx	84 vx
Total subsequences	154,659	1,089,647	2,218,129
Unique subsequences	104,977	573,789	876,554
Smallest subsequence match size	30	30	30
Longest subsequence match size	231	596	596
Smallest match set	8	11	19
Detection session			
	28 vx	56 vx	84 vx
Total subsequences	903,832	2,721,137	4,816,314
Unique subsequences	75,389	258,375	486,064
Smallest subsequence match size	30	30	30
Longest subsequence match size	194	255	255
False negatives	17	12	10
Smallest match set	19	23	24

set, the name of the matching virus was saved along with the subsequence and the name of the virus it was created from. Testing of the training session produced a database of subsequences of known viruses that matched in one or more recorded operation sequences of the other known viruses in the test set.

For the detection session, a set of 200 viruses was created, executed and the complete replication sequence recorded, encoded and saved to a database. The names of the 200 viruses are listed in Appendix Table 13.The viruses in this set were not related to any family of viruses used for the training session. The purpose of this was to search for the subsequences produced from the training session in unknown viruses. This was a simulation detection of virus behavior in currently executing processes. The 200 viruses were not used in training and therefore can be considered to be unknown viruses that detected based only on their exhibition of virus replication behavior. Testing of the detection session produced a database of subsequences that occurred in one or more viruses of the training and detection sessions.

Test results from the training and detection sessions are in Table 5. The first column is the reported test results. The next three columns are the three tests sets. Total subsequences is the total number of matches for each operation subsequence of the test set. The training session produced a high amount of subsequences with at least one match. The total for unique subsequences represents the number of unique operation subsequences with at least one match. The results show the number of unique subsequences decreased as the test set size increased. The set of 28 viruses had 70% unique subsequences, the set of 56 had 50% unique and the set of 84 had only 40% unique. The decrease in unique subsequences resulted in smaller number of subsequences appearing in a larger number of viruses.

Many subsequences appeared in multiple viruses with at least one appearing in 87 viruses. There was no match with a subsequence longer than 596. Our testing created subsequences up to length 1000, the results showed only subsequences of half that size were needed. The smallest match set is the minimum number of sequences needed to detect all the viruses in a test set. The set of 28 was detectable with 8 subsequences; the set of 56 with 11 and the set of 84 with only 19 subsequences. In the testing of the detection session with the set of 200 viruses, 8% of the subsequences were unique for the set of 28 viruses, and 10% for both sets of 56 and 84 viruses. The set of 28 produced 17 false negatives, the set of 56 decreased that amount to 12 and the set of 84 further decreased to 10. With testing completed, 10, or 2.5% of the set of 200, viruses were still undetected.

Further analysis of the operation sequence testing revealed viruses whose complete replication sequences had a large number of matched subsequences. Table 6 lists some of these viruses. The first column contains the name of the virus whose complete replication sequence contains multiple matched subsequences. The column for number of sequences

Table 6 Viruses containing multiple sequences

Virus name	Number of subsequences	Detected viruses
Bagle.a	11	130
Eyeveg.m	7	96
Plexus.a	6	62

is the amount of matched subsequences contained in the complete replication sequence of the virus. The last column is the matching viruses, the number of viruses detected during our testing with one of the subsequences found in the virus listed in the first column.

Consider the virus Bagle.a. The number of subsequences for this virus is 11. This means the complete replication sequence of Bagle.a contains 11 subsequences that successfully detected one or more viruses during testing. The number of matching viruses is 130. Out of 284 tested viruses, 130 were detected with virus behavior because they contained at least one of the 11 subsequences belonging to the complete replication sequence of the Bagle.a virus. The subsequences produced from the complete replication sequence of this one virus detected 130 viruses, 45% of the total number of viruses used in our testing. This observation suggests that many viruses can be detected based on the virus replication of a small number of known viruses. This can be used to possibly show that virus replication is relatively consistent in all viruses due the limited number of ways that replication can occur in a virus.

Statistical analysis was performed on the results of the operation sequence testing. The results are in Table 7.

From Table 7, most of the detected viruses are Email worms followed by Network worms, peer to peer worms and Win32 Viruses. The 95% confidence interval for the mean number of detection by all groups is between 40.34138 and 51.82528.

Analysis of variance (ANOVA). The statistical model is

$$y_{ij} = \mu + \tau_i + e_{ij}$$

We want to test the hypothesis that the average number of detection by all groups (four groups) are the same. That is

$$H_0 : \tau_1 = \tau_2 = \tau_3 = \tau_4 = 0$$

vs

$$H_a : \tau_i \neq 0 \quad \text{for some i}$$

Using Splus [21], we have the following analysis of variance table [17]:

Source	*Df*	Sum of Sq.	Mean Sq.	*F* Value	*P* value
Treatment	3	4272.89	1424.298	2.116596	0.1046218
Residuals	80	53833.52	672.919		

From the above table, the P-value indicates that the means are not statistically different at 5% level of significance. However, they are statistically different at 11% level of significance.

Regression analysis. We assume that y_i is the observed value of the random variable Y_i, which depends on x_i according to the following model [6],

$$Y_i = \beta_0 + \beta_1 x_i + e_i, \quad i = 1, \ldots, n, \tag{1}$$

where y_i = Detection of the viruses which are responses of the model and x_i = replication sequence, β_0 is the intercept, the value of Y when x equals zero. The slope β_1, the regression coefficient, is the rate of change in detection for a unit change in sequence, e_i's are random errors and are independently and identically distributed normal random variable with $E(e_i) = 0$, $V(e_i) = \sigma^2$. Using Splus, we have the following regression analysis

```
*** Linear Model ***

Call: lm(formula $=$ Detection ~ Sequence, data $=$ Ancova, na.action $=$ na.exclude)
Residuals:
    Min      1Q Median    3Q   Max
 -37.78 -13.97 0.7393 11.04 47.98

Coefficients:
              Value Std. Error t value Pr(>|t|)
(Intercept) 11.4994  3.9041     2.9455  0.0042
   Sequence 11.7613  1.1542    10.1897  0.0000

Residual standard error: 17.68 on 82 degrees of freedom
Multiple R-Squared: 0.5587
F-statistic: 103.8 on 1 and 82 degrees of freedom, the p-value is 3.331e-016
```

Table 7 Operation sequence statistical analysis

Statistics	Email worms	Peer to peer worms	Network worms	Win32 Viruses
Minimum	2.00000	7.000000	4.000000	2.000000
1st Quartile	46.00000	13.000000	29.000000	8.000000
Mean	57.38095	43.095238	45.952381	37.904762
Median	59.00000	52.000000	47.000000	46.000000
3rd Quartile	68.00000	70.000000	59.000000	62.000000
Maximum	130.00000	89.000000	83.000000	76.000000
Std Deviation	27.60340	28.607525	20.920029	25.955548
Total:	1205.00000	905.000000	965.000000	796.000000
95% LCL for mean:	44.81604	30.073249	36.429698	26.089938
05% UCL for mean:	69.94587	56.117227	55.475064	49.719586

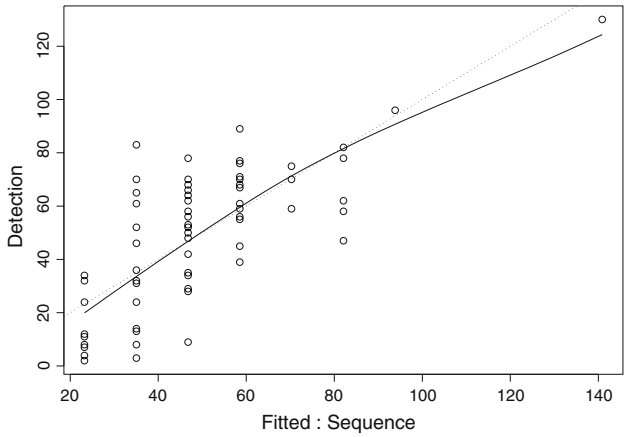

Fig. 7 Detection of viruses vs fitted values

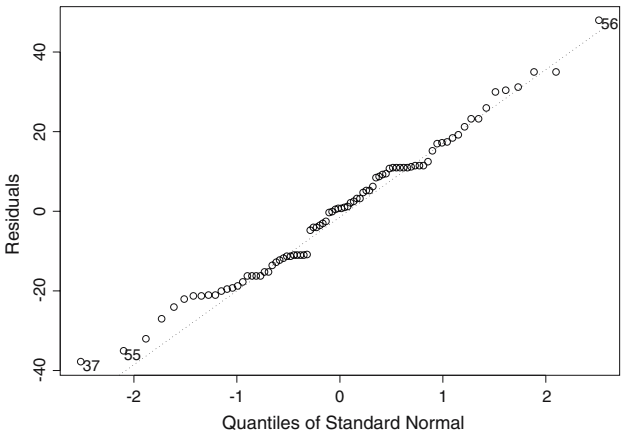

Fig. 8 Normal probability plot of ordinary least squares residuals which is detection of viruses minus fitted values of virus detection

Since the p value is 0.0000, we can reject the null hypothesis, $H_0 : \beta_1 = 0$ at 1% or less significance level. Therefore, we might conclude that the sequence has significant effect on the detection of viruses. On the other hand, detection of viruses can be predicted from the regressor sequence.

The fitted regression model is

$$\hat{y} = 11.4994 + 11.7613 \times \text{Sequence}$$

From the above fitted model we will be able to predict the number of detection of the viruses for known sequences.

The plots of response vs fitted values and normal probability plot of residuals are given in Figs. 7 and 8, respectively. From these figures we observed that the assumptions of the linear regression model are satisfied. The $R^2 = 0.56$, which indicates that about 56% of the total variation has been explained by the regressor sequence.

5.3 Replication state frequency testing

The complete replication sequence of each virus in the three sets was created in the same manner as the operation sequence detection model training session testing. The Po, Pr, Pw, Ps, Pc values for each replication state over the entire complete replication sequences set was calculated and recorded. This was repeated for each test set.

The detection session was performed using all the replication states. The set of 200 viruses was executed and the complete replication sequences recorded, encoded and saved to a database. The occurrence percentage for each replication state was calculated for each individual virus and compared to the Po, Pr, Pw, Ps, Pc values from the training session. If the amounts equaled or surpassed these values from the training session, then the sequence was recorded for exhibiting virus replication behavior.

Test results for both sessions are listed in Table 8. In the training session the results list the values of Po, Pr, Pw, Ps, Pc for each test set. All of these values decreased as the test set size increased not including written and searched. This increase may imply that these two states are the most frequently occurring of all the replication states in a virus. For the detection session, only false negative amounts are listed. A process was flagged as a virus if all process's replication

Table 8 Frequency testing results

Training session			
	28 vx	56 vx	84 vx
Occurrence percentage			
Read	6%	5%	4%
Written	8%	15%	20%
Open	35%	27%	25%
Closed	31%	30%	26%
Searched	18%	23%	24%
Detection Session			
	28 vx	56 vx	84 vx
False negatives	200	0	0

Table 9 Classification table with a cut value of 0.500

			Predicted		
			Response		Percentage Correct
Observed			0.00	1.00	
Step 1	Response	0.00	23	5	82.1
		1.00	3	25	89.3
Overall percentage					85.7

states matched or exceeded the corresponding occurrence percentages from the training session.

The set of 28 had no detections. All 200 viruses were false negatives. For each virus, only the occurrence percentage of some replication states matched or exceeded the occurrence percentages from the training session. In the set of 56 and 84, all 200 viruses were detected. For each virus, all replication state occurrence percentages matched or exceeded the occurrence percentages from the training session. No false negatives occurred.

A fitted logistic regression model [6] was created for the set of 28 viruses along with a set of 28 benign processes. The occurrence percentages were calculated for the 28 benign processes. Fitting of the model was done using all possible permutations of written, read, closed, opened and searched replication states. The model was fitted using the occurrence percentage of written, read, opened and closed which detected the highest number of viruses and benign processes. The details of the model without using a safe list are in Tables 9 and 10. Values for Hosmer and Lemeshow test were: Chi-square = 2.547, DF = 7 and Significance = 0.924. The summary of the model: −2 log likelihood = 7.535(a), Cox and Snell R square = 0.714 and Nagelkerke R Square = 0.952. Note that the estimation was terminated at iteration number 18 because parameter estimates changed by less than 0.001.

The variables, written and read are significant at 10% level of significance. Cox R-square = 0.714, that means roughly 71% of the total variations have been explained by the regressors, write, read, open, close. The p-value of Hosmer and Lemeshow test is 0.924, which indicates a very good fit.

The fitted logistic regression model is

$$ln\frac{\hat{\pi}_i}{1-\hat{\pi}_1} = 2.581 - 0.004x_1 - 0.017x_2 + 0.015x_3 - 0.029x_4 \quad (2)$$

where $x_1 = Pw$, $x_2 = Pr$, $x_3 = Po$ and $x_4 = Pc$. The predicted probability model is

$$\hat{\pi} = \frac{e^{2.581-0.004x_1-0.017x_2+0.015x_3-0.029x_4}}{1+e^{2.581-0.004x_1-0.017x_2+0.015x_3-0.029x_4}} \quad (3)$$

A logistic regression model was also fitted for the set of 56 and 84 viruses along with the set of 28 benign processes using the Po, Pr, Pw, Ps, Pc values of write, read, open and close. All three models were validated using the set of 200 viruses and a set of 60 benign viruses. The detection results for the fitting and validation of the three sets using the safe list are listed in Table 11.

5.4 Discussion

Testing the operation sequence model produced 10 false negatives in the detection session. To eliminate the false negatives two options can be used. First, analyzing each false negative to create an replication subsequence that can be used to detect it. This would add an additional 10 subsequences to the smallest match set. The disadvantage of this option is the increase in size of the match set. The advantage is the newly

Table 10 Variables used in the classification model shown in Table 9

		B	S.E.	Wald	*df*	Sig.	Exp(B)	95.0% C.I. for EXP(B)	
		Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper
Step 1(a)	Written	−0.004	0.002	2.741	1	0.098	0.996	0.991	1.001
	Read	−0.017	0.009	3.720	1	0.054	0.983	0.967	1.000
	Opened	0.015	0.015	0.958	1	0.328	1.015	0.986	1.045
	Closed	−0.029	0.020	2.116	1	0.146	0.972	0.935	1.010
	Constant	2.581	0.826	9.772	1	0.002	13.211		

Table 11 Logistic regression detection with WROC

Model fitting			
	28 vx	56 vx	84 vx
Detected viruses	25/28	56/56	82/84
Detected benign	28/28	28/28	28/28
Model validation			
	28 vx	56 vx	84 vx
Detected viruses	115/191	117/191	145/191
Detected benign	60/60	60/60	60/60

added subsequences may detect an unknown virus at some future point that is not detected by the other subsequences in the set. This advantage prevents false negatives and eliminates the need to add more subsequences to the set. The second option is to combine the operation sequence model with other replication detection models which is a previously researched approach [3, Chap. 9]. The test results for the frequency model did not produce any false negatives. Even though no false negatives occurred in the frequency model, false negatives are possible in cases where a virus has very few replications. In these cases the frequency of use of certain operations may not be high enough for our detection model to identify the virus. Combining these two models together could eliminate the false negatives produced in our testing. A process would be flagged as viral if it satisfies one of the two detection models and is not on the safe list.

The two detection models and two regression models produced no false positives during testing. This resulted from using a safe list to avoid flagging processes that were predetermined to be benign. The safe list works on the assumption that a process name is added to the list after determining the process to be benign. To keep a safe list accurate, the process name of newly installed programs should be added to the list before running the program. If a false positive is produced, the process may be seldom used and overlooked when updating the list. A false negative can occur if a newly installed process name is added to the safe list believed to be benign when in fact it is infected with a virus. The safe list was built by recording the names of processes running on three computers: a residential desktop, an office workstation and an enterprize server. The three computers were monitored for a 24 h period where the name of all the processes that ran during that period were recorded. The process names were later checked manually to assure they were of known user/server/system programs. The resulting list contained 117 process names.

An outside source was asked to record the complete replication sequences of 28 and 60 processes running on an enterprize server and an office workstation. This was done by the person without knowledge of which processes were on our safe list. The 60 processes were added to the set of 200 viruses. We repeated the testing of the detection session. This time a process would be flagged as viral only if it matched the criteria of the detection model being used and the process name did not appear on the safe list. These same steps were applied with the set of 28 benign processes used for the fitted logistic regression for the replication state frequency model. The results were the same with no false positives.

The two detection models and regression models were tested without using the safe list. The detection models produced no false negatives but a high number of false positives. This may be due to the fact that replication is a general process that can occur in both benign processes and viruses. The regression models produced almost no false positives but a high amount of false negatives. This research is focused on minimizing false negative and increasing true positive production. The safe list component easily removes all false positives and produces very effective detection when used with a model producing little or no false negatives. The trade off is a slight overhead increase in using the safe list in return for little or no false positives. The alternative is using a model that algorithmically produces minimal false positives but at the possible cost of high false negatives. A combination of the models presented here could lead to a new hybrid model that algorithmically produces little or no false negatives and false positives. Another option would be to inspect the operation arguments and use that information to aide in detection improvement.

The viruses used for our testing were executed in a virtual machine. The virtual machine ran with Windows XP SP2 installed and no network or Internet access. These conditions may have been sub-optimal for some of the viruses. This may have resulted in some viruses to have executed less replication attempts than the virus would normally execute in more optimal conditions. Creating more optimal conditions for virus execution could possibly have produced better data resulting in a reduction in the amount of false negatives produced by our testing.

6 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 occurrence to detect if a process is possibly a virus. Testing was performed for both detection models and statistical analysis conducted including regression models. The operation sequences detected over 250 viruses with 43 subsequences. Detection of 130 virus which is 45% of all tested viruses were detected with the complete replication

sequence of just one virus. The frequency testing also detected all the viruses used in testing. Our testing showed both replication sequences and frequency can be plausibly used to detect known and unknown viruses. The testing also showed a relative consistency in the replication process of different viruses. This consistency may suggest a limited number of ways that a virus can replicate. Our testing results show both detection models to be effective in detecting both known and unknown viruses producing minimal false negatives. We also controlled false positives through the addition of a safe list. Our future work includes testing the algorithms with training sets based on just one family of viruses. Creating a hybrid detection model from the models presented here to minimize false negatives and false positives. Adding information gathered from system call arguments to our detection models to attempt further reduction of false negatives and false positives.

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.

Appendix

Table 12 Set of 84 viruses

Email Worms	Peer to Peer Worms	Network Worms	Win32 Viruses
Badtrans.a	Agobot.a	Afire.b	Cabanas.e
Bagle.a	Agobot.b	Afire.c	DarkSide.1371
Bagle.j	Agobot.c	Afire.d	Elkern.a
Bagle.k	Agobot.d	Bozori.b	Enumiacs-6656
Bagle.m	Backterra.a	Bozori.e	Levi-2961
Bagle.n	Banuris.a	Bozori.j	Mental
Bagle.o	Bereb.a	Dabber.a	Mental-10000
Dumaru.r	Bereb.b	Dabber.b	Neshta.a
Eyeveg.m	Gedza.b	Dabber.c	Parite.a
Klez.a	Gedza.c	Domwoot.c	Parite-b
Klez.e	Habaku.b	Doomjuice.b	Seppuku.6834
Klez.i	Kifie.a	Kidala.a	Small.a
Merkur.b	Kifie.c	Kidala.b	Small.b
Mimail.j	Kifie.f	Lebreat.a	Tapan-3882
Mydoom.ax	Niklas.b	Muma.b	Thorin.11932
Mydoom.b	Niklas.c	Muma.c	Thorin.b
Plexus.a	Opex.a	Opasoft.a	Thorin.c
Sircam.a	Polip.a	Padobot.m	Xorala
Sircam.d	Zaka.a	Sasser.b	Younga.4434
Sober.a	Zaka.f	Theals.c	ZMist
Sober.f	Zaka.m	Vesser.a	ZPerm.b

Table 13 Set of 200 viruses

Email Worms	Peer to Peer Worms	Network Worms	Win32 Viruses
Abotus	Abuva	3DStars	Aidlot
Actem	Adil	CodeGreen.a	Andras.7300
Agist.a	Alcan.a	Cycle.a	Apathy.5378
Alanis	AntiFizz	Ezio.a	Apoc.a
Aliz	Aplich	Francette.a	Arch.a
Altice	Apsiv	Francette.b	Aris
Amus.a	Aritim	Francette.c	Artelad.2173
Anarch	Ariver	Francette.d	Bacros.a
Android	Blaxe	Francette.e	Banaw.2157
Anel	Cabby	Francette.g	Barum.1536
Animan	Cake	Hiberium.b	Basket.a
Anpir.a	Carfin	Maslan.a	Bayan.a
Antiax	Cassidy	Maslan.b	BCB.a
Antites	Cayen.a	Mytob.q	Bee
Aplore	Cocker	Protoride.aa	Beef.2110
Apost	Compatex	Protoride.ai	Bender.1363
Assarm	Compux.a	Protoride.al	Bika.1906
Atirus	Cozit	Protoride.ar	Blateroz
Avoner	Dafly.b	Protoride.b	Bluback.1376
Babuin	Dani	Protoride.bk	Blueballs.4117
BabyBear	Delf.a	Protoride.e	Bogus.4096
Badass	Druagz	Protoride.f	Bondage.968.a
Bandet.a	Erdam	Protoride.g	Butter
Banza	Flocker	Raleka.b	CabInfector
Bater.a	Franvir	Rega.a	Cecile
Benny	Furby	Salie.a	Civut.a
Bimoco.a	Gagse	SdBoter.a	Cloz.a
Black	Gotorm	SdBoter.b	Cmay.1222
Blare	Grompo	SdBoter.c	Cornad
Blitzy	Halfint	SdBoter.g	Crosser
Bonorm	Huntox	SdBoter.k	Delfer.a
Bormex	Ident	Shelp.a	Devir
Borzella	Insta.a	Spoder.a	Dictator.2304
Botter.a	Inter	Stap.b	Dislex
Bumper	Irkaz	Stap.e	Gipiras.a
Burnox	Kabak.a	Stap.f	Hezhi
Calil	Kamadina	Syner.a	Jlok
Calposa	Kamafe	Webdav.a	Kenfa.a
Carfrin	Kanyak.a	Welchia.a	Netlip
Cervivec	Kapucen.b	Welchia.b	Niya.a
CWS.a	Kazeus	Welchia.c	Porad.a
Happy	Kenfo	Welchia.e	Sinco
Yoxec	Kevor	Xatch.a	Spreder
Zar.a	Kovirz	Zan	Sugin
Zhangpo	Krepper	Zusha.a	TeddyBear
Zircon	Lamerx	Zusha.b	VChain
Zoek	Lemb.b	Zusha.c	Watcher.a
Zoher	Vagas.a	Zusha.e	Zevity
Zush	Walrain	Zusha.f	Zorg.a
Zwur.a	Weakas	Zusha.h	Zori.a

References

1. Adleman, L.M.: An abstract theory of computer viruses. In: CRYPTO '88: Advances in Cryptology, pp. 354–374. Springer, Heidelberg (1988)
2. Bradley, T.: The new virus fighters. Datamation, January 2006. http://www.pcworld.com/article/id,124163-page,4/article.html
3. Christodorescu, M., Jha, S., Maughan, D., Song, D., Wang, C., (eds.) Malware Detection. Springer, Heidelberg (2007)
4. Cohen, F.: A Short Course on Computer Viruses. Wiley Professional Computing, 1994. ISBN 0-471-00769-2
5. Daniel R. Ellis, John G. Aiken, Kira S. Attwood, Scott D. Tenaglia.: A behavioral approach to worm detection. In: WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode, pp. 43–53. ACM Press, New York, (2004)
6. Draper, N.R., Smith, H.: Applied Regression Analysis. 3rd ed. Wiley, New York (1998)
7. Eskin, E.: Anomaly detection over noisy data using learned probability distributions. In: Proceedings of 17th International Conference on Machine Learning, pp. 255–262. Morgan Kaufmann, San Francisco (2000)
8. Evers, J.: Computer crimes cost 67 billion, fbi says. CNET News.com, January 2006
9. Filiol, E.: Computer Viruses: from Theory to Applications. IRIS International series, Springer, Heidelberg (2005). ISBN 2-287-23939-1
10. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for unix processes. In: Proceedings of 1996 IEEE Symposium on Computer Security and Privacy (1996)
11. Gostev, A.: Kaspersky security bulletin 2006: Malware evolution. Viruslist.com, February (2007)
12. Hofmeyr, S., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. J. Comput. Security **6**, 151–180 (1998)
13. Irp function codes. http://msdn2.microsoft.com/en-us/library/ms796136.aspx
14. Kaspersky, E.: Problems for av vendors: some thoughts. Virus Bull. April 2006. http://www.virusbtn.com/virusbulletin/archive/2006/04/vb200604-comment
15. Livingston, B.: How long must you wait for an anti-virus fix? Datamation, February 2004. http://itmanagement.earthweb.com/
16. Lee, W., Stolfo, S., 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, pp. 50–56. AAAI Press, New York (1997)
17. Montgomery, D.C. : Design and Analysis of Experiments. Wiley, New York (2001)
18. Mutz, D., Valeur, F., Vigna, G., Kruegel, C.: Anomalous system call detection. ACM Trans. Inf. Syst. Secur. **9**(1), 61–93 (2006)
19. Offensive computing malware repository. http://www.offensivecomputing.net
20. Sekar, R., Bendre, M., Dhurjati, D., 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, pp. 144. IEEE Computer Society, Washington, DC (2001)
21. S-plus statistics sofware package. http://en.wikipedia.org/wiki/S-PLUS
22. Stolfo, S., Apap, F., Heller, K., Eskin, E., Hershkop, S., Honig, A, Svore, K.: A comparative evaluation of two algorithms for windows registry anomaly detection. J. Comput. Security, **13**(4), (2005)
23. Szor, P.: The Art of Computer Virus Research and Defense. Symantec Press and Addison-Wesley, 2005. ISBN 9-780321-304544
24. Von Neumann, J.: Theory of self-reproducing automata. Technical report, University of Illinois (1966)
25. Vx heavens. http://vx.netlux.org/
26. Warrender, C., Forrest, S., Pearlmutter, B.A.: Detecting intrusions using system calls: Alternative data models. In: IEEE Symposium on Security and Privacy, pp. 133–145 (1999)