An Empirical Study of Malware Evolution

Archit Gupta, Pavan Kuppili, Aditya Akella and Paul Barford
University of Wisconsin-Madison

***Abstract*—The diversity, sophistication and availability of malicious software (malcode/malware) pose enormous challenges for securing networks and end hosts from attacks. In this paper, we analyze a large corpus of malcode meta data compiled over a period of 19 years. Our aim is to understand how malcode has evolved over the years, and in particular, how different instances of malcode relate to one another. We develop a novel graph pruning technique to establish the inheritance relationships between different instances of malcode based on temporal information and key common phrases identified in the malcode descriptions. Our algorithm enables a range of possible inheritance structures. We study the resulting "likely" malcode families, which we identify through extensive manual investigation. We present an evaluation of gross characteristics of malcode evolution and also drill down on the details of the most interesting and potentially dangerous malcode families.**

I. Introduction

Malicious activity in the Internet is growing at an alarming rate. There are daily reports in the technical and popular press about new vulnerabilities and new types of attacks, and the rapidly increasing economic incentives are sure to catalyze this activity for a long time to come. Well known examples of malicious activity include denial of service, spam, information gathering, and resource gathering. In all cases, this activity is based on the use of software, also called "malcode" or "malware", that enables attacks to be carried out from one or more hosts distributed throughout the Internet.

Creating effective countermeasures for these threats is extremely difficult. First, the ever increasing complexity of networked systems and software means that it is difficult to build them to be inherently free from vulnerabilities. Second, network and end-host security today is realized as a patchwork of add-on software, features and capabilities that are unlikely to ever close all opportunities for intelligent and determined attackers. Third, and perhaps most significantly, the authors of malcode are well aware of the details of network and end host security mechanisms, and are developing increasingly sophisticated and effective methods for subverting them.

In this paper, we present a study of malware characteristics that focuses on key properties that are common to "malware families" and on the evolution of key malware characteristics over time. The goal of our work is to highlight trends in malware design that can broaden our understanding of the effectiveness of existing defenses, and inform the design of sound malware defense mechanisms in the future.

Our study is based on a large corpus of malware metadata derived from McAfee's threat library database. This metadata is used by McAfee for internal documentation, and is partially available to the community through its web site [17].[1] The metadata used in our study was compiled over a period of 19 years. The malware described in the data set is highly diverse in terms of sophistication of design, level of potency, methods of spreading, and vulnerability targets. For instance, some of the malware we study are documented as having spread via floppy diskettes!

To understand the trends in malware design, we mine the metadata for "relationships" between different instances of malcode and study how these relationships evolved over time. We consider two malware instances to be related when they share important characteristics. Examples of malware characteristics that can reflect relationships include targeting a common vulnerability, using a common method for scanning or denial of service and, especially, sharing specific pieces of code – a practice that is widely assumed to be common in malcode development. We refer to a group of malware that share many important characteristics as belonging to a single *malware family*.

We evaluate malware relationships in several different ways. We ask, for example, what is the lifetime and size of a typical malware family? What features are common to the malware belonging to long-lived families? How many instances of malware arise from a single potent "parent" (or a small set of parents) and what features do the "children" differ in? To the best of our knowledge, these aspects of malware relationships and evolution have not been systematically explored in prior work.

We face a key challenge in identifying malware relationships, grouping malware into families, and extracting the distinguishing features. Our study is based on metadata that describes specific instances of malcode and not on the malcode itself (we know of no openly available malcode repository maintained over a similar period of time). Most of the descriptions are entered in plain English sentences by human experts. Hence we are faced with the challenge of mining relationship information from *unstructured text*.

To address this challenge, we develop a novel analysis method that has two components: text mining and graph pruning. The first component identifies the *frequent significant phrases* in the textual descriptions of all instances of malware in our data set. This process enables suppression of common phrases (*e.g.,* "The following entry") that may cause us to infer spurious relationships between malware. More importantly, it exposes the key features of malware behavior expressed in specific sequences of text. After identifying the frequent

[1]Other anti-virus vendors have similar malware metadata available online (*e.g.,* [24], [25]).

significant phrases across all malware instances, we construct a "feature vector" for each instance based on the occurrence of specific significant phrases their textual descriptions. We use the feature vectors to estimate the "similarity" or the extent of the relationship between two malware instances. The salient feature of our text pruning and similarity inference approach is that we can draw useful observations on malware similarity without leveraging any domain-specific information.

The second component of our methodology, graph pruning, begins by considering all instances of malware as a fully connected graph. Edges Between malware instances are labeled with the extent of similarity, and are oriented temporally from the older instances to the more recent one. We prune the edges using two different pruning parameters and decompose the graph into *likely malware families*. The graph that results after the pruning process is a "forest" of malware family trees. The edges in this forest reflect both the temporal and feature-based relationships between instances of malcode. We explore the sensitivity of the pruning parameters and we also compare the resultant malware families with the *malware names* (*e.g.*, W32/Bagle.n@MM), that are included in the metadata. These labels provide a means for validating the malware families that result from our analysis.

Our analysis identifies many different intriguing characteristics of malware families and how they have evolved. Results with tuned parameters show that there are 669 distinct families in the metadata. Among these are well known families such as Mytob, Adware, W32/Bagle and lesser known such as Acid and Coco. We find instances of families that are very short lived (*e.g.*, a Loveletter family which lasts just 18 days), and others that persist for years. We find families that have a large fanout *i.e.*, many children after the root (frequently the case for well known malware such as Downloader) and others that are quite narrow (indicating a lack of viability of a particular family). One of the most interesting aspects of our analysis is how new malware families evolve from old families – we provide several examples of this phenomenon as well.

II. Related Work

There are many empirical studies of malicious activity in the Internet. Well known examples of these include [15], [18]–[22], [28]. These studies are often focused on a particular segment of malicious activity such as denial of service attacks or worm outbreaks, and the reports frequently coincide with the emergence of new threats. More recently, Freiling *et al.* [13] and Rajab *et al.* [23] provided empirical details on the escalation of botnet activity – one of the most potent threats in the Internet today. Similar to our work, these studies take advantage of a particular measurement infrastructure such as Dshield.org [27] or distributed honeypots [2] as a means for gathering data, and typically provide statistical characterizations of the data. The key difference is that we only rely on meta-data, not actual malcode (source or binaries) or measured behavior.

Evaluating the details of malcode binaries once they have been captured can provide interesting insights. However, it is challenging because malcode authors are increasingly using techniques to confound this kind of analysis. Disassemblers, debuggers and system monitors are common tools used by AV companies in the process of generating signatures and creating the metadata used in our study [4], [5], [8]. An alternative is to evaluate malware source code which can sometimes be found on the Web or in Usenet newsgroups (See [10] for a recent study of this kind).

Evolutionary properties of malware has also been studied (*e.g.*, [11], [12], [14]). Excellent, comprehensive reference material on malware can be found in [26]. Ma *et al.* present a study more closely related to our own that infers the phylogeny (*i.e.*, behavior characteristics) of malware shellcode [16]. Our work is most similar to these studies in that we too aim to establish evolutionary relationships between malware. In contrast with past studies, however, our work focuses on malware metadata spanning many years, which enables us to provide a long-term view into the evolution of malware features. We also evaluate the key properties of potent malware families.

From an algorithmic stand-point, our study is informed by prior work in data and text mining. Zaki describes an efficient algorithm for identifying frequent sequences in large data sets in [29]. While efficiency is less of an issue in our work, methods for finding all frequent sequences such as those described in [9] are important.

III. Malware Families

Our dataset contains a wealth of information on a large number of malware instances. Almost all of this information is entered by hand, after capturing and deconstructing the malicious code in a controlled environment. Since the information is entered manually, it is semantically rich and quite detailed. However, not all malicious code is described in the same detail or using the same English constructs.

In this section, we begin by providing details on the malware dataset. We then describe the first component of our analysis, which creates a uniform schema for malware descriptions based on *text mining*. The schema provides a baseline from which we can establish relationships between malware instances. Then, we describe our graph pruning algorithm that we use to generate "likely" malware families.

A. Description of the Dataset

The malware meta data that we evaluate is the McAfee Avert Labs Threat Library [17]. There are 44,504 malware instances in the database, spanning a period of 19 years from 1987 to 2006. The database schema provides a variety of information - *e.g.*, the malware name, discovery date of the malware, size of the malware (in bytes), malware type, "danger" of payload, prevalence, etc. However, from the point of view of inferring relationships between malware, the most useful fields are those with semantically rich textual descriptions. There are three such fields in the database: malware characteristics, methods of infection, and indications of infection. A total

of 8,182 malware instances include sufficiently detailed text descriptions and our analysis focuses on these.

B. Challenges in Mining the Data

While the textual descriptions in the three key fields of the database provide fascinating information, they carry no specific structure or organization. Thus, we are faced with the challenge of systematically organizing these descriptions with as little human input as possible, and uncovering the most informative properties of malware. This is crucial in order for us to infer the relationships between malware with high confidence. Our first insight is that we can use techniques from Information Retrieval to mine the most defining properties of the malware. We elaborate on this in Section III-C.

A second insight we use is that the information obtained from unstructured data when combined with other structured information can provided interesting views into the evolution of malware. As an example, we use the discovery date to "orient" the edges in our malware relationship graph and to indicate the possibility that the newer of the two malware was spawned from the older one. We describe this in Section III-D.[2]

C. Schema Discovery Via Text Mining

In order to make the unstructured textual descriptions more useful toward the goal of establishing malware relationships, we create a suitable schema and convert the textual descriptions into a set of informative tuples. Each malware instance is mapped onto a tuple containing fields that describe key properties.

To obtain the appropriate schema from the textual descriptions, we use *frequent phrase extraction*. We consider phrases which occur frequently throughout textual descriptions of all instances of malware in the database, and make each such phrase a column in the schema. Then, each malware instance is represented by a tuple in the schema and the columns take boolean values indicating the presence or absence of the corresponding phrases in the description of a given instance. Since our goal is to use the schema to identify relationships (based on similar properties) between malware instances, frequent phrases is a better choice than a standard bag-of-words schema.

We used a frequent phrase extraction tool from the IR community, Extrphr32 [3], which extracts the *maximal* frequent phrases: a maximal frequent phrase is not a substring of any other frequent phrase. The tool also considers only those phrases which do not start or stop with "stop words", which are common English words like "the", "but", etc. To determine if a phrase is frequent, we use a parameter σ.

Frequent phrase extraction was applied to the "virus characteristics", "methods of infection", and "indications of infection" text fields in the database. We experimented with a range of values and selected $\sigma = 30$ which yielded ≈ 6500 frequent phrases. However, some of these were common English phrases that were clearly not specific malware properties. To eliminate these phrases, we developed an approach for automatically determining if a phrase is an ordinary English phrase or something relevant to malware and security. To do this, we use the Google search engine, which is likely to be one of the world's largest on-line repositories of English phrases. We submitted each of the 6500 common phrases as a Google search query. Intuitively, we expect common English phrases to have a significantly higher number of Google results than the malware-specific phrases. The scatter plot of the phrase frequencies is given in Figure 1. The common English phrases are typically found in hundreds of millions of documents. Based on the scatter plot, we chose 10 million as a cutoff to retain most of the technical information while eliminating most of the common English phrases. After this filtering, we were left with ≈ 4500 phrases. A few examples of the phrases that were used in our schema follow: (1) "This virus constructs messages using its own SMTP engine. Target email addresses are harvested from files". (2) "memory resident at the top of system memory but below the 640k dos boundary, hooking interrupt 21.". (3) "Its spreading activity remains only in the german language version of microsoft word. However, the virus may be able to execute its payload in another language version."

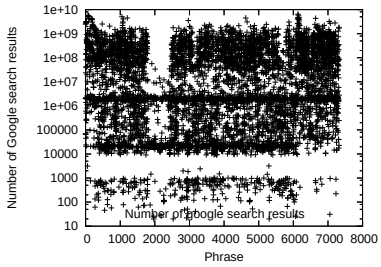


Fig. 1. Scatter plot of frequent phrases in the malware meta-data based on frequency of occurrence in Google search results.

Using the resultant schema, we create and populate a table that contains the most significant defining features for each malware instance.

D. Establishing Malware Families

Our goal is to identify malware families based on shared key properties. In general, we say that two malware instances are related when they share several important characteristics, such as method of infection, method of spreading, and the symptoms exhibited by infected hosts. Based on anecdotal evidence [7], we speculate that such similarities arise for one or both of the following reasons: (1) The authors of the malicious code share some routines with each other. This practice of code sharing is known to be common in the black hat community. Or, (2) The authors of the malware create code that has similar observed behavior (*e.g.,* exploit the same set of vulnerabilities), but there is no explicit code sharing. While there could be other reasons, we argue that identifying relationships on the basis of shared properties is of intrinsic value in understanding malware and its characteristics, and can

[2]Our analysis ignores other structured fields, such as the potency and the prevalence of malware. This is because we want to focus primarily on understanding how the different malware families evolve over time.

be valuable in the design of defense mechanisms.

Our goal is to derive a graph G', whose vertices represent the unique malware instances in our dataset, and whose edges imply that the corresponding malware instances are strongly related. Also, we wish to make the edges of the graph directed in order to indicate an evolutionary relationship between malware instances.

The graph G' is actually a collection of multiple different *malware families*. The graph G' should satisfy the following properties: (1) A node n should only have incoming edges from a set S of previously seen malware only if n is "strongly related" to S. (2) There should be no spurious or unnecessary edges.

We begin with a completely connected graph G with directed, weighted edges: each vertex in this graph is a malware instance, and the weight of an edge is the similarity between the sets of frequent phrases associated with the malware instance. Edges in G point from the older instance to the more recent instance based on the "time of discovery" field for each. The similarity metric ranges between 0 and 1 and is defined as follows: For an edge $A \rightarrow B$ between two malware instanced A and B, let $S(A \rightarrow B)$ be the set of properties in the frequent phrase database which are common between A and B. The weight of the edge is $\frac{|S(A \rightarrow B)|}{|B|}$, where $|B|$ is the number of defined properties in B and $|S(x)|$ is the cardinality of set $S(x)$. Thus, the similarity metric captures the fraction of B's properties which are also shared with A.

Note that if $|B|$ is very small, then we do not have sufficient information to relate it to other malware. To prevent instances like these from corrupting our relationship graph, we introduce a parameter γ. If $|B| < \gamma$, we do not consider B in creating the final graph G'. We experimented with a range of values and settled on $\gamma = 10$ for our study. The result was that 2700 malware instances were eliminated from consideration. We concluded that this is was reasonable since manual examination of these instances showed that their descriptions were, in fact, quite minimal.

Next, we describe our technique for systematically deleting edges from the graph G to obtain the graph, G' that exposes significant sharing between malware instances.

To achieve the desired property that edges are incident on a node only if it shares a significant number of features with some previously seen set of malware, we introduce a pruning parameter δ_1. Consider a malware instance A. Assume $|A|$ (the number of features for A) is sufficiently high and a large fraction ($> \delta_1$) of its properties are in common with a set S of malware instances with earlier timestamps. This means that A shares several key features with the malware in S, and it is therefore likely to be closely related to or even derived from the malware in S. If, on the other hand, the total contribution of these incoming edges is less than δ_1, then we can consider the malware to have evolved independently.

However, δ_1 by itself is not sufficient since spurious edges are still possible. For instance, consider the case where malware instances B and C both evolve independently from A. Assume both B and C are similar to A, and similar to each other, and that B has an earlier time stamp than C. With just the δ_1 parameter, we will have two incoming edges: one from A, and one from B into C. Likewise if A spawns a number of children, we will have many spurious edges among the children of A. We need to remove these spurious edges from the graph.

We introduce a second pruning parameter δ_2 to address the spurious edge problem. If an edge does not *uniquely* contribute more than δ_2, we prune the edge. Specifically, say a node C has a pair of incoming edges $A \rightarrow C$ and $B \rightarrow C$. We check if $\frac{|S(A \rightarrow C) - S(B \rightarrow C)|}{|C|} < \delta_2$ (the numerator in the inequality computes the set difference). If this inequality is satisfied, we delete the incoming edge with the lesser weight.

Thus, the complete graph pruning algorithm is as follows. For each node, incoming edges that do not have a sufficient unique contribution are pruned using the δ_2 parameter. Next, all of the remaining incoming edges are pruned if they together do not contribute at least δ_1 fraction of the node's properties. A formal description of our graph construction algorithm is given in Figure III-D.

It should be noted that the algorithm allows for the possibility of a given malware instance being related to or having evolved from multiple parents. This kind of "bundling of threats" has, in fact, been observed in the wild. Note that $\frac{1}{\delta_2}$ is the maximum number of incoming edges on any malware instance. Thus δ_2 captures the maximum number of parents allowed for any node in the graph.

Obtaining G'

1. Each malware instance is a vertex. Start with a completely connected graph. The direction of each edge is from the older malware to the more recent one.

2. For a node i with $k =$ the set of incoming edges
   Consider all ${}^kC_2$ combination pairs of incoming edges.
   For every incoming edge pair (x, y),
   if $weightminus(x, y) < \delta_2$
   Delete either x or y
   (delete the one that has a lesser weight).

3. For a node i with $r =$ the remaining set of incoming edge
   If $weightunion(r) < \delta_1$, delete all the edges in r.

4. Repeat steps 2 and 3 for all nodes.

Fig. 2. Constructing the malware relationship graph. If a node C has a pair of incoming edges x and y, define $weightunion(x, y)$ as $\frac{|S(x) \cup S(y)|}{|C|}$ and weightminus(x, y) as $\frac{|S(x) - S(y)|}{|C|}$.

The final graph G' should contain several directed acyclic sub-graphs. In each component sub-graph, there will be one or more malware instances with zero edges incident on them. We refer to these instances as *roots*.

Sub-graphs with more than one root are caused by nodes with multiple parents at the lower levels. We split these components into as many families as the number of roots. In doing this, we assign nodes with multiple parents to the family which contributes the "heaviest" edge incident on the malware instance. We perform this decomposition of our relationship graph G' into malware families only because it allows us to

simplify our analysis and speak of "likely" malware families. With this decomposition, we can now study key properties of the likely families, such as the total number of malware instances, the life-span of the family, the total number of generations, etc.

Evaluation of the malware family trees in Section III-F indicates that our algorithm works well. However, it is possible that malware instances that are classified into different families by our algorithm are actually related. Indeed, when we examine the resulting families closely (see Section V), we noticed that several well-known malware instances (*e.g.*, instances of `Bagle` malware) are spread across multiple families. However, our classification ensures that members within a family are much more likely to be strongly related to each other than to members across families.

E. Parameter Selection

Our algorithm uses two pruning parameters δ_1, and δ_2. The characteristics of the resultant malware families hinges crucially on how these parameters are chosen. An overly permissive setting for these parameters (low values for both) could cause us to infer relationships where there are none. An overly conservative choice may cause us to miss important relationships.

We systematically consider the impact of the values for the δs. A very low value of δ_1 will lead to a completely (or almost completely) connected graph, while a very high high value of δ_1 will result in a graph with no edges. A very low value of δ_2 can result in a huge number of parents, while a very high value of δ_2 will result in no node having more than one parent. In Table I we illustrate the effect of 20 different parameter choices on the entire relationship graph (more instances were considered but these are representative). The candidate choices for δ_1 and δ_2 lie on the more conservative side in order to to identify several "good" relationships accurately.

We analyzed four features of the relationship graph: the overall number of single-parent and multiple-parent nodes, the number of roots and the total number of nodes that do not belong in any family (*i.e.* isolated nodes - neither have incoming nor outgoing edges). First, as expected, the number of single-parent nodes decreases with higher values of δ_1 and is unaffected by δ_2. The number of multiple-parent nodes drops drastically with δ_2 and becomes insignificant when $\delta_2 > 0.3$. The total number of roots does not show a clear monotonic trend as it has a complex dependence on both δ_1 and δ_2. On the other hand, the total number of isolated nodes increases with δ_1 (and δ_2), with the total number at $\delta_1 = 0.9$ almost 50% higher than the number at $\delta_1 = 0.7$.

Since we would like to have as few isolated nodes as possible (to prevent the families from becoming degenerate), while keeping our parameter choice as conservative as possible (to eliminate spurious edges), we chose $\delta_1 = 0.7$ and $\delta_2 = 0.3$ to establish the family trees evaluated in Sections IV and V. We note that a few different parameter choices (*e.g.* (δ_1, δ_2) = (0.6, 0.3), (0.7,0.3)) provide roughly similar trade-offs.

F. Robustness and Validation

It is important that the parameter choice be robust to the amount of data in the database. To verify this, we split our database into 3 random parts, each of size 2727 nodes. We ran the algorithm independently over these three subsets, and generated the statistics in Table I for the number of single-parent nodes, multiple-parent nodes, roots, isolates, and entropy (defined below). In each subset, the statistics follow essentially the same trend as in Table I. For instance, the number of single-parent nodes decreases with the increase in δ_1. There are almost no multiple-parent nodes in all three cases when $\delta_2 > 0.3$. Likewise, the number of roots follows a very similar trend to the above table, increasing up to certain values of δ_1 and δ_2, and then falling. The number of isolated nodes in all three tables increase sharply with the increase in δ_1.

A complete validation of all of the malware families that are generated by our algorithm is not possible. Furthermore, even the task of quantifying the accuracy of the relationships we infer is challenging. Both of these limitations arise because we only have access to the malware meta-data. We focus on the latter issue in this paper because it is more tractable.

One approach we considered was to verify that malware instances identified as being related also share similar names. The experts who generate the malware metadata attempt to name malware instances according to a group that they believe it belongs to. For example, most variants of W32/Gaobot are named W32/Gaobot.*.*. To verify that the malware classification due to our algorithm aligns with the names provided by McAfee, we developed an "entropy metric", described below.

Assume our algorithm generates k families $T_1, .., T_k$. Also assume there are N families of malware according to names in the McAfee database. If a McAfee family i has n_i members, and these are distributed across the k families output by our algorithm. If, for the McAfee family i, the fractions of its members across our k families are f_1, .., f_k, then the entropy of that family in our classification is $e_i = \Sigma_{j=1}^{k}(-f_j log f_j)$. The mean entropy of all the McAfee families will be $\frac{\Sigma_{i=1}^{N} n_i * e_i}{\Sigma_{i=1}^{N} n_i}$.

The entropy value lies between 0 and $log_2 k$. The entropy value will be smaller if members of the same McAfee family are assigned to a single family generated by our algorithm. Note that this metric is based only on the malware instances that have been classified into named families by McAfee. Note also that this metric does not quantify the evolutionary patterns. Nonetheless, it provides a reasonable sanity check.

For $(\delta_1, \delta_2) = (0.7, 0.3)$ we obtain $k = 669$ families. Our algorithm has an entropy of 1.19 in this case (see Table II). Drawing a simple analogy, the entropy metric can be interpreted as the mean number of binary questions to ask in order to decide which family a malware instance is placed in. Thus, for the members of a McAfee-named family, we need to ask only ~ 1 question (equivalent to deciding between 2 families) which we believe is fairly reasonable. (A bad algorithm can end up with an entropy value of $log_2 669 = \sim 9$).

We used the McAfee names only during the entropy validation and do not use them anywhere in our algorithm since

$\delta_1 \rightarrow$ $\delta_2 \downarrow$	0.6	0.7	0.8	0.9
0.1	3767	2962	2015	1170
0.2	3767	2962	2015	1170
0.3	3767	2962	2015	1170
0.4	3767	2962	2015	1170
0.5	3767	2962	2015	1170

(a) Num. single-parent nodes

$\delta_1 \rightarrow$ $\delta_2 \downarrow$	0.6	0.7	0.8	0.9
0.1	1362	1748	1832	1174
0.2	463	419	304	134
0.3	66	65	42	14
0.4	6	6	6	2
0.5	0	0	0	0

(b) Num. multiple-parent nodes

$\delta_1 \rightarrow$ $\delta_2 \downarrow$	0.6	0.7	0.8	0.9
0.1	282	565	1061	1495
0.2	600	791	794	622
0.3	513	669	612	474
0.4	474	620	578	460
0.5	476	620	575	458

(c) Num. roots

$\delta_1 \rightarrow$ $\delta_2 \downarrow$	0.6	0.7	0.8	0.9
0.1	2771	2907	3274	4343
0.2	3352	4010	5069	6256
0.3	3836	4486	5513	6524
0.4	3935	4594	5583	6550
0.5	3939	4600	5592	6554

(d) Num. isolated nodes

TABLE I
THE EFFECT OF δ_1 AND δ_2 ON THE RELATIONSHIP GRAPH.

$\delta_1 \rightarrow$ $\delta_2 \downarrow$	0.6	0.7	0.8	0.9
0.1	0.56	0.97	1.44	1.97
0.2	1.26	1.31	1.34	1.48
0.3	0.93	1.19	1.22	1.36
0.4	0.91	1.15	1.19	1.45
0.5	0.91	1.14	1.20	1.45

TABLE II
THE ENTROPY IN THE NAMES ASSIGNED BY MCAFEE.

we do not want our family trees to be biased by the McAfee names. It is important to note that simply using the McAfee names to construct families is not very helpful. There is no clear way of deciding the edges within members of a single McAfee name-based family. Furthermore, having an algorithm independent of the names helps us use the names for validating the algorithm. More importantly, a significant portion of the malware instances do not fall into *any* McAfee named family. For instance, among the 3092 edges (in the $\delta_1 = 0.7, \delta_2 = 0.3$ case), 469 edges are between members of the same McAfee named family, 611 edges are between members of different McAfee named families. The remaining 2012 edges are incident on malware neither of which fall in any McAfee named family. So, these 611 + 2012 edges are completely new in the sense they cannot be inferred by just looking at McAfee names. The 611 edges between different McAfee named families are also interesting. Though some of them might be false positives, the others might give useful new insights into how the different families evolved from one another. We give some examples of these edges in the family descriptions in Section V, while leaving an in-depth inspection of all these edges for future work.

To further examine the quality of our inferred relationships, we augmented the above checking of names with a manual check of the accuracy of some of the relationships. Specifically, for each family we obtained, we first checked if most of the members share a common McAfee name prefix (*e.g.* W32/Gaobot). If this check is not satisfied, we manually check the meta-data to see if we erroneously inferred a relationship. In almost all the cases, we observed strong similarities between a "parent" and its "child". Additionally, some well known malware evolutions were also observed in our families. For example, our families expose the evolution of `Mytob` from `Mydoom` and `Zotob` from `Mytob`, both of which are well documented in the popular and technical press.

In Section V, we show the trees for some of the largest families we identified. A complete list of the families we inferred may be found at our Project Web site [1].

IV. CHARACTERISTICS OF MALWARE EVOLUTION

As mentioned earlier, 8182 malware from the original malware database had a non-trivial textual description. Our analysis focuses on this subset. After employing the parameters described in Section III and applying the tree decomposition process, we identified 669 distinct malware family trees. Overall, 4486 malware instances were not classified into any family because of our choice of pruning parameters. The trees included 2962 "single-parent" nodes and 65 "multiple-parent" nodes. The analysis in this section is focused on examining the key properties of the 669 families, with particular emphasis on the larger families.

A. Family Tree Size and Fanout

Figure 3(a) shows the cumulative distribution of the number of malware instances in each family tree. A large number of trees are quite small: $> 90\%$ of the trees have less than 12 nodes and 50% of the trees have just two nodes (a single edge). It is interesting to note that a handful of the families among those identified are very large: 8 of the families have more then 50 malware instances each and the maximum number of malware instances in a family is 119! Later in this section, we delve deeper into some of the properties of 2 large trees. In Section V, we study the key features that are retained across generations of malware in some of these large families.

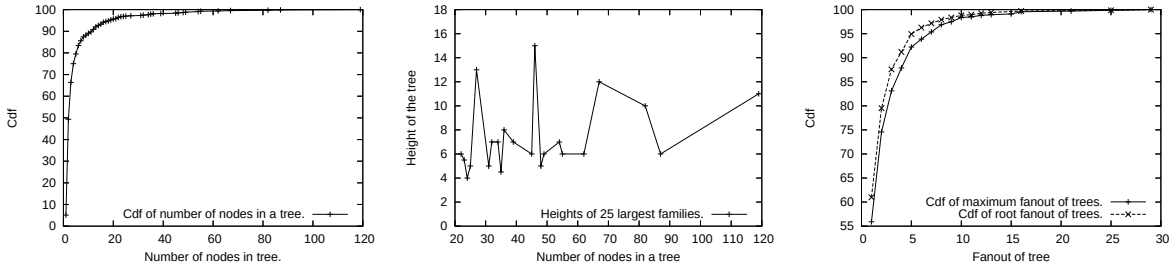
Next, we consider the number of "generations" in each malware family. It is likely that a new generation malware is developed in response to specific counter-measures that were developed by AV companies to contain the previous generation. Thus, this analysis provides a perspective on the arms race between malware authors and AV experts. In Figure 3(b), we show the distribution of the heights/generations of the 25 largest family trees. On average, these families span 7 generations. In Section IV-B, we study the time-span of these generations, and find that some of them span a few years.

Figure 3(c) shows the distribution of the maximum fanout in family trees, as well as the fan-out of the root of the trees. In several families, each malware instance spawns few other malware instances over time: 92% of the trees have a maximum fanout less than 5; and 95% of the roots have a fanout less than 5. However, we do observe three trees with a maximum fan-out > 20, and the maximum fanout observed in a tree is 29.

Since the root and maximum fan-out distributions are different, it is clear that the root does not necessarily have the largest fan-out. This may happen when some intermediate malware bundles together the capabilities of several of its predecessors and in turn becomes the source for multiple future strains.

B. Family Tree Life-span

Each edge in the graph we obtain has an associated *length*. We define the length of an edge as the difference in the "time



(a) (b) (c)

Fig. 3. Figure (a) shows the distribution of the number of nodes in family trees. Figure (b) shows the height of the largest 25 family trees. Figure (c) shows the distribution of the maximum and root fan-outs in family trees.

of discovery" field in the two malware instances; we measure the length in days. Edge lengths help us understand the time duration over which the the successors of potent malware instances are developed and released.

Figure 4(a) shows the distribution of the lengths of all edges in derived family trees. The figure shows that 90% of the edge lengths are less than 730 days long. Thus, it appears that most malware are spawned from their predecessors in under two years.

Figures 4(b) and (c) show scatter-plots of the fanout of malware instances versus the mean and the minimum length of their outgoing edges, respectively. These plots helps us understand the correlation between the popularity of a malware instance - defined in terms of how many immediate successors are spawned from it - and the time to the evolution of its successors. An interesting trend is evident: malware instances with a high fanout do not have any long outgoing edges. In other words, it seems that malware instances that spawn a lot of children (perhaps because the malware's source code was reused very frequently), do so relatively quickly. Focusing on malware that spawns few successors, we note that a much longer time may elapse before they spawn their first successors. For example, in Figure 4(c), there are several cases where the minimum edge length is > 1000 days for malware with a fan-out < 5.

Figure 4(d) shows the mean length of edges at each depth in family trees. This can be interpreted as the mean time for one generation to evolve into the next. As can be seen from the figure, the mean time for the generations to evolve seems to fall exponentially with the generation number.

C. Evolution Dynamics of Malware Families

We now consider the temporal patterns in the "birth", life-span, and "death" of malware families. Our focus is on understanding how these evolutionary dynamics have changed over the past decade or so. We believe that this analysis sheds more light on the effect of two concurrent phenomena on the overall prevalence of malware: (1) the ongoing race between malware code writers, and the anti-virus companies, which could cause some families to have a very long life time; and (2) the improvements in operating system and application software security, which could cause the death of some families.

Figure 5(a) shows the time line of the distribution of "time of discovery" fields in the malware, binned by the year. This graph includes all 8182 malware instances have rich text descriptions. Thus, this plot includes even the isolated nodes in our relationship graph. The key observations are the acceleration in new instances since 2001 and dips between 1994-1996 and in 1998.

Figure 5(b) shows the distribution of the lifetime of the 25 largest malware families as well as the distribution of the lifetimes of all families. The lifetime of a family is defined as the difference in the timestamp of the root and the timestamp of the most recent member of the family. We observe that 80% of all families have a lifetime less than 730 days (around 2 years). However, a small fraction of the families, roughly 9%, last more than 1960 days (around 5 years). Among the top 25 families, ~ 50% have a lifetime greater than 950 days (around 2.5 years), and 18% of them have a lifetime more than 2350 days (around 6.5 years). Thus, it appears that the large families also have very long lifetimes.

We consider families that were last seen in 2006 to be "active". These families have not been included in the aforementioned lifetime distributions. There are 68 "active" families, of which 46 have more than 2 nodes. Furthermore, we found that 9 of the 25 largest families are still active. The 46 active families with > 2 nodes have been in existence for mean time of 544 days.

Figure 5(c) delves deeper into the malware evolution dynamics. Here, we compare the cumulative counts over time of the total number of trees born and the total number of trees that died since the first family originated in 1987[3]. The gap between the two cumulative counts indicates the number of families alive in a given year. Note that there is a huge gap between the two plots in the early 1990s, indicating that a lot of families were active. By 1998, the two curves almost meet indicating (perhaps) that most of the vulnerabilities of the early nineties were patched. Along the same lines, we can infer that the early 2000's saw a revival of sorts in malware exploits (the two plots are almost parallel). These observations are supported by the anecdotal evidence on the prevalence of malware in recent years.

The right half of Figure 5(c) is even more interesting: the slope of the "birth of trees" curve becomes very steep, yet the gap between the two curves remains roughly fixed. This is representative of the ongoing tussle between malware authors and AV companies. For every malware family the AV com-

[3]By "dying" we simply mean that there are no further generations of the family, not that the malware is removed from the Internet

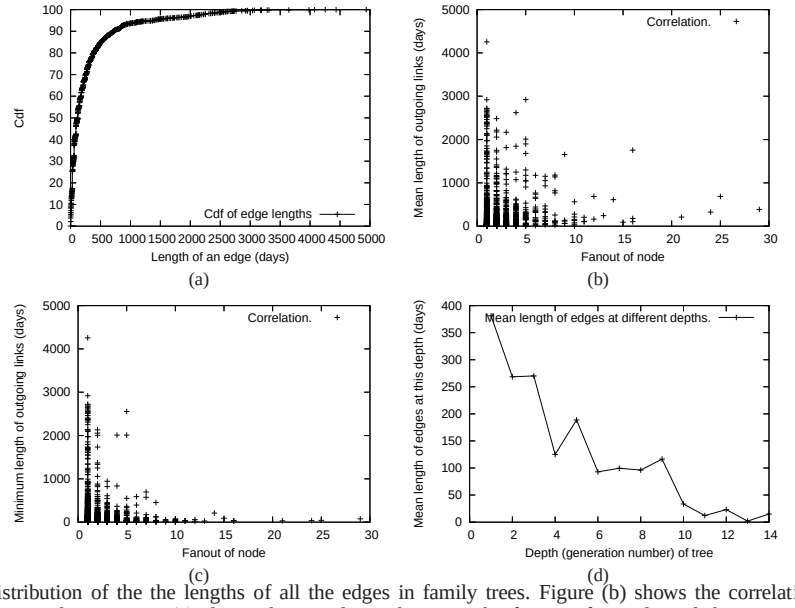


Fig. 4. Figure (a) shows the distribution of the the lengths of all the edges in family trees. Figure (b) shows the correlation between the fanout of a node and the mean length of its outgoing edges. Figure (c) shows the correlation between the fanout of a node and the minimum length of its outgoing edges. Figure (d) shows the correlation between the depth of a tree and the mean length of edges at that depth.

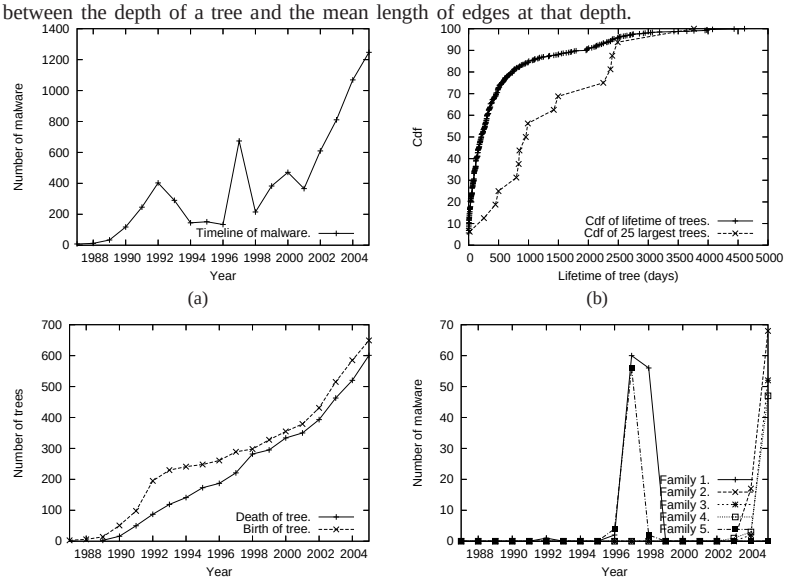


Fig. 5. Figure (a) shows the number of malware instances born each year. Figure (b) shows the distribution of lifetimes for all family trees and for the 25 largest family trees. Figure (c) shows The cumulative count of the number of families born and the number of families that die. Figure (d) shows the timeline of the five largest malware families.

panies eliminate, malware authors are able to come up with newer families that (possibly) exploit newer vulnerabilities.

Finally, we focus on the 5 largest families and dig a bit deeper into their life-spans (a more in-depth analysis is presented in the next section). Figure 5(d) shows the timelines of the five largest families. The largest family was chiefly active in 1997 and 1998. The members of this family do not have any specific McAfee family name. Most of the members of this family are viruses that infect files. These viruses mostly spread via floppy diskettes and online downloads. The second and fourth largest families are Adware/Spyware/Keylog families. They were widely active in 2005 (68 and 47 malware respectively in 2005). They continue to be active in 2006 and must be monitored closely. Likewise, the third largest family (a Downloader family) was widely active in 2005 (52 malware). The fifth largest family was a Word Macro family which was mainly active in 1997, and died after that.

V. A Deeper Look at Some Malware Families

The malware families we identify are useful to domain experts as they can aid in the development of strong countermeasures for future malware. We believe that understanding these issues is important to evaluate the effectiveness of malware defenses developed over the years.

In this section we drill down on the details and unexpected characteristics of some of the largest families we identified in by our algorithm. We name each family according to the most commonly appearing McAfee-assigned name across all

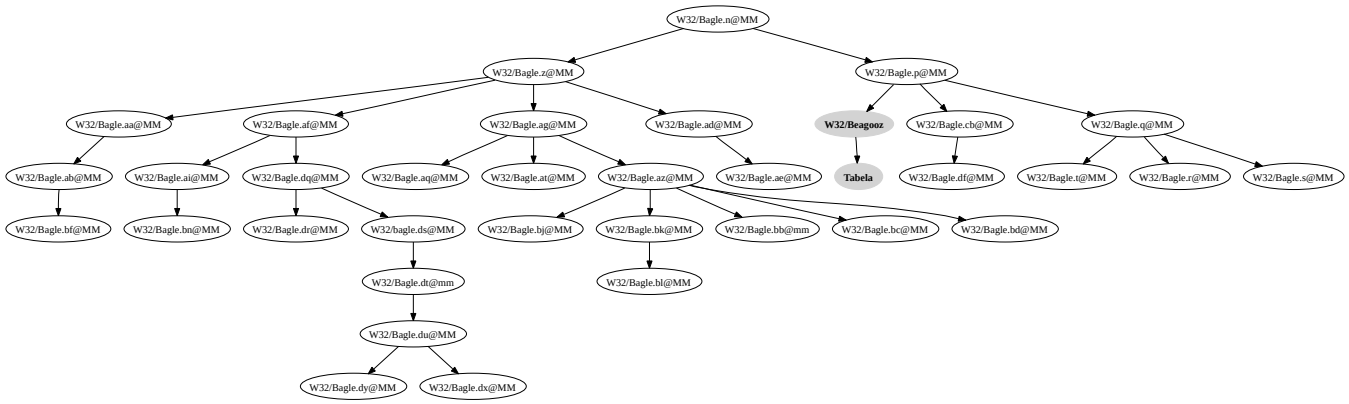


Fig. 6. A Bagle family tree. The first instance (W32/Bagle.n@MM) appeared on 03/13/2004 and the last instance (Tabela) appeared on 04/15/2006.

malware instances in the family. We show graphical representations of one of the malware family trees. The rest are available at [1] due to space limitations.

A. A `Mytob` *Family*

The "Mytob" malware instances are spread across several families in our classification. We discuss here the largest Mytob family. An important feature of this tree is that in addition to Mytob instances, Zotob instances are evident. It has been reported the popular press that the same hacker responsible for creating the Zotob worm also authored several variants of the Mytob family [6]. It is instructive to note that our classification algorithm is able to unearth such evolutionary trends without the direct aid of specific text describing the evolution. There are three other small families in which Mytob instances appear, and these families show relations between Mytob and Mydoom, Polybot, and Gaobot (Gaobot also spawns Sdbot in the same family) malware. It is well known that Mytob is a variant of Mydoom. The largest Mytob family has 46 nodes, a height of 15 (the maximum among all the Mytob families), and a maximum fanout of 8.

Most of the Mytob variants in this family spread via email. We examined some of the phrases which were common across the different generations of the malware in this family. We noticed phrases such as "sender address", "mass mailing worm", "mail propagation", "arrives in an email message", "via SMTP", "via SMTP constructing messages using its own SMTP engine the worm guesses the recipient email server prepending the target domain", and "worm contains strings which it uses to randomly generate or guess email addresses these are prepended as user names". Looking at the common phrases at the various depths can also provide useful information. For instance, the backdoor component was added in the 8th generation of the family.

Another interesting aspect of this family is its structure. This family is dominated more by depth than fanout, and it has 15 generations. This suggests that the Mytob authors are designing new variants to get past the latest defenses. The family is more or less a linear chain in the initial parts. Then `W32/Mytob.cv@MM` malware instance spawns 8 children. It takes 2 months from the root to `W32/Mytob.cv@MM`. This suggests that it took roughly around two months before a Mytob instance became popular and spawned new variants. Two of `W32/Mytob.cv@MM`'s children, `W32/Mytob.eu@MM`, and `W32/Mytob.dl@MM` seem to be spawning important strains and the family appears to be evolving along these two main sub-families.

B. A `Bagle` *Family*

The W32/Bagle.* collection of malware is one of the most prolific in our entire dataset. In our classification, we found that the Bagle malware instances were spread across fourteen different families. The biggest among the families is shown in Figure 6. This family has 36 nodes, a depth of 8 and a maximum fanout of 5. The non-Bagle members in this family are `W32/Beagooz` and `Tabela`. Some of the phrases which were common to the malware in this family include "address is spoofed", "worm opens", "mail propagation", "copies itself to folders", "contains its own smtp engine", "contains a remote access component", "peer to peer applications", "kazaa bearshare limewire", "email addresses are harvested", "spam", and "mass mailing worm". As is well known (and as the phrases seem to indicate), the Bagle variants copy themselves to the shared folders of popular peer-to-peer applications.

The malware instances belonging to this family were most active 2004, with 22 variants being discovered in that year. Only six new variants were discovered in 2005, and eight in the first few months of 2006. This may suggest that the malware had lost some of its prevalence, but there are a few unpatched vulnerabilities that continue to be exploited by the new variants of Bagle. We speculate that the high number of variants in 2004 may be due to the rise and popularity of peer to peer applications around that time, and that the drop in 2005 may have come about due to the community's growing awareness of the security problems associated with popular peer-to-peer applications.

VI. Conclusions

Users throughout the Internet are plagued by malicious attacks on an on-going basis. The task of defending against these attacks is complicated by many factors, including complexity, scale, and the increasing sophistication of malware authors. The premise of our work is that an expanded perspective on malware behavior and in particular the relationships between

malware variants will eventually lead to the development of more effective countermeasures.

In this paper we present an analysis of malcode meta-data compiled by McAfee, one of the largest AV companies in the world. The meta-data describes malware that was collected by McAfee and other AV companies over a period of 19 years. The objective of our work is to identify and evaluate relationships between malware instances based on the details of their descriptions. We do this through a process that begins by decomposing the descriptions into frequent phrases, and then pruning the resulting set to eliminate the superfluous phrases. Next, we establish relationships between instances of malware using a tunable graph pruning algorithm that is based on the similarity of frequent phrases between all instances of malware in our data set. In our analysis, we show the trade offs in graph structure using different parameter settings and select a configuration that results in a graph that we validate as being "likely" using the malware names applied by McAfee.

The resulting families have rich structure. We identify 669 distinct malware families. Some of the families are very large, containing in excess of 50 members. We found that some of the families were active for a few years at a stretch, while others last no more than a few days. Detailed examination of the families reveals many instances where specific traits (as identified by a specific phrase) are inherited after many months and that one instance of malware may spawn many others. We believe that the malware families identified by our algorithm are useful to domain experts and can aid in developing proactive strategies to counter malware attacks. Another application of our technique would be to identify similar instances of malware in different repositories that use different naming strategies.

The malware families are available at our project web-site ([1]) for general perusal. We plan to pursue several extensions to this work. First, we hope to expand the corpus of malware meta-data in order to flesh out the evolutionary characteristics of malware in greater detail. Second, we believe that adding the behavioral characteristics such as those identified in [16] and others will further enrich our analysis. Finally, we will work more closely with AV companies and others concerned with malware analysis, to develop methods for anticipating future trends in malware development. We hope that this will enable AV companies to generate malware counter-measures in a more proactive fashion.

ACKNOWLEDGEMENTS

The authors would like to thank Joe Telafici and McAfee Avert Labs for generously providing access to threat library database used in this study. This work was supported in part by NSF grants CNS-0626889, CNS-0716460 and CNS-0831427. Any opinions, findings, conclusions or other recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the NSF.

REFERENCES

[1] Malware Evolution Research at UW-Madison. http://www.cs.wisc.edu/∼archit/projects/malware/.

[2] The Honeynet Project. http://project.honeynet.org, 2003.

[3] ExtPhr32 - Frequent phrase extraction tool. http://instruct.uwo.ca/gplis/677/extphr32/extphr32.htm, 2005.

[4] Regmon. http://www.sysinternals.com, 2005.

[5] The SoftICE Driver Suite. http://www.compuware.com, 2005.

[6] Zotob and Mytob were originated by Russian hacker. http://www.crime-research.org/news/30.08.2005/1462/, 2005.

[7] Malware dangers grow as e-criminals pool resources. http://www.itweek.co.uk/itweek/news/2160344/malware-dangers-grow-criminals, 2006.

[8] The IDA Pro Disassembler and Debugger. http://www.datarescue.com, 2007.

[9] H. Ahonen-Myka. Mining All Maximal Frequent Word Sequences in a Set of Sentences. In *Proceedings of ACM International conference on Information and Knowledge Management*, New York, NY, October 2005.

[10] P. Barford and V. Yegneswaran. *An Inside Look at Botnets*, volume 27 of *Advances in Information Security, Malware Detection*. Springer, 2007.

[11] M. Eichin and J. Rochlis. With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988. In *Proceedings of IEEE Security and Privacy*, Oakland, CA, June 1989.

[12] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff. A Sense of Self for Unix Processes. In *Proceedings of IEEE Security and Privacy*, Oakland, CA, June 1996.

[13] F. Freiling, T. Holz, and G. Wicherski. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks. In *Proceedings of The 10th European Symposium on Research in Computer Security (ESORICS '05)*, September 2005.

[14] J. Kephart and S. White. Directed-Graph Empdemiological Models of Computer Viruses. In *Proceedings of IEEE Security and Privacy*, Oakland, CA, June 1991.

[15] A. Kumar, V. Paxson, and N. Weaver. Exploiting Underlying Structure for Detailed Reconstruction of an Internet Scale Event. In *Proceedings of ACM Internet Measurement Conference*, Berkeley, CA, November 2005.

[16] J. Ma, J. Dunagan, H. Wang, S. Savage, and G. Voelker. Finding Diversity in Remote Code Injection Exploits. In *Proceedings of ACM Internet Measurement Conference*, Rio de Janeiro, Brazil, October 2006.

[17] McAfee. Avert labs threat library. http://vil.nai.com, 2007.

[18] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Stanifor, and N. Weaver. The Spread of the Sapphire/Slammer Worm. http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

[19] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Stanifor, and N. Weaver. Inside the Slammer Worm. In *Proceedings of IEEE Security and Privacy*, Oakland, CA, June 2003.

[20] D. Moore, C. Shannon, and J. Brown. Code Red: A Case Study on the Spread and Victims of an Internet Worm. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.

[21] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial of Service Activity. In *Proceedings of the 2001 USENIX Security Symposium*, Washington D.C., August 2001.

[22] R. Pang, V. Yegneswaran, P. Barford, v. Paxson, and L. Peterson. Characteristics of Internet Background Radiation. In *Proceedings of ACM Internet Measurement Conference*, Taormina, Italy, October 2004.

[23] M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of ACM Internet Measurement Conference*, November 2006.

[24] Sophos. Threat analysis. http://www.sophos.com/security/analysis, 2007.

[25] Symantec. Enterprise support knowledgebase. http://www.symantec.com, 2007.

[26] P. Szor. *The Art of Computer Virus Research and Defense*. Addison Wesley, 2005.

[27] J. Ullrich. Dshield. http://www.dshield.org, 2005.

[28] V. Yegneswaran, P. Barford, and J. Ullrich. Internet Intrusions: Global Characteristics and Prevalence. In *Proceedings of ACM SIGMETRICS*, San Diego, CA, June 2003.

[29] M. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1), 2001.