# Malware Biodiversity Using Static Analysis

Jeremy D. Seideman[1](✉), Bilal Khan[2], and Antonio Cesar Vargas[3]

[1] The Graduate Center, City University of New York, New York, USA
jseideman@gradcenter.cuny.edu

[2] Department of Mathematics and Computer Science, John Jay College, CUNY, New York, USA
bkhan@jjay.cuny.edu

[3] NacoLabs Consulting, LLC, New York, USA
cesar@nacolabs.com

**Abstract.** Malware is constantly changing and is released very rapidly, necessarily to remain effective in the changing computer landscape. Some malware files can be related to each other; studies that indicate that malware samples are similar often base that determination on common behavior or code. Given, then, that new malware is often developed based on existing malware, we can see that some code fragments, behavior, and techniques may be influencing more development than others. We propose a method by which we can determine the extent that previously released malware is influencing the development of new malware. Our method allows us to examine the way that malware changes over time, allowing us to look at trends in the changing malware landscape. This method, which involves a historical study of malware, can then be extended to investigate specific behaviors or code fragments. Our method shows that, with respect to the method in which we compared malware samples, over 64 % of malware samples that we analyzed are contributing to the biodiversity of the malware ecosystem and influencing new malware development.

## 1 Introduction

When studying *malware*, it is tempting to treat it as artificial life [27]; whether or not malware can actually be considered artificial life is a study on its own. When studying artificial life, though, concepts from biology are used to aid in measurement and analysis [34]. Our study of malware requires us to adopt some of those concepts from biology. We begin to build our method with these concepts. To start, we consider each malware sample as an *organism*.

Minor differences between organisms do not necessarily separate them into different species – two organisms can exhibit some differences and still be considered the same species. With malware, this is seen with *variants* – malware that is based on earlier malware with small changes. Looking at variants leads to a discussion of what actually constitutes relationships among malware [11]. As malware writers often reuse other code or infection techniques used in earlier malware, we can say that the earlier malware has an influence on later "offspring" malware, in

R. Doss et al. (Eds.): FNSS 2015, CCIS 523, pp. 139–155, 2015.
DOI: 10.1007/978-3-319-19210-9_10

this possible indirect fashion. As malware release rates are reported to be as high as 82,000 new malware samples per day [22], we can infer that there would be a lot of similarity between those new samples and existing ones, since malicious code is often based on existing code. Due to the way malware is released, we wanted to examine how malware samples relate to each other over time.

By definition, Malware interacts with, and has an effect upon, its "environment" – the infected system or systems upon which it operates [7,10]. In this study, that environment is the *ecosystem* in which our organisms exist. In biology, *biodiversity* can "refer to the number and size of populations in a community [20, p. 12]." Since that community can contain many different species, many ecologists prefer to examine communities based on their *species diversity*, which examines the number of species within the community and the number of members of each species [5, p. 1130]. The relationship of organisms with each other, with other organisms, and with other parts of the ecosystem is the organism's *ecological niche*, which represents not just a thing or a place, but the process as a whole including the things and the places [23, p. 72]. In our research, we see the side effects of malware, but they are not the same as the effects of living organisms. As a result, we will not see certain types of interaction between malware samples that we would see among biological organisms, such as competition between samples, despite the fact that this interaction is known to exist [27].

The ecosystem reflects the generalized state of computers as a whole, with the various operating systems, applications, and users and their associated vulnerabilities that can allow for malware infection. In effect, we are looking at malware through the lens of species diversity within an ecological niche that encompasses the collected computers in the world. Our definition of biodiversity, which will be explored below, therefore is more restrictive than the definition found in biology as we cannot reliably determine certain qualities about malware infection, such as the number of infected machines or the number of actual malware organisms.

Through a variety of methods, malware samples can be analyzed in order to see how they fit into this ecosystem. Malware analysis techniques are generally grouped into two types, static and dynamic, indicating how the samples are examined. Our study employs static analysis, as we are examining the malware binaries as they sit on disk. Our analysis method was partially inspired by previous studies, e.g. [16,17], which looked at fixed-width segments of binary code. Knowing that variants of malware are going to exhibit similar characteristics [32] allows us to group them together; while that particular study involved dynamic analysis of malware, the fact that malware behavior is a result of program code implies that variants of malware may have similar byte sequences.

The remainder of the paper is organized as follows: Sect. 2 discusses prior work related to our study, Sect. 3 explores our research method, Sect. 4 discusses our results, and Sect. 5 describes our findings and future directions for the work.

## 2 Prior Work

Often, malware is grouped together based on some sort of behavioral classification method; there are several ways to do this. Generally, the malware samples are analyzed dynamically, yielding a representation of their behavior, which can be used to determine the relationship between the samples. Lee and Mody classified malware by examining the behavior as a sequence of events that occur to the operating system, and then utilized several different methods to cluster similar malware together, creating families [18]. The goal of their research was to find a way to automate classification and use machine learning in order to produce more reliable clustering, in order to better defend against future malware.

Bailey, *et al.* devised a way to label malware that could provide a unified way to identify malware and respond to the shortcomings of many anti-virus systems, by profiling malware behavior on execution and examining the information overlap between those profiles [3]. Their results were quite promising, as they were able to consistently cluster malware into appropriate families. They were also able to show that anti-virus software often did not group malware together properly, and that different vendors would often group malware differently, which reduces the utility of anti-virus software in general. At the same time, Jacob, *et al.* promotes the use of behavioral detection to overcome shortcomings over other detection methods, while using several different grouping techniques to create families, but also does not discount the usefulness of static analysis to perform more detailed analysis [14].

Recently, the authors of [1] used Hidden Markov Models (HMMs) to classify malware. They state that similar malware will have similar behavioral profiles, and even though static methods are more efficient than dynamic methods, they do have limitations. Their method scores malware differences with an untrained HMM, clustered with $k$-means. Their method was able to easily differentiate between the malware families in their corpus, and they point out several improvements to their method that would also show promise, but conclude that their method would be useful in malware analysis. The authors of [33] have previously shown that the use of HMMs is useful in classifying similar malware even in the face of metamorphism, provided that malware samples are compared using a method that identifies code sequences that identify similar behavior, since even with metamorphic techniques, some of the code may remain the same and can be identified as such.

Seewald views the classification problem as one of machine learning, executing samples within a restricted execution environment, allowing for the recording of behavior [24]. He then tried several clustering algorithms to see if the clusters corresponded to the sample names, and to determine which clustering methods worked best. Bayer, *et al.* also examines several similar techniques; that study has shown positive results, demonstrating a highly efficient and accurate analysis and classification method of malware samples. They utilized several forms of behavior to create their profiles and create malware clusters, but also presented an efficient algorithm that avoids calculating all pairwise comparisons of samples in their data set [4].

All of this prior work shows the importance of classification by using behavior, and how classification is useful when performing analysis. None of them, however, explore the changing malware landscape over time and attempt to determine the influence of samples on the development of later samples. Our study will serve to examine the relationship between related samples in order to determine whether or not samples are creating "offspring" samples, whether they are variants or completely new types of malware.

One of the major influences for our study was the work by Karim, *et al.* which built phylogeny models based upon static analysis of malware samples, using both fixed-width code segments and permutations of those code segments. From there, they also employed some clustering techniques and were able to build a phylogenetic tree to compare detection rates of anti-virus programs in order to verify if their classification was correct [16]. Another major influence was the work by Kolter and Maloof, who also used fixed-width code segments, analyzing the frequencies that occurred in their data set in order to select those that were most relevant [17]. Their data set, which was made of benign and known malicious samples, and then samples from the wild, was then analyzed with several different methods to see how well their system could detect an unknown malware sample. Again, these studies did not take into account the temporal dimension, so the influence of samples over time could not be seen.

There has been other work that uses the concept of ancestor-descendant relationships between malware samples. One of particular interest is the work of Darmetko, *et al.*, which sought to use artificial intelligence techniques to determine relationships between malware samples in a lineage. Their method was able to correctly identify lineages to a high degree and correctly identify parent-child relationships in cases when they were known (which was a feature of their malware corpus). Their approach, however, did not take into account the temporal dimension, opting instead to use structures within the individual samples in order to identify parent-child relationships [8].

The work of Jang, *et al.* is also of interest as it also seeks to automatically determine software lineage. Their work represents the parent-child relationship with a straight line lineage and a directed acyclic graph, by employing a variety of methods – both static and dynamic – to determine the difference between samples. Their difference measurements included the use of $n$-grams of code. Their results also showed a high-degree of accuracy in identifying these relationships. They decided upon this approach instead of using phylogenetic trees because they felt that their method provided better metrics to judge the quality of their results [15]. As our method uses existing data about our malware corpus to build phylogenetic trees, we are able to use this existing data along with the relationships that we determine in order to perform our analysis.

## 3 Method

Our method of analysis is applied to a historical study of malware. We start by determining the relationship between those samples and then creating a data structure that represents that relationship. We can then use a time line to determine how samples contribute to the biodiversity of the malware ecosystem.

### 3.1 Malware Corpus

As we are engaged in a historical study of malware, we required a tagged and dated malware corpus for our analysis. We used the VX Heavens malware corpus [31] as our source. Using a custom email submission engine, we submitted the samples to VirusTotal [30], obtaining the labels that various anti-malware software would assign each sample. Taking a list of discovery dates from Symantec's Threat Explorer [28] we removed samples for which we had no discovery date, leaving us with a corpus, $\mathcal{M}$, of $N = 32,573$ samples. When $\mathcal{M}$ is sorted chronologically by discovery date, we refer to the $j^{th}$ malware sample, where $j \in \{1...N\}$, as $M_j$. The discovery date of sample $M_j$ is denoted by $Date(M_j)$. We refer to the set of malware discovered on date $d$ as $\mathcal{M}_d$.

### 3.2 Static Analysis

We analyzed our corpus with a static analysis technique. Our comparison method is based on the methods presented by Karim, *et al.* [16] and Kolter, *et al.* [17], where $n$-grams of bytes are extracted from each sample. However, instead of weighting the importance of an $n$-gram using TFxIDF or Information Gain as in those studies, we opted to use the $n$-gram distribution of bytes within the sample.

For every malware sample $M_j$, we computed the $n$-gram frequency probability distribution, $H_n(M_j) : P_n \rightarrow [0,1]$[1], where $P_n$ is the set of $n$-grams. We chose a value of $n = 4$ for our study, based partially on the previous work which showed the best results at that value of $n$. Furthermore, many x86 machine code instructions are four bytes or less (with the average being around two bytes [12]), increasing the chance that an extracted $n$-gram contains an entire instruction when $n = 4$. As the maximum size of an instruction is 15 bytes [13], any instructions requiring multiple sequences of four bytes will appear together in multiple samples if the same instruction is used. Using a larger $n$ would not give a better result as the $n$-grams would likely contain multiple instructions, reducing their usefulness. As we are scanning the entire file, the $n$-grams can also include non-instruction bytes. The distribution gives us the list of interleaved $n$-grams and the frequency of each.

For each pair of malware samples, $M_a$ and $M_b$, we computed the cosine similarity based on their $n$-gram distributions, $A = H_4(M_a)$ and $B = H_4(M_b)$. Using $A$ and $B$ as vectors, cosine similarity uses the angle $\theta$ between them as a measure of similarity [26]. Using cosine guarantees the similarity ranges from 0 to 1. To calculate the distance between two samples as represented by cosine similarity, we subtracted the similarity from 1:

$$D(M_a, M_b) = 1 - \cos(\theta) \tag{1}$$

[1] In other words, $H_n(M_j) : P_n \rightarrow \{y | 0 \leq y \leq 1, y \in \mathbb{R}\}$.

where $\cos(\theta)$ is calculated as the ratio of the **dot products** of the two vectors to the product of the **Euclidean lengths** of the vectors [9]:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{p \in P_4} (A_p \times B_p)}{\sqrt{\sum_{p \in P_4} (A_p)^2} \times \sqrt{\sum_{p \in P_4} (B_p)^2}} \tag{2}$$

in order to get the distance between two files, so that identical files (which would have $\theta = 0$ between them) have a distance of 0, while two completely different files have a distance of 1. For example, Trojan.Win32.Deltree.b and Virus.Boot.Havoc.l are very different as evidenced by a cosine difference of 0.983270. On the other hand, Virus.DOS.Spanska.1120.a and Virus.DOS.PS-MPC.748 are very similar, as their cosine difference is 0.000379.

We found that computing the distance between all possible pairs would be too time-consuming, so we pruned the number of required comparisons by only comparing samples to those that appeared earlier in time. We also noticed variability in the sizes of the samples and decided to reduce the number of comparisons by only comparing samples that are within a certain size threshold of each other, the rationale being that even if two samples are similar based on their $n$-gram distributions, they are probably not closely related if they are vastly different in size. We felt that by restricting the comparisons, we would be able to obtain more meaningful results. We selected a size threshold of 25 % for our dataset, and only compared samples that were within a 25 % binary size difference from each other.

It is important to note that all static analysis methods, including cosine similarity, are weak against obfuscation [3], where a sample's binary code can be changed through encryption and compression, and polymorphism and metamorphism [19,21], where samples alter their infectious payload in order to avoid matching known signatures of their infection, yielding malware which can be functionally similar but have different internal structures [33]. Polymorphic malware alters the routines used to decrypt the encrypted payload (which means that static techniques would work on the decrypted payload), while metamorphic malware alters the actual program code, through techniques like code substitution, garbage instruction insertion and subroutine permutation. Malware generation kits allow malware creators to perform these operations and others on malware code to create "new" malware that would be difficult to detect using static analysis but still maintain the same or similar behavior.

## 3.3 Tree Building

To demonstrate the relationship between pairs of malware, we constructed a phylogenetic tree. Samples were ordered by discovery date and added to the tree in order, so that we could see their ancestor-descendant relationship. The tree was built up incrementally by attaching each sample to the node in the tree with which it had the lowest distance. At time $i$ we constructed tree $T_i = (V_i, E_i)$

from tree $T_{i-1} = (V_{i-1}, E_{i-1})$, using malware samples $\mathcal{M}_i$ that appeared on that day[2]:

$$V_i = V_{i-1} \cup \mathcal{M}_i \tag{3}$$

and

$$E_i = E_{i-1} \cup \{(x, y) | x \in \mathcal{M}_i \wedge y \in V_{i-1} \wedge D(x, y) \text{ is minimal.}\} \tag{4}$$

such that $M_j$ connects to $V^a_{i-1}$, a vertex in tree $T_{i-1}$ for malware sample $M_a$, the closest sample that appeared before $M_j$. The tree generated is unique to the distance metric employed; the choice of a different distance metric will produce a different tree as the relationships between samples would be measured and expressed differently.

Our method does not take into account a threshold distance value, above which a new sample should become a new root node within a forest instead of being attached to an ancestor. Ongoing research is exploring different ways to identify better methods to attach nodes to the tree, in order to identify more accurate clusters of malware that help illustrate better ancestral relationships.

There were cases where there were samples that were larger than older samples in the corpus. In these cases, if they had a size difference larger than the 25 % threshold we used to prune the number of comparisons, we had no distance measurements and were unable connect them to an existing node in the tree. These nodes would then become the root of a new tree, thus turning our tree into a forest. For example, when Trojan.Win32.FormatAll.d appeared, there were no existing samples that were within the size difference threshold for it to attach to, so it formed the root of a new tree within the forest.

## 3.4 Stasis Criteria

If a node does not appear to spawn any offspring, it may be a "dead-end" in terms of development in the ecosystem. When a node is no longer contributing to the biodiversity of the ecosystem, we declare the node as "**in stasis**." The *stasis coefficient* at time $i$ is represented as $X_i$. We calculate $X_i$ by selecting a *stasis criterion*, which helps us define when a sample is to be placed in stasis. As a stasis criterion, we chose the *current mean* of the age difference between parent and child; our stasis coefficient was calculated as:

$$X_i = mean(Date(M_v) - Date(M_u) | (M_u, M_v) \in E_i) \tag{5}$$

Nodes in the tree or forest can have zero or more children. We defined $Child(M_j, i)$ as the set of children of $M_j$ at time $i$, directly connected to $M_j$ and appearing in the corpus after $M_j$. We then say that a node is *in stasis* if the difference between the sample's discovery date and the current date is greater than the stasis criterion, and the sample has not produced any offspring that

[2] Any nodes that appeared on the same day were added to the tree at the same time, since we could not determine the order in which the samples were discovered beyond the discovery date.

are *not* in stasis at time $i$. We check the node's children since, despite the age of the node, if it produced children that are not in stasis it is contributing to the overall biodiversity. At a given time $i$, the set of nodes that are in stasis, $S$, is defined as:

$$S_i = \{M_x \in V_i | i - Date(M_x) > X_i \wedge \forall M_y \in Child(M_x, i), i - Date(M_y) > X_i\} \tag{6}$$

Stasis differs from extinction in that stasis does not permanently remove the node from the ecosystem. In order to determine when a sample is no longer contributing to the biodiversity of the ecosystem, we use the stasis coefficient to calculate whether or not we should place a node in stasis at $i$. However, if a node in stasis at $i$ spawns a child at time $i + \delta i$, we would no longer consider the node to be in stasis at $i + \delta i$.

It is important to note that placing a node in stasis has no effect on that node's parents or children. If a node is placed in stasis, it is possible for that node's parent or child to remain out of stasis; if we were to remove nodes in stasis from the tree, placing a node in stasis would split the tree into multiple components.

### 3.5 Biodiversity Calculation

Our biodiversity calculation is based on our determination of which nodes in the forest are contributing to the biodiversity at each time $i$ on the time line. The set of nodes that are "alive" and contributing to the biodiversity of the ecosystem at each time $i$ are those nodes that are not in stasis at $i$:

$$Alive_i = V_i \setminus S_i \tag{7}$$

The biodiversity, $\beta$, at time $i$ is $0 \geq \beta_i \geq 1$ and is calculated as:

$$\beta_i = \frac{|Alive_i|}{|V_i|} \tag{8}$$

and is calculated at every time $i$ and can then be analyzed for a data set. We can also examine the rate of change of $\beta$ from one time point to another by taking the derivative of $\beta$. A $\beta$ of 1 indicates that all nodes are alive. A $\beta$ of 0 indicates that all nodes are in stasis. This can only happen if there is a time $i$ on the time line where no new nodes were discovered and there have been no new nodes discovered for a time period longer than the time period calculated by $X_i$. In other words, it is a date on the time line that occurs during a period of no development, after more than $X_i$ time has passed since the last sample was discovered.

## 4 Results

Our corpus consists 32,573 samples spread over 1,562 time points (where a time point is a date where malware samples appeared), ranging from 1 to 5,691 samples at each point, for an average of 20.85 samples appearing per time point.

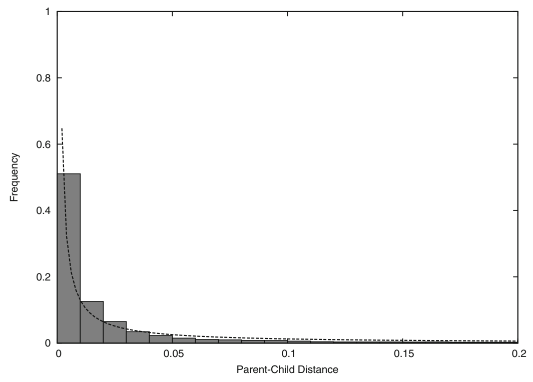

**Fig. 1.** Histogram of parent-child distances, using $D(M_a, M_b)$.

Our analysis was performed on the COS25 dataset – the comparison of samples using cosine similarity when the binary sizes were within 25 % of each other.

We calculated the pair-wise distances between samples and built the phylogenetic trees at each time $i$ to calculate $X_i$ and $\beta_i$. Overall, the average parent-child age difference was 806 days, with a standard deviation of 658 days. This means that samples can be more closely related to older samples than to more recent ones, suggesting the existence of infection techniques and code segments that are reused for a very long time. The average parent-child distance was 0.1149 with a standard deviation of 0.2483, indicating that child samples tend to be very similar to their parents. While there were some children that had a larger distance to their parent, this value indicates this was not the norm.

The frequencies of parent-child distances as determined by our cosine similarity distance measurement decay as expected by a power law. Figure 1 shows the frequency histogram, fit to $y = ax^b$ when $a = 0.0013$ and $b = -1.0066$.

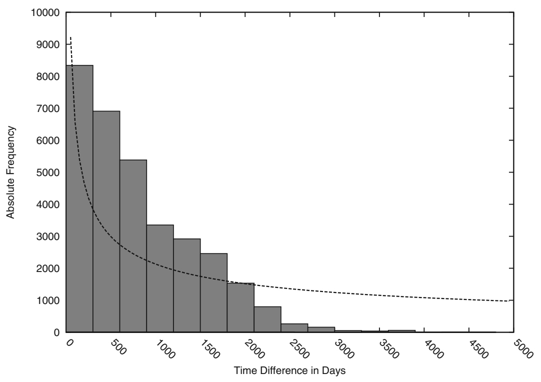

**Fig. 2.** Histogram of parent-child age differences.

We see that, while there are sample pairs that exhibit a larger parent-child distance, the majority of the parent-child distances are small. In the figure, we limit the x-axis to a value of 0.2 as only a tiny fraction of the parent-child distances were greater than 0.2.

Figure 2 is a histogram of the frequencies of parent-child age differences, including the best-fit curve $y = ak^b$, when $a = 62539.2$ and $b = -0.4892$. The distribution deviates from the best-fit curve but still demonstrates a decay. We saw more of the smaller age differences, with fewer large age differences. While there were occasional parent-child age differences that were exceedingly large, the fact that the majority of them were small suggests that new malware tends to be based on more recent malware that is in turn based on older samples. However, we know that the average parent-child age difference is large, which we see in the deviation of the distribution from the expected values. The majority of the parent-child age differences occur at a greater frequency than actually expected, skewing the average towards a higher number. While a node is more likely to attach to a more recent sample than an ancient one, we see a high number of instances of children attaching to an older parent than a more recent one.

Over the entire time line,the average $\beta$ was 0.6412, with a standard deviation of 0.2041. In other words, 64.12 % of the samples were contributing to the biodiversity of the ecosystem. A high average $\beta$ would mean the majority of the samples present at any time were spawning offspring. A lower average $\beta$ does not necessarily mean the opposite – it could mean is that our malware ecosystem is similar to a natural one, and that samples that go into stasis do not return to the ecosystem. Rather, their offspring lead to future progeny – ancestral samples are eventually removed but their descendants remain to spawn new samples. Since our measure of biodiversity takes into account all of the samples that have *ever* appeared, we see that stasis limits the overall biodiversity, while still allowing for new lineages of malware samples.

## 4.1 Malware Offspring and Families

The phylogenetic tree tells us a variety of things about our data set. We can determine families, calculate node procreation rates and separate out those nodes which can be considered a "dead-end" in terms of malware development. For example, Virus.DOS.SillyC.217.d was released in 1993 and never spawned a child in the data set that we could link to, based on our method. We can then say that the code used in that sample did not appear to lead to further development of later samples.

We found that 22.9 % (7,463) of the nodes in our tree had child nodes, meaning 77.1 % of the nodes in the tree were leaf nodes. This does not mean that they were all dead-ends, though, only that at that time we did not have a child to attach to them. On the average, nodes with children had an out-degree of 4.33, with a standard deviation of 13.35.

The number of offspring that a malware sample produced appears to obey a power law distribution, as seen in Fig. 3, a histogram of the frequencies of the out-degrees of the nodes in our tree fit to $y = ax^b$ where $a = 0.2185$ and

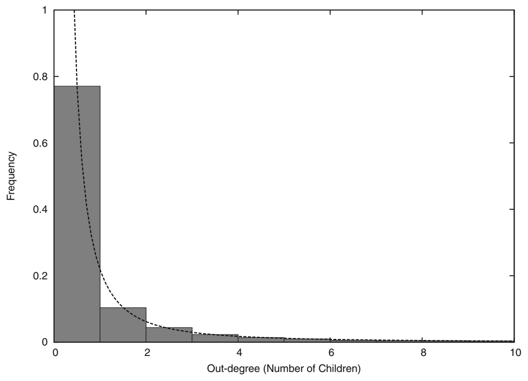

**Fig. 3.** Histogram of number of children for COS25.

$b = -1.8187$. Nodes with out-degrees greater than 10 represented a very small percentage of the frequencies and are absent from the histogram. As show in the figure, the majority of the nodes have an out-degree near the mean, which was expected, despite the low average out-degree and high standard deviation. There is a large amount of variability in the out-degrees present in our tree.

Of the nodes that produced children, there were several that produced a large number, although these samples were clearly outliers from what would be expected based on the frequency distribution of out-degrees. Table 1 shows the samples that have generated the greatest number of children, based on having the highest out-degree. This does not necessarily indicate that these samples are extremely long-lived; all of those children *could* have appeared soon after the parent. It could also mean that these samples were the progenitors of large families of malware, or that these samples were responsible for large branches of their respective families, leading to many descendants. What we do know is that a large number of children were spawned by this sample, something which can be explored further.

We found 276 root nodes in the forest – nodes that have an in-degree of 0, of which 111 of those are singletons (with no children). Singletons are dead-ends as they are not spawning anything. We cannot reliably connect them to other families

**Table 1.** Top 5 child-creating samples of COS25.

| Sample | Discovery date | Children |
|---|---|---|
| Trojan-Dropper.Win32.Small.mj | 20020404 | 508 |
| Virus.DOS.PS-MPC.186 | 19920714 | 285 |
| IM-Worm.Win32.Lewor.c | 19971208 | 282 |
| Trojan-Spy.Win32.Small.gc | 20020504 | 267 |
| Trojan-Downloader.Win32.Small.bmd | 20020404 | 236 |

at this time because they were not compared to other samples due to size difference. The remaining 165 root nodes are in subtrees that represent malware families – groups of closely related malware that share common traits and ancestry.

## 4.2 Biodiversity Over Time

The biodiversity curve in Fig. 4 shows how the biodiversity in the ecosystem changed over time. The biodiversity is marked by periods of rapid increase and decrease. When samples appear in large numbers, we see the curve spike due to the arriving samples, and then remain either relatively flat before rapidly decreasing (due to stasis) or gradually decreasing (due to low malware development rates) as nodes are removed, changing the ratio of living nodes to total nodes. There are time when the biodiversity increases gradually, as the rate of new sample arrival exceeds the rate of stasis for old samples. This could be due to those samples attaching to a small number of existing nodes (leading to a smaller increase), or a larger number of existing nodes (leading to larger increase, as those older nodes would now remain "alive"). However, the second curve in the figure shows the total samples as they appear along the time line, and shows that while the biodiversity can correlate with new arrivals, increasing biodiversity and rapid malware development are not necessarily connected. We can see this between 2006–2008, where the number of samples is increasing, albeit slowly, while the overall biodiversity falls. This could be the result of those samples attaching to recent samples (placing older samples into stasis), while also causing the average parent-child age difference to fall, thus forcing even more samples into stasis.

Taking the derivative of the biodiversity curve shows us that generally, the biodiversity exhibits small percentages changes between days. This is shown in Fig. 5, which shows the daily changes in biodiversity, along with the number of new samples that appear each day. The small changes are to be expected as new samples were appearing all the time. However, we notice that there are occasional peaks and valleys in the graph, with the three largest of each

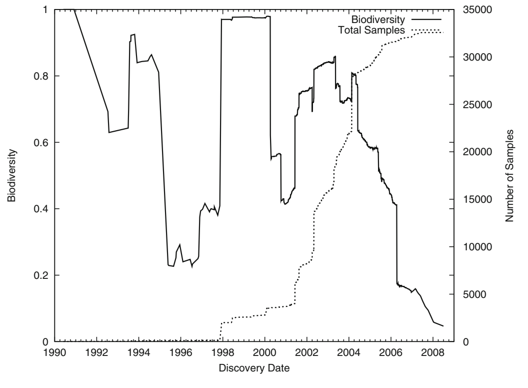

**Fig. 4.** Biodiversity curve of COS25 for current mean stasis criterion.

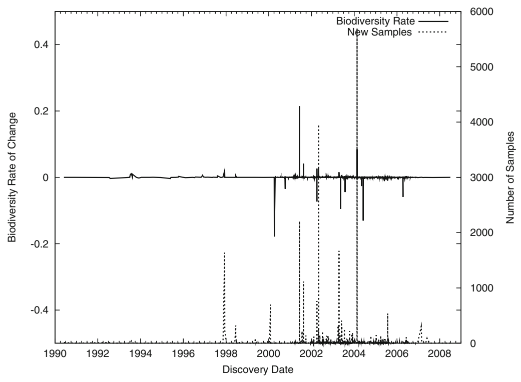

**Fig. 5.** Derivative biodiversity curve of COS25 for current mean stasis criterion.

summarized in Table 2. Peaks occur when a large number of samples appear due to the new nodes increasing the number of living nodes and a larger fraction of nodes are creating offspring. The valleys tend to occur towards the end of a period of low malware development, where more nodes are going into stasis than are appearing in the ecosystem.

In the peak and valley table, the "New" column refers to nodes that appeared that date. The "Alive" column includes both these new nodes and existing nodes that have not gone into stasis. Similarly, the "Total" column refers to all of the nodes that exist in the tree as of that day, both alive and in stasis.

We found there is no necessary correlation between and peak and a valley in our study; they do not appear in definite pairs. This is because the samples are increasing at a gradual rate punctuated by the occasional burst, but there is no regularity in the timing of those bursts. Figure 6 shows the total samples as it increases over our time line, while also showing the number of samples that appear at each time point. We suspect that the peaks in the graph correspond to

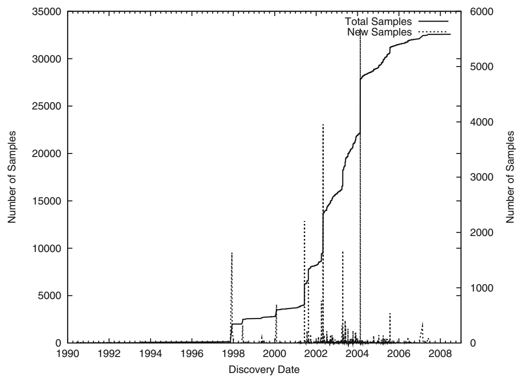

**Fig. 6.** Total samples and arriving samples, by discovery date.

**Table 2.** Top three peaks and valleys in COS25 derivative curve for current mean stasis criterion.

| Discovery date | Type | Value | New | Alive | Total |
|---|---|---|---|---|---|
| 20000406 | Valley | −0.1786 | 1 | 2197 | 3535 |
| 20010608 | Peak | 0.2143 | 2206 | 4230 | 6230 |
| 20010817 | Peak | 0.0411 | 1116 | 5798 | 7763 |
| 20030514 | Valley | −0.0953 | 32 | 14215 | 18636 |
| 20040219 | Peak | 0.0845 | 5691 | 22525 | 27841 |
| 20040603 | Valley | −0.1305 | 1 | 17995 | 28353 |

some milestone in operating systems or applications development, as generally there is an increase in malware upon the discovery of an exploitable vulnerability, which would then drive the release of a patch to resolve it [2,6].

### 4.3 Effect of Malware Packers

A common technique by which malware evades static analysis methods is through the use of packers, such as UPX [29], which may make a sample appear similar to other samples that are packed using similar methods. In our corpus, we found 11,378 instances of packed malware – approximately 34.9 % of the samples. Examining the packed samples we found that the average distance between a packed sample and its parent was 0.0414, with a standard deviation of 0.1012. This smaller average and lower amount of variation indicates that our results are affected by the use of packers; packed malware tends to be more closely related according to our distance metric as they all exhibit a similar characteristic – the use of an unpacking routine to execute.

## 5 Conclusions

We have presented a method by which we can measure the biodiversity exhibited within an ecosystem. Our method determines whether or not a malware sample is exhibiting an effect on the overall ecosystem as measured by whether or not it has apparently spawned a variant or other descendant. We showed that generally, this measurement of biodiversity does not change much from one day to the next, but over the entire time line, we can see peaks and troughs, which can relate to the inconsistent development and release of malware. Our analysis shows rapid malware development exhibited in our corpus with a generally high level of biodiversity, which could indicate that malware samples are being written quickly, building on previously used techniques that remained successful over time. It is definitely worth examining how the observed biodiversity of the malware ecosystem compares to the overall state of software evolution; a similar study of known benign software would inform us how malware development relates to general software development.

This method can be adapted to examine and identify prevalent malware behavior and code techniques. Instead of looking at the biodiversity of the ecosystem as a whole, research can be done by which either a specific sample's offspring are tracked, or whether a specific feature remains prevalent and influences other development. Identifying prevalent behavior and code techniques can help defend against future infection because they can be used to aid in the differentiation between malicious and benign samples. In order to perform this kind of analysis, the overall method may require adjustment, but specifically the similarity measure between samples must be changed.

An important future direction for the use of this method is to perform a biodiversity analysis of our corpus using dynamic analysis techniques in order to examine samples at run-time. Statically, samples may appear similar but at execution time, those samples may appear very different. For packed or encrypted malware, examining the sample's payload at execution may show a large difference between samples that seemed similar from a static perspective. By examining samples at execution time, we can also compare the behavior that the samples exhibit, and determine the similarity between them based on the effects on the infected system. As a result, we will be able to construct a different phylogeny tree based on a different distance measurement, and examine how the resulting biodiversity calculation changes with respect to the method by which the samples are compared.

It is also important to note that a future study with this method can examine malware ecological niches in a more isolated fashion – while here we looked at the malware ecosystem as a whole, it is possible to restrict the environment such that we can look at certain factors that limit malware growth and propagation, namely OS conditions (such as OS version, sub-version, patch levels) and application vulnerabilities, both of which have been shown to have an effect on malware rates within ecosystems [25].

Finally, we can also examine the effect that changes in the data set can have on the calculations, whether we adjust the size threshold used in our comparison between samples, or take random samples of the data set and perform the same biodiversity calculations. As it is possible for a new feature to increase the size difference between a parent and child, we would like to see how new strains of malware differentiate from their ancestors, in order to better identify when we can consider a new version of malware as a new "species" in the ecosystem.

## References

1. Annachhatre, C., Austin, T., Stamp, M.: Hidden markov models for malware classification. J. Comput. Virol. Hacking Tech., 1–15 (2014). http://dx.doi.org/10.1007/s11416-014-0215-x
2. Arora, A., Krishnan, R., Telang, R., Yang, Y.: Impact of vulnerability disclosure and patch availability - an empirical analysis. In. Third Workshop on the Economics of Information Security (2004). http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.81.9350

3. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 178–197. Springer, Heidelberg (2007). http://dx.doi.org/10.1007/978-3-540-74320-0_10
4. Bayer, U., Comparetti, P.M., Hlauschek, C., Krügel, C., Kirda, E.: Scalable, behavior-based malware clustering. In: Proceedings of NDSS 2009 (2009). http://www.isoc.org/isoc/conferences/ndss/09/pdf/11.pdf
5. Campbell, N.A.: Biology, 4th edn. The Benjamin/Cummings Publishing Company Inc., New York (1996)
6. Cencini, A., Yu, K., Chan, T.: Software Vulnerabilities: Full-, Responsible-, and Non-Disclosure (2005). http://www.cs.washington.edu/education/courses/csep590/05au/whitepaper_turnin/software_vulnerabilities_by_cencini_yu_chan.pdf
7. Cohen, F.: Computer virus: theory and experiments. Comput. Secur. **6**, 22–35 (1987). http://www.cs.washington.edu/education/courses/csep590/05au/whitepaper_turnin/software_vulnerabilities_by_cencini_yu_chan.pdf
8. Darmetko, C., Jilcott, S., Everett, J.: Inferring accurate histories of malware evolution from structural evidence. In: The Twenty-Sixth International FLAIRS Conference (2013)
9. Dot Products (2009). http://nlp.stanford.edu/IR-book/html/htmledition/dot-products-1.html
10. Filiol, E., Helenius, M., Zanero, S.: Open problems in computer virology. J. Comput. Virol. **1**(3–4), 55–66 (2006). http://dx.doi.org/10.1007/s11416-005-0008-3
11. Gheorghescu, M.: An automated virus classification system. In: Virus Bulletin Conference, pp. 294–300, Oct 2005
12. Ibrahim, A., Abdelhalim, M.B., Hussein, H., Fahmy, A.: Analysis of x86 instruction set usage for Windows 7 applications. In: 2010 2nd International Conference on Computer Technology and Development (ICCTD), pp. 511–516 (2010)
13. Intel Corporation: Intel® 64 and IA-32 Architectures Software Developer Manuals (2013). http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html/
14. Jacob, G., Debar, H., Filiol, E.: Behavioral detection of malware: from a survey towards an established taxonomy. J. Comput. Virol. **4**(3), 251–266 (2008)
15. Jang, J., Woo, M., Brumley, D.: Towards automatic software lineage inference. In: Proceedings of the 22nd USENIX Conference on Security, pp. 81–96. USENIX Association (2013)
16. Karim, M.E., Walenstein, A., Lakhotia, A., Parida, L.: Malware phylogeny generation using permutations of code. J. Comput. Virol. **1**(1–2), 13–23 (2005)
17. Kolter, J.Z., Maloof, M.A.: Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res. **7**, 2721–2744 (2006). http://www.jmlr.org/papers/v7/kolter06a.html
18. Lee, T., Mody, J.J.: Behavioral classification. In: Proceedings of EICAR 2006, pp. 1–17, May 2006
19. Li, Z., Sanghi, M., Chen, Y., Kao, M.Y., Chavez, B.: Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy. pp. 32–47 (2006). http://doi.ieeecomputersociety.org/10.1109/SP.2006.18
20. Mader, S.S.: Inquiry into Life (Customized for Brooklyn College), 9th edn. The McGraw-Hill Companies Inc., Primis Custom Publishing, New York (1999)

21. Newsome, J., Karp, B., Song, D.X.: Polygraph: automatically generating signatures for polymorphic worms. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, pp. 226–241 (2005). http://doi.ieeecomputersociety.org/10.1109/SP.2005.15
22. Annual Report Panda Labs - 2013 Summary (2013). http://press.pandasecurity.com/wp-content/uploads/2010/05/PandaLabs-Annual-Report_2013.pdf
23. Salthe, S.N.: Evolutionary Biology. Holt, Rinehart and Winston Inc., New York (1972)
24. Seewald, A.K.: Towards autmating malware classification and characterization. In: Proceedings of Sicherheit 2008, pp. 291–302 (2008). http://alex.seewald.at/files/2008-01.pdf
25. Seideman, J., Khan, B., Ben Brahim, G.: Determining vulnerability resolution time by examining malware proliferation rates. In: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1678–1682 (2013)
26. Singhal, A.: Modern information retrieval: a brief overview. IEEE Data Eng. Bull. **24**(4), 35–43 (2001)
27. Spafford, E.H.: Computer viruses as artificial life. Artif. Life **1**(3), 249–265 (1994)
28. Threat explorer - spyware and adware, dialers, hack tools, hoaxes and other risks (2012). http://www.symantec.com/security_response/threatexplorer/
29. UPX: the Ultimate Packer for eXecutables - Homepage (2010). http://upx.sourceforge.net/
30. VirusTotal (2008). http://www.virustotal.com
31. VX heavens (2010). http://vxheaven.org/
32. Wagener, G., State, R., Dulaunoy, A.: Malware behaviour analysis. J. Comput. Virol. **4**(4), 279–287 (2008)
33. Wong, W., Stamp, M.: Hunting for metamorphic engines. J. Comput. Virol. **2**(3), 211–229 (2006). http://dx.doi.org/10.1007/s11416-006-0028-7
34. Woodberry, O.G., Korb, K.B., Nicholson, A.E.: Testing punctuated equilibrium theory using evolutionary activity statistics. In: Korb, K., Randall, M., Hendtlass, T. (eds.) ACAL 2009. LNCS, vol. 5865, pp. 86–95. Springer, Heidelberg (2009)