Identifying Malware Genera using the Jensen-Shannon Distance Between System Call Traces

Jeremy D. Seideman
The Graduate School and University Center
City University of New York
New York, USA
Email: jseideman@gc.cuny.edu

Bilal Khan
Dept. of Math & Comp. Science
John Jay College, CUNY
New York, USA
Email: bkhan@jjay.cuny.edu

Antonio Cesar Vargas
NacoLabs Consulting, LLC
New York, USA
Email: cesar@nacolabs.com

Abstract

The study of malware often involves some form of grouping or clustering in order to indicate malware samples that are closely related. There are many ways that this can be performed, depending on the type of data that is recorded to represent the malware and the eventual goal of the grouping. While the concept of a malware family has been explored in depth, we introduce the concept of the ***malware genus****, a grouping of malware that consists of very closely related samples determined by the relationships between samples within the malware population. Determining the boundaries of the malware genus is dependent upon the way that the malware samples are compared and the overall relationship between samples, with special attention paid to the parent-child relationship. Biologists have several criteria that are used to judge the usefulness of a genus when creating a taxonomy of organisms; we sought to design a classification that would be as useful in the world of malware research as it is in biology. We present two case studies in which we analyze a set of malware, using the Jensen-Shannon Distance between system call traces to measure distance between samples. The case studies show the genera that we create adhere to all of the criteria used when creating taxa of biological organisms.*

1 Introduction

Classification methods are extremely useful when studying malware. By finding ways to classify and group different malware samples, it is possible to compare malware in a useful way, such as by limiting research to groups that exhibit certain properties or features. Classifying malware samples also aids in malware defense, removal and analysis as similar malware may have similar weaknesses, structures and effects, all of which may be indicated through better detection methods [14, 23].

The classification of malware has been the subject of many research papers, such as [1, 2, 9, 14, 22], all of which approach the problem in a slightly different manner. The obvious problem with approaching classification in a different manner is that each study determines its own classification scheme, which leads to inconsistency among classification – a cluster or family determined by one study might not agree with those determined by another. That problem is not isolated to research studies; the anti-virus industry previously dealt with the same issue, leading to the 1991 New Virus Naming Convention (also called the CARO Convention). The CARO Convention attempted to standardize virus and malware names, based on family, group, and variant names [21]. This approach was successful in a limited fashion, as many vendors still use their own naming schemes.

All classification, however, is based on the assumption that there are common elements among malware samples. As the rate of malware release has been reported to be as high as 82,000 new malware samples per day [19], it follows that this would be the case. In the world of biology, it is extremely common to group organisms together based on "morphology, physiology, ecology and genetics" [4]. These groupings are called **taxa**, indicating a grouping at one of several levels of organization [3, p. 471]. The most common way that many organisms are identified is the **bino-**

mial name originated by Linnaeus [3, p. 469], made up of a **genus** (plural **genera**) and a **species**.

If one considers the name of a malware sample as a species, and a variant name as a subspecies, the concept of a family may be too broad to accurately represent some malware relationships. For example, the malware sample Email-Worm.Win32.NetSky.j is the "j" variant of the NetSky worm, which we see affects Win32 systems, and is a type of email worm. Based on this model, we see that Email-Worm.Win32.Skybag.a is another email worm that affects Win32 systems, but we cannot determine anything about the relationship between NetSky and Skybag. It is possible that NetSky and Skybag are actually very similar, but are not considered variants of the same basic code. We propose the concept of the **malware genus**, a way of grouping malware samples that are closely related, despite not being variants of the same piece of malware. The genus should group similar malware but, at the same time, ensure that those groups consist only of close relatives.

In general, a genus is a group that has common features of some kind. Biologically, however, there are additional requirements for a genus to be considered useful [8]:

- A genus must exhibit **monophyly** – the members are descended from a single common ancestor
- A genus must have **reasonable compactness** – it is not expanded more than it needs to be
- A genus must show **distinctness** – there is some quality that is unique to the genus

Our method identifies malware genera that satisfy all three of these requirements. By developing this method, we have created a technique for malware analysts to use to identify the set of closely-related malware in an overall malware landscape. This set will be of a relatively small size, and will be based upon an observable characteristic that all of the samples have in common, which will all be traceable back to a single ancestral sample within the corpus.

The remainder of this paper is organized as follows: Section 2 examines prior research in this field related to our study, Section 3 describes the methodology employed for our analysis, Section 4 provides a discussion of our results including two case studies and Section 5 summarizes our findings and provides future direction for research.

2 Prior Work

There are many ways to identify similar pieces of malware. While there are methods based on static analysis (reading the malware sample as it resides on disk) and performing statistical analysis on that analysis, such as Karim, *et al.* [11] and Kolter and Maloof [13], those methods are victim to the weaknesses of static analysis [23]. Dynamic analysis provides a better way to examine a sample's effects. However, the statistical methods employed were useful in creating our method, and served as a good example of how to do large-scale comparison of malware samples. While behavioral analysis is often cited as a way to overcome those weaknesses in static analysis, Jacob, *et al.* cite that static analysis techniques are very useful when performing a detailed malware analysis [10].

Gheorghescu, in [9], proposes a method that seems to bridge the gap between static and dynamic analyses, in that the method presented relies on the use of Control Flow Graphs (CFGs). CFGs are derived from the sample itself, usually based upon the binary or the source code. However, CFGs, which are constructed in a static form, better illustrate the malware behavior in that pieces of the graph are grouped together into blocks of code that work as a unit. These blocks could map to a system call, and thus allow us to examine program flow. Mutz, *et al.* also used system calls [16]. Instead of looking for malware, though, the authors were interested in intrusion detection systems. They acknowledge that system call traces provide a wealth of information for detection, but generally analysis ignores the arguments to those calls. Their system created profiles that include the arguments, which allowed for better detection performance. Their work could inform future work on malware analysis involving system calls.

Since we needed to perform a dynamic analysis of the samples, we look at the various methods that have been explored in the past. Bailey, *et al.* presented a behavior-based clustering method for malware [1]. Their method was very influential as it combined the dynamic analysis techniques along with categorization. Their goal was to overcome some of the shortcomings in malware labeling techniques. They even examine shared information between samples. However, their method is not based on the direct observation of the malware samples, but by first creating a behavior profile that represents the system side effects during execution. Behavior-based analysis is used for malware detection as well, as shown by Gao, *et al.*, who designed a method to measure differences in behavior, taking into account the differences in OS platforms, and differences in software process [7]. Through this, the authors were able to identify when a process was being altered based upon deviation from expected behavior when presented with a known input.

Lee and Mody used a dynamic approach as well, based upon the interaction of the malware sample with the operating system [14]. They took a machine-learning approach to the classification, employing several methods to derive malware families. Their research, however, specifically avoided using API calls and instead looked at operating system events as a sequence. Seewald also employed machine learning to classify malware, but also employed a restricted execution environment [22]. Using that environment, the author was able to create a behavioral profile and identify characteristics, which were then classified by several different methods. Bayer, *et al.* employed a similar approach by testing out several related techniques, and were able to create a highly scalable approach that worked very efficiently [2]. They did note that the behavioral profiles are dependent on the behavior that is exhibited during the actual execution of a sample, which could change based on the parameters of the environment in which it runs.

The research reported by Park, *et al.* takes a similar approach to ours, in that it executes malware samples in a restricted environment and captures the system call traces generated [20]. Their approach, however, requires the comparison to a known malware sample, so it is effectively a dynamic signature. They create a graph of the system call traces and use graph comparison algorithms to measure the difference between them. Kinable and Kostakis take a graph of function calls and apply various metrics to determine the similarity between them [12]. They then apply clustering algorithms to show that it is possible to differentiate malware samples and cluster them appropriately. Their approach, though, is ignorant to when the samples are discovered or the order in which samples appeared. Our method, on the other hand, takes into account the actual time that the malware sample was discovered, and does not require a graph comparison.

3 Method

We applied our analysis method to a dated corpus of known malware. Our study is concerned with the relationship between malware samples in a historical context, so we represented the relationship between malware samples taking into account the temporal difference between them.

3.1 Malware Corpus

In order to perform a historical study, we required a set of known malware samples along with their discovery dates in order to construct a time line. We selected the corpus available from VX Heavens [26] due to the large number of samples and variability of sample type present. The samples were submitted to VirusTotal [25], a service that subjects a sample to a collection of anti-virus software and returns the results from multiple scans. This allowed us to map the files in the corpus to their names as designated by several anti-virus vendors. Using Threat Explorer [24], a tool created by Symantec, we were able to record discovery dates for many samples. Any sample for which there was no discovery date was discarded. This gave us a corpus, $\mathcal{M}$ of $N = 32,573$ malware samples. We sorted the samples chronologically by the discovery date to create a time line. We refer to the set of malware that appears on a given day d on the time line as $\mathcal{M}_d$. When all the malware samples in $\mathcal{M}$ are placed in some order, we refer to the j^{th} sample as M_j where $j \in 1...N$ is the index of that order.

3.2 Dynamic Analysis

Our samples were analyzed with dynamic analysis techniques, using the Norman Sandbox [17], a commercial application that can execute a sample in a restricted environment. This application allowed us to execute malware samples, examine the effect upon the system and record the effects, without causing any harm to the host computer. We used the sandbox execution to capture the system call traces of the malware samples. As we were looking at the interaction between the malware sample and the infected system, we were interested in the functions that the sample called in order to actually perform the intended behavior. There were some samples that did not generate any observable behavior, such as boot-sector viruses or malware that targets vulnerabilities that have since been patched in the sandboxed environment. Any samples for which we did not collect a system call trace were removed from our corpus, so our corpus was reduced to $N = 26,299$ samples.

We then reduced the system call traces into a probability distribution, $\xi(M_j) : S_t \to [0,1]$, of system calls within the execution, where S_t is the set of system calls present in trace. We examined the system call traces based on the supposition that similar malware samples will interact in a similar manner with the operating system, and should therefore have a similar distribution of system calls within the system call trace. For each pair of malware samples, M_p and M_q, we calculated the Jensen-Shannon Distance between them using $P = \xi(M_p)$ and $Q = \xi(M_q)$, so $D(M_p, M_q) = JSD(P\|Q)$.

3.3 Jensen-Shannon Distance

We chose to use the Jensen-Shannon Distance, which is a metric version of the Jensen-Shannon Divergence, to measure the difference between malware samples. The Jensen-Shannon Divergence measures the amount of mutual information between two probability distributions. We chose it because it gives us the advantage of dynamic analysis (namely, being able to observe and quantify system interaction) while also allowing us to use the statistical analysis methods suggested by works such as [11] and [13].

The Jensen-Shannon Divergence of two probability distributes, P and Q, is denoted as $JSD(P\|Q)$, and is based on the Kullback-Liebler Divergence [6, 15, 18], $KL(P\|Q)$:

$$KL(P\|Q) = \sum_i P(i)log\left(\tfrac{P(i)}{Q(i)}\right)$$

By calculating the *mixture distribution*, $R = \frac{1}{2}(P + Q)$, we calculate the Jensen-Shannon Divergence by taking the Kullback-Liebler Divergence of each distribution with the mixture distribution:

$$JSD(P \| Q) = \frac{1}{2} KL(P \| R) + \frac{1}{2} KL(Q \| R)$$
$$= \frac{1}{2} \sum_i P(i) \left(log \tfrac{P(i)}{R(i)} \right) + \frac{1}{2} \sum_i Q(i) \left(log \tfrac{Q(i)}{R(i)} \right)$$

However, the Kullback-Liebler distance is only defined for probability distributions where both P and Q total to 1, and $P(i) = 0$ implies $Q(i) = 0$. When looking at distributions such as system call traces, there will be cases where system calls are made in one sample that are not made in the other. In order to take this into account, we used a technique called *absolute discounting* in which we apply a *smoothing factor*, ε, to the formula, to prevent the infinite divergence that would occur without it [5]. This changes the definitions of $P(i)$ and $Q(i)$. With $U = P_0 \cup Q_0$, and P_0, Q_0 representing the original probability distributions, let

$$disc_p = \varepsilon \left(\tfrac{|U| - |P_0|}{|P_0|} \right)$$

be the *discount* such that

$$P(i) = \begin{cases} P_0(i) - disc_p & \text{if } P_0(i) > 0 \\ \epsilon & \text{otherwise} \end{cases}$$

with the definitions of $disc_q$ and $Q(i)$ following based on Q and Q_0 respectively.

We set $\varepsilon = 0.0000001$, a sufficiently small value to prevent large changes in the divergence values of samples, while still allowing us to calculate the divergence. In order to set the lower and upper bound of the Jensen-Shannon Divergence to 0 and 1, respectively, we use the base-2 logarithm in our calculation [15]:

$$JSD(P \| Q) = \frac{1}{2} \sum_i P(i) log_2 \left(\tfrac{P(i)}{R(i)} \right) + \frac{1}{2} \sum_i Q(i) log_2 \left(\tfrac{Q(i)}{R(i)} \right)$$

The Jensen-Shannon Divergence itself is not a metric, but the square root of it is [6]. Therefore, The Jensen-Shannon Distance metric is:

$$D(M_p, M_q) = \sqrt{JSD(P \| Q)}$$

3.4 Phylogenetic Tree

In order to demonstrate the parent-child relationship between the malware samples based on our distance metric, we constructed a phylogenetic tree. We were able to do this by taking advantage of the fact that we had ordered the malware samples by discovery date in order to add nodes to the tree in order.

We built the tree incrementally, by attaching each sample to an existing node in the tree with which it had the lowest distance. Using the discovery dates, we created an ordered list of malware samples, and iterated over that list. At point j, tree $T_j = (V_j, E_j)$ was built from $T_{j-1} = (V_{j-1}, E_{j-1})$, using malware sample or samples that appeared at j (denoted as $\mathcal{M}_j$) by adding $\mathcal{M}_j$ to the set of vertices using the smallest edge possible, based on inter-sample distance:

$$V_j = V_{j-1} \cup \mathcal{M}_j$$

and

$$E_j = E_{j-1} \cup \{(x, y) | x \in \mathcal{M}_j \wedge y \in V_{j-1} \wedge D(x, y) \text{ is minimal.}\}$$

3.5 Edge Pruning

In order to determine the boundaries of the various genera, we examined the edges in our tree, in order to remove those that were too large. We wanted to identify genera of closely related malware, so we needed to derive a method by which we could create a threshold value, τ, of distance between samples. A edge whose weight was greater than τ would indicate that the relationship between those samples was not close enough to include them in the same genus.

Our calculation of τ was based on statistical information present in the tree, namely the mean distance between parent-child samples, and the standard deviation from that mean. By varying a factor, φ, we were able fine-tune the threshold value:

$$\tau = \overline{D(x, y)} \pm \varphi \sigma_{D(x,y)}$$

for values of $0 \leq \tau \leq 1$.

Next, we pruned edges from tree $T = (V, E)$ that had a weight greater than τ, creating $T' = (V, E')$. While the vertices in the tree remained the same, the edges were updated as:

$$E' = E \setminus \{E_k | w(E_k) > \tau\}$$

For each tree T', we calculated the number of components generated after removing the edges, and the number of components consisting of a single vertex with no attached edges. The number of genera identified in that tree is the difference between the number of components and the number of singletons. As τ varies from 0 to 1, the number of genera present increases to a peak, and then decreases. An example of this is graphed in Figure 1. The value at which is peaks is the τ that, for the dataset, difference metric and tree, identifies the number of genera present in our sample. At values of τ above that peak, the tree divides into fewer components, but the edges between the nodes may represent a relationship that is not as close. At values of τ below the peak, edges are removed because the threshold value is very low, so even closely related samples are then disconnected from each other, leading to more singletons. At the peak, though, we have a balance between the linking of closely related samples, and the separation of more distantly-related samples.

It is important to note, however, that singletons are not isolated – they can be considered genera unto themselves. We do not consider them in our study because we wish to identify genera that have multiple members. Genera that have single members do not help us examine parent-child relationships among malware.

4 Results

In order to illustrate our method, we selected two different phylogenetic trees created using our dataset and distance metric. As stated, identification of genera is dependent upon analysis type; by altering the way that malware samples were ordered, we changed the resultant tree and identified different parent-child relationships. For that reason, we calculated different sets of genera. The results of both studies are summarized in Table 1.

In both cases, the criteria for useful genera are satisfied – as our tree is built by attaching a new node to a single existing node, all subtrees within are already monophylyl. The genera are also compact – aside from the genera with large populations, which we see from the average and standard deviations are obvious outliers, the size of these genera are small. We see that the majority of genera identified have, at most, 40 members total spread over several generations. The third criterion – uniqueness of a characteristic – is not as obvious to see. Our distance metric examines the amount of mutual information within the system call traces. By having a very small distance between system call traces, we see that there is a high amount of information overlap between them, which indicates that the system call traces are similar. Similar system call traces imply similar operations. Therefore, the characteristic that the malware within a genus share is that similar operation.

4.1 Name-Based Ordering

The Name-Based Ordering of the malware samples is performed by grouping the malware samples in discovery date order, and then within each date, ordering the samples in alphabetical order. The rationale to this is that, while some samples may share a discovery date, the names of malware variants are assigned alphabetically. This leads to the assumption that Email-Worm.Win32.Bagle.p would have been discovered before Email-Worm.Win32.Bagle.q, since 'p' precedes 'q' indicating which variant was discovered and named first. The malware samples, ordered by date and then by name, are indexed in such a way that, for each iteration of tree building, we are adding M_j to the closest previous sample, even if it is M_{j-1}. This means that a sample can attach to a previous sample from the same day. Overall, the average distance between a parent and a child was 0.1095, with a standard deviation of 0.1348.

Using this ordering and we were able to identify 3,072 malware genera, which we found when $\tau = 0.0556$. Figure 1 shows the number of genera identified for various values of τ, indicating the peak of the graph where we identified the most genera. At that point, the average parent-child distance was 0.0145 with a standard deviation of 0.0168. Those genera contained anywhere from 2 to 419 members, averaging 5.17 with a standard deviation of 15.09. While there may have been a large genus within, the low average value shows that the genera tended to be smaller, but the standard deviation shows that there was a high amount of variation in those values. We found that there were anywhere from 2 to 16 generations of malware contained within each genus, with an average of 2.65 and a standard deviation of 1.25, which tells us that generally, a genus represented a small chain of offspring. We saw that the lifetime of a genus (based upon the age difference between the root node of the genus and the latest descendant) ranged from 0 to 3700 days, averaging 621.73 days with a standard deviation of 861.43 days. This indicates that many of the genera represent successful malware, such that it inspired newer versions, either through shared code or shared infection techniques.

4.2 Date-Based Ordering

The Date-Based Ordering of Malware takes a slightly different approach when ordering samples. As before, the samples are grouped in discovery date order, however no

	Name-Based	Date-Based
Genera	3072	2405
Members per Genus	2–419	2–213
Average	5.17	7.37
Std. Dev.	15.09	14.46
Generations per Genus	2–16	2–9
Average	2.65	2.74
Std. Dev.	1.25	1.23
Genus Longevity (days)	0–3700	1–3700
Average	621.73	1338.60
Std. Dev	861.43	991.50

Table 1. Summary of Identified Genera.

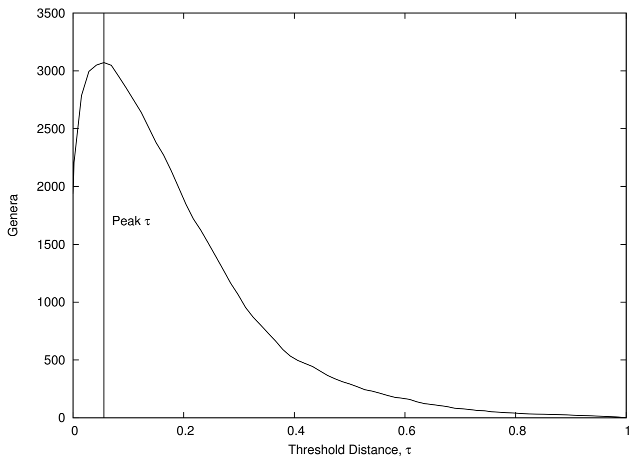

Figure 1. Genera identified for values of τ: Name-Based Ordering of Malware.

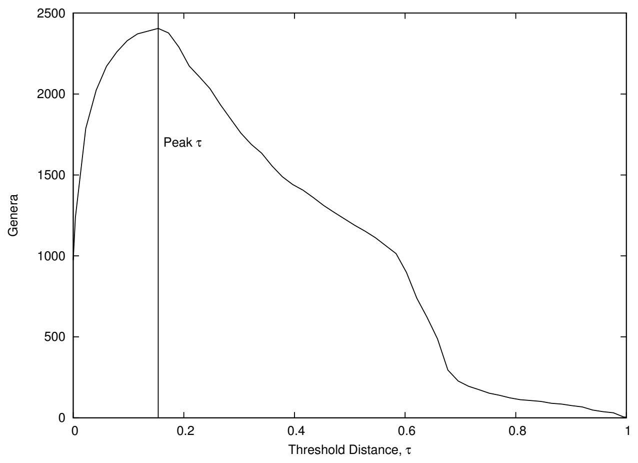

Figure 2. Genera identified for values of τ: Time-Based Ordering of Malware.

further sorting is performed. The rationale to this is that sample names may have been assigned arbitrarily, or by different analysts, and therefore do not necessarily inform us of the actual discovery order. In this case, we assume that all samples that arrive on a day are descended from samples that were discovered on previous days. In other words, at each iteration of tree building, we are adding $\mathcal{M}_j$ to the tree, and any of those new nodes can attach to any previous node. Since we cannot determine the order of samples discovered on the same day, they must be added to the tree at the same time. No sample can attach to a sample that was discovered on the same day – nodes cannot attach to other nodes in $\mathcal{M}_j$. Overall, the average parent-child distance in this tree was 0.1723 with a standard deviation of 0.1870.

With this ordering we identified 2,405 genera, which corresponds to a peak at $\tau = 0.1536$. The graph of the number of genera and the corresponding value of τ is shown in Figure 2. The average parent-child distance at that point was 0.0517 with a standard deviation of 0.0472. These genera contained between 2 and 213 members, averaging 7.37 with a standard deviation of 14.46, again indicating the high amount of variability in the number of members but still generally limiting the number of members to a small number. Similarly, there were between 2 and 9 generations encompassed by each genus, and with an average of 2.74 and a standard deviation of 1.23, again we see that the genera represent a short chain of offspring. The higher average lifespan of a genus, paired with a higher standard deviation, indicates how programming and infection techniques are constantly reused, and that some of those techniques are useful over a longer time frame.

5 Conclusion

We introduced the concept of the malware genus, a closely related group of malware samples. We presented a method by which these genera in malware collections can be identified. These genera fulfill the qualities for usefulness that are normally used when biologists are creating taxonomies for organisms. While the idea of creating malware families has been explored in the past, the genus is more restrictive and helps highlight close relationships between samples. The relationship between samples is not affected by the age difference between samples; in other words, it is reasonable to expect two very closely related (or even nearly identical) samples separated by several months or years would be in the same genus.

While our genera satisfied the stated criteria, it is important to note that there will be examples of a malware sample that seemingly should be included within a genus but is not. This can happen when you have malware samples within a broader family, grouped into that family due to some evaluation criterion, but which are not as close as expected. This can indicate a larger-scale change in the malware family. While the two genera may be part of the same overall family, the fact that they are separated shows that some sort of divergence in the phylogeny has occurred – it may be at this point that multiple new malware samples are spawned. This divergence point could be correlated with, for example, a milestone in OS development (e.g. a vulnerability patch which requires a significant change in the behavior of the malware for it to remain virulent) or the discovery of a new vulnerability, which requires a new method of infection but could have similar behavior as well.

Future directions of this work include an examination of the correlation between the genera that we identify and malware families as identified by other clustering and classification methods. While our classification method can be used alone as part of a larger malware analysis, identifying genera that adhere to more restrictive conditions can also aid in evaluation of malware grouping methods and assist in determining shared characteristics that occur in multiple related malware samples. The author of [15] presented a generalized form of the Jensen-Shannon Divergence that could be applied to an entire genus or population in order to determine the degree to which the samples relate to one another.

References

[1] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of internet malware. In *Proceedings of RAID 2007*, pages 178–197, 2007. `http://dx.doi.org/10.1007/978-3-540-74320-0_10`.

[2] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Krügel, and E. Kirda. Scalable, behavior-based malware clustering. In *Proceedings of NDSS 2009*, 2009. `http://www.isoc.org/isoc/conferences/ndss/09/pdf/11.pdf`.

[3] N. A. Campbell. *Biology*. The Benjamin/Cummings Publishing Company, Inc., New York, 4th edition edition, 1996.

[4] Classification of Species, 2009. `http://classes.entom.wsu.edu/348/classification.htm`.

[5] M. Elhadad. NLP09 Assignment 1: Computing KL Divergence, 2013. `http://www.cs.bgu.ac.il/~elhadad/nlp09/KL.html`.

[6] D. Endres and J. Schindelin. A new metric for probability distributions. *Information Theory, IEEE Transactions on*, 49(7):1858–1860, July 2003.

[7] D. Gao, M. K. Reiter, and D. X. Song. Behavioral distance for intrusion detection. In *Proceedings of RAID 2005*, pages 63–81, 2005. `http://dx.doi.org/10.1007/11663812_4`.

[8] Genus - definition from biology-online.org, 2014. `http://www.biology-online.org/dictionary/Genus`.

[9] M. Gheorghescu. An automated virus classification system. *Virus Bulletin Conference*, pages 294–300, Oct 2005.

[10] G. Jacob, H. Debar, and E. Filiol. Behavioral detection of malware: from a survey towards an established taxonomy. *Journal in Computer Virology*, 4(3):251–266, 2008.

[11] M. E. Karim, A. Walenstein, A. Lakhotia, and L. Parida. Malware phylogeny generation using permutations of code. *Journal in Computer Virology*, 1(1-2):13–23, 2005.

[12] J. Kinable and O. Kostakis. Malware classification based on call graph clustering. *Journal in computer virology*, 7(4):233–245, 2011.

[13] J. Z. Kolter and M. A. Maloof. Learning to detect and classify malicious executables in the wild. *Journal of Machine Learning Research*, 7:2721–2744, 2006. `http://www.jmlr.org/papers/v7/kolter06a.html`.

[14] T. Lee and J. J. Mody. Behavioral classification. In *Proceedings of EICAR 2006*, May 2006. `http://secureitalliance.org/blogs/files/73/1244/Behavioral_Classification.doc`.

[15] J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, Jan 1991.

[16] D. Mutz, F. Valeur, G. Vigna, and C. Kruegel. Anomalous system call detection. *ACM Trans. Inf. Syst. Secur.*, 9(1):61–93, 2006.

[17] Norman Sandbox, 2009. `http://www.norman.com/technology/norman_sandbox/`.

[18] F. Österreicher and I. Vajda. A new class of metric divergences on probability spaces and its applicability in statistics. *Annals of the Institute of Statistical Mathematics*, 55(3):639–653, 2003.

[19] Annual Report Panda Labs – 2013 Summary, 2013. `http://press.pandasecurity.com/wp-content/uploads/2010/05/PandaLabs-Annual-Report_2013.pdf`.

[20] Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel. Fast malware classification by automated behavioral graph matching. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, page 45. ACM, 2010.

[21] C. Riau. A virus by any other name: Virus naming practices, Jun 2002. `http://www.securityfocus.com/print/infocus/1587`.

[22] A. K. Seewald. Towards autmating malware classification and characterization. In *Proceedings of Sicherheit 2008*, pages 291–302, 2008. `http://alex.seewald.at/files/2008-01.pdf`.

[23] J. Seideman. Recent advances in malware detection and classification: A survey. Technical report, The Graduate School and University Center of the City University of New York, 2009.

[24] Threat explorer - spyware and adware, dialers, hack tools, hoaxes and other risks, 2012. `http://www.symantec.com/security_response/threatexplorer/`.

[25] VirusTotal, 2008. `http://www.virustotal.com`.

[26] VX heavens, 2010. `http://vx.netlux.org`.