Detecting Malware Samples with Similar Image Sets

Alex Long
Invincea Labs
alex.long@invincea.com

Josh Saxe
Invincea Labs
josh.saxe@invincea.com

Robert Gove
Invincea Labs
robert.gove@invincea.com

ABSTRACT

This paper proposes a method for identifying and visualizing similarity relationships between malware samples based on their embedded graphical assets (such as desktop icons and button skins). We argue that analyzing such relationships has practical merit for a number of reasons. For example, we find that malware desktop icons are often used to trick users into running malware programs, so identifying groups of related malware samples based on these visual features can highlight themes in the social engineering tactics of today's malware authors. Also, when malware samples share rare images, these image sharing relationships may indicate that the samples were generated or deployed by the same adversaries.

To explore and evaluate this malware comparison method, the paper makes two contributions. First, we provide a scalable and intuitive method for computing similarity measurements between malware based on the visual similarity of their sets of images. Second, we give a visualization method that combines a force-directed graph layout with a set visualization technique so as to highlight visual similarity relationships in malware corpora. We evaluate the accuracy of our image set similarity comparison method against a hand curated malware relationship ground truth dataset, finding that our method performs well. We also evaluate our overall concept through a small qualitative study we conducted with three cyber security researchers. Feedback from the researchers confirmed our use cases and suggests that computer network defenders are interested in this capability.

Categories and Subject Descriptors

Human-centered computing: Human computer interaction, Visualization; **Security and Privacy**

General Terms

Algorithms, Security, Human Factors

Keywords

Visualization, human computer interaction, security, malware

1. INTRODUCTION

A significant body of research engages the problem of identifying similarity relationships between malware samples, but to our knowledge no research literature thus far has engaged the problem of similarity analysis on the sets of visual image resources, such as desktop icons and button skins, that can be extracted from malicious executable files.

VizSec'14, November 10, 2014, Paris, France.

Such an approach to malware similarity analysis may be of practical utility in at least three ways. First, some malware is well protected against both static and dynamic analysis such that existing similarity analysis methods cannot detect its shared code relationships with other malware samples. This malware often contains images visible to analysts, however, as it needs these images to appear as desktop icons so as to trick the user into running the malware program. In these cases, similarity analysis over these sets of images can provide insight into malware sample relationships that existing malware feature comparison techniques cannot.

A second merit of shared malware image analysis is that identifying shared image relationships between malware samples can highlight themes in the ways in which malware authors are attempting to trick users into running malicious programs, accelerating this analysis by grouping visually similar images together. Third, in contexts where analysts sift through thousands of malware samples, visually grouping malware samples based on their shared visual attributes might aid analysts in deciding which malware samples to analyze first.

Motivated by such possible applications, in this paper we propose a method for identifying and visualizing shared embedded graphical asset relationships within large malware corpora. In exploring the problem of identifying and visualizing malware samples that contain similar sets of embedded images, we make two principal contributions:

1) We give a scalable and intuitive method for computing the similarity between pairs of malware samples based on the visual similarity of their sets of images, and evaluate this algorithm using hand-labeled malware image set pairs as ground truth. We further evaluate this concept in a series of human-subject interviews.

2) We give a visualization method which allows analysts to quickly understand the relationship between the set of images in a single malware sample to the sets of images within a large database of other malware samples.

The remainder of this paper is organized as follows. Section 2 gives a high level overview of the architecture and logical structure of our prototype system, and then gives detailed, reproducible descriptions of our algorithms and visualization. Section 3 contains our evaluation, including both traditional accuracy measures on our ability to match malware samples with similar image sets and a qualitative evaluation of the efficacy of our method. Section 4 treats related work, highlighting overlaps between our work and work in computer vision, information retrieval, malware analysis and malware visualization, and Section 5 gives a summary of our results and contributions.

2. SYSTEM OVERVIEW

Our system is comprised of the following logical pipeline, outlined in Figure 1. First, we extract images from a corpus of malware using the static analysis tool wrestool [1]. Second, we index the set

of images extracted from each malware sample into an image nearest neighbors database, which we have designed to identify near-duplicate relationships between malware images. Third, when the user queries the image nearest neighbors database with a malware sample from the corpus, we retrieve all of the malware samples connected by way of image set similarity relationships to the query sample (giving the user the option of expanding the network further by doing an *n*-hop depth first search of the graph starting from the query sample). Fourth, to present the user with malware sample image set similarity relationships, we visualize the sample similarity network. Below we give details about each part of this four stage process.

2.1 Image Extraction

To extract images from malware samples we leverage an open source tool called wrestool to parse the Windows Portable Executable file format of our malware binary test corpus and thereby extract graphical images from the malware samples' resources sections. In so doing, we discover images that are unobfuscated and uncorrupted and not dynamically computed by the malware program, thereby likely missing some images that may be displayed to the user when the malware is run (for example, images that might be used to trick the user into giving up their login credentials to a banking website). While this is a limitation of our system, we have found in practice that many malware samples contain unobfuscated images of interest, probably because malware samples often need to present users with desktop icons that trick them into clicking them before the malware sample can deobfuscate any other images and code that it may contain.

2.2 Image Set Indexing

To index malware samples' images in a database, we represent each individual malware sample image as a binary feature vector so that it can be easily compared, using a distance function, to other malware samples' images. We chose to use the "average hash" binary feature representation for each image, which is computed as follows:

1) The image is reduced to grayscale.
2) The image is then stretched or shrunk as necessary to 32x32 pixel resolution.
3) The 32x32 pixel matrix is then flattened into a single 1024 dimensional vector.
4) For each entry in the 1024 dimensional vector, if its brightness value is above the average value for the entries in the vector, it is set to 1, otherwise it is set to 0.

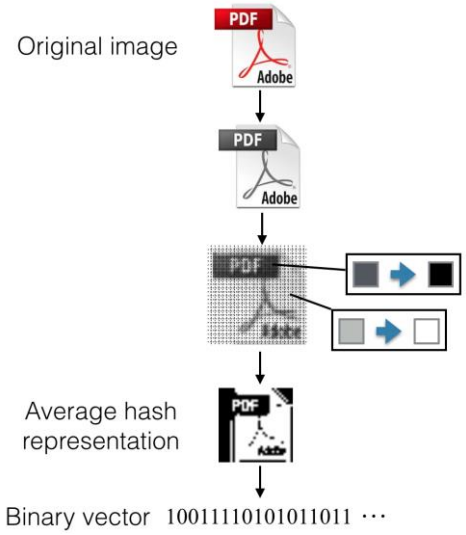

Figure 2. Transformation of an image through the average hash algorithm from start to finish.

We chose to use this "average hash" representation because it aligned well with our similarity analysis goals. For example, we want to match all extracted images that look like Internet Explorer icons regardless of differences in resolution or image format, and we found in practice that the average hash representation excels at this kind of comparison when vectors are compared using Euclidean distance. While other feature representations like SIFT and GIST are very effective on image similarity matching tasks that require rotation and scale invariance, we are not currently interested in capturing similarity relationships under these transformations, although exploring invariance to such transformations may be a fruitful direction for future work.

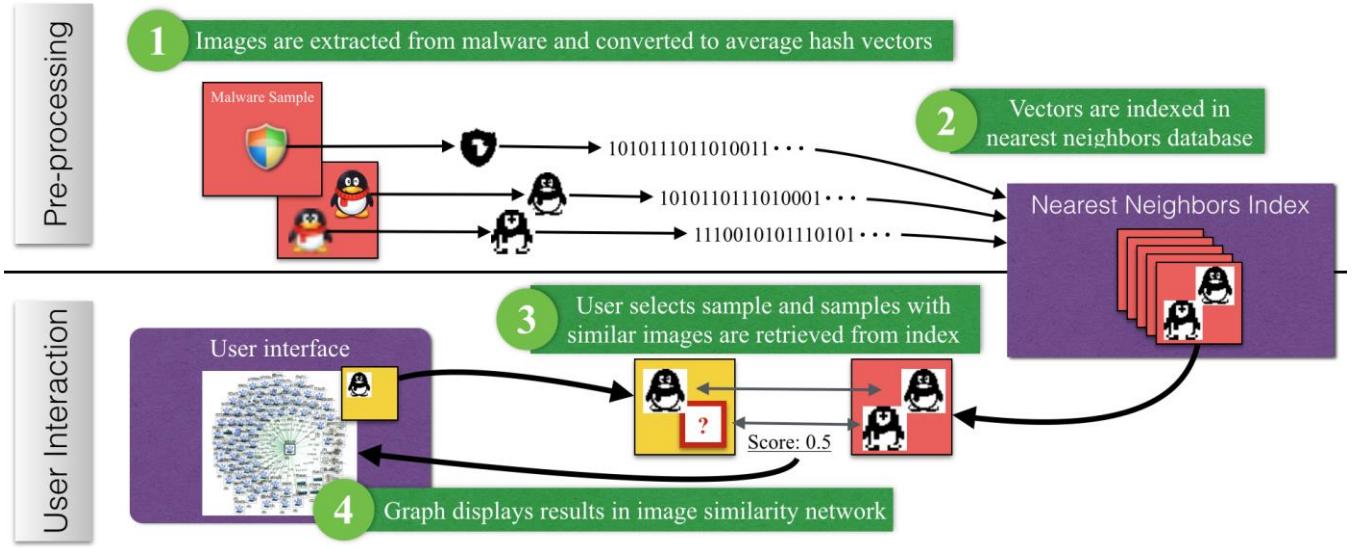

Figure 1. In our system images are extracted from malware and converted to average hash vectors (1) and indexed in a nearest neighbors database (2) which can then be queried through the user interface (3) to view results in a force-directed graph (4)

Once we have obtained binary vectors for the malware samples in our malware corpus, we store these vectors using a third party, open source in-memory nearest neighbor index, FLANN (Fast Library for Approximate Nearest Neighbors) [2]. We instantiate the FLANN index so that it performs approximate nearest neighbor retrieval under L2 distance using the random tree construction method described in [3], allowing us, given an image binary vector, to retrieve the top *n* most visually similar images based on the average hash model described above. The use of FLANN's approximate nearest neighbor functionality allows us to find similar images far more quickly than if we were using a brute force linear distance calculation method.

2.3 Retrieval of Malware Samples with Similar Sets of Images Given a Query Malware Sample

Having extracted images from malware samples, computed their average hash binary feature vector representation, and then indexed these vectors within the FLANN nearest neighbor store, we now have the problem of identifying, given a query malware sample, other malware samples with similar sets of images.

To explain how we have addressed this problem we first introduce two supporting concepts. First, the concept of the *image match*, a pair of images whose similarity value is above a threshold (in our experiments we set this threshold in FLANN to 3000) such that we believe the images are visual near duplicates. Second, *image set similarity*, a distance between malware sample's sets of images based on their *image match* relationships.

Our malware image set similarity calculation process occurs in two stages. First we iterate over all of the image vectors for the query malware samples, querying the *image nearest neighbor database* to identify the malware samples that have at least one *image match* with the images in the query malware samples.

Second we iterate over all of the malware samples that have at least one *image match* so as to compute their *image set similarity* with the query malware sample. This higher order similarity is calculated as follows: for each image in the malware sample with *fewer* images, we check to see if it has an *image match* in the other sample. If it does, we declare them a match and "remove" these samples from the analysis.

Then we check the next image in the malware sample with the smaller set of images, and so on. Finally, we divide the number of identified matching image pairs by the total number of images in the malware sample with *more* images (or by the total number of images in either malware sample, if they have an equal number of images) to obtain a final similarity score.

We take this seemingly unconventional approach to calculating malware samples' visual similarity because we believe the most intuitive model for malware visual similarity involves calculating to what degree there are one-to-one correspondence relationships between images in dyadic malware relationships. By only allowing images in either malware sample to be counted in a single image match, we prevent a situation where, for example, a malware sample with one Skype icon will be deemed to have 100% similarity with another malware samples with two differently sized Skype icons. Using our algorithm such a similarity would calculate to 50%, which we believe is the intuitive similarity between these two image sets.

2.4 User Interface

The UI was designed with the intent of putting the graphical content of the malware, which is the subject of our research, front and foremost. The majority of the screen space in the UI was therefore dedicated to a force-directed graph which displays the malware as nodes, with their corresponding extracted images displayed within the node.

The graph applies forces to the malware sample nodes which naturally push samples with dissimilar image sets away from each other and attract samples with similar image sets to each other, facilitating the potentially difficult task of finding relationships within a large body of malware. The interactivity of the graph also lends itself to this cause by allowing users to drag nodes and rearrange the graph as needed. After dragging a node, the graph's repulsion and attraction forces help to reorient the graph around the user's new arrangement.

The edges between nodes indicate that we found a similarity between the images of the connected nodes. On the left side of Figure 3 you can see a list of all the malware samples used in this experiment (about 200,000). By clicking any one of the samples in the list, the FLANN index is queried with the images found in the selected malware sample to find matching images. The graph then renders the results, with the selected malware sample drawn with a black border and the other malware samples drawn with a gray border. Edges are colored on a gradient from dark green to light green indicating whether the two linked malware samples have a high similarity score or a low similarity score.

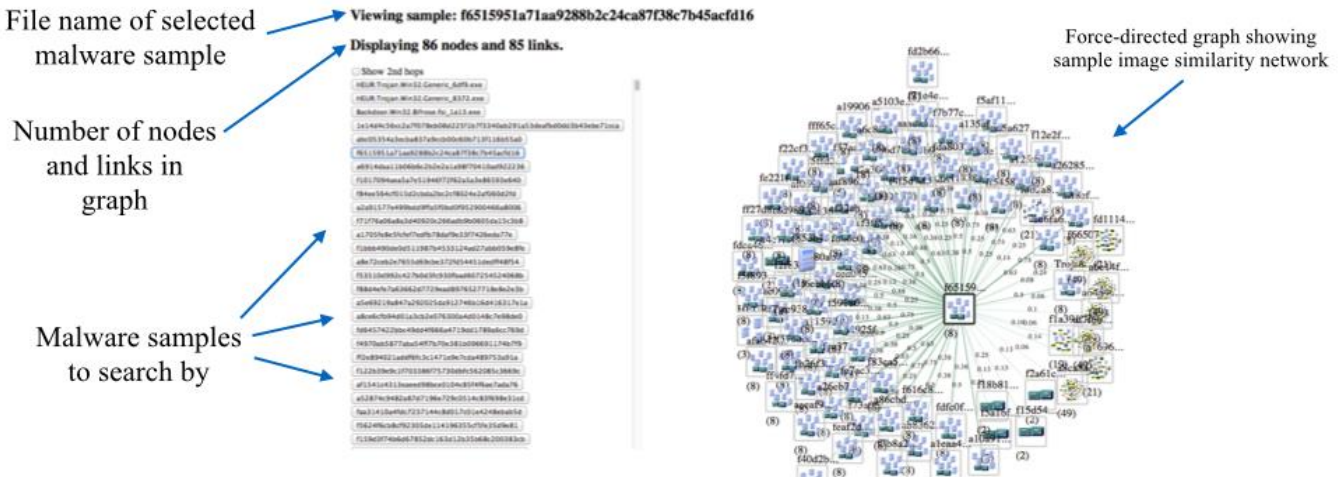

Figure 3. Interactive force-directed graph visualizing images found in malware samples.

The results of our image set similarity formula can also be seen in the visualization, in which the edges between nodes are labeled and darkened to indicate the similarity score between the image sets:

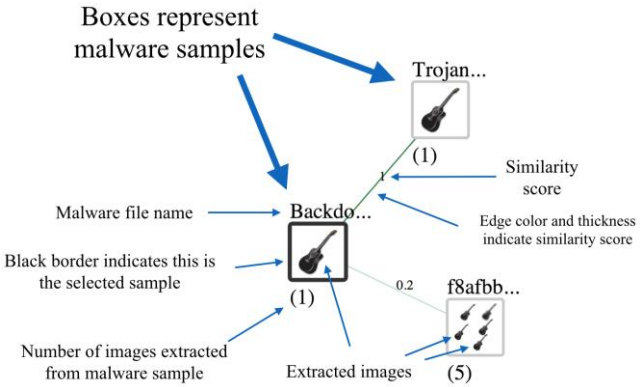

Figure 4. Visualization zoomed in to show similarity scores.

As you can see in Figure 4, the selected malware sample, Backdoor.Win32.Rbot.hyj_df88.exe, contains a single image of a guitar; the image count is displayed in parenthesis below the node for additional confirmation. Two other malware samples were found that contained similar images, one which also contains a single image of a guitar, and the other which contains five images of the same guitar. As you can see the from the darker green edge and the number displayed along the edge, the malware sample with a single guitar image has a 1 (100%) matching score, while the sample with five guitar images only has a 0.2 (20%) score. Our scoring formula is essentially penalizing the malware sample with five guitars for the four extra images.

We decided to further explore our method of looking at malware with similar images by also incorporating an *n*-hop depth first search of the graph starting from the focal sample. In Figure 5, a second hop has been enabled and each malware sample found to be sharing images with the focal sample (shown in the top left with a black border), also has any malware samples that it itself is sharing images with also shown and linked in the graph. As you can see, although the samples in the bottom right do not share any images with the focal sample in the top left, they do share an image with a sample which shares images with the focal sample.

The example in Figure 6 shows a similar structure as the previous figure at a larger scale.

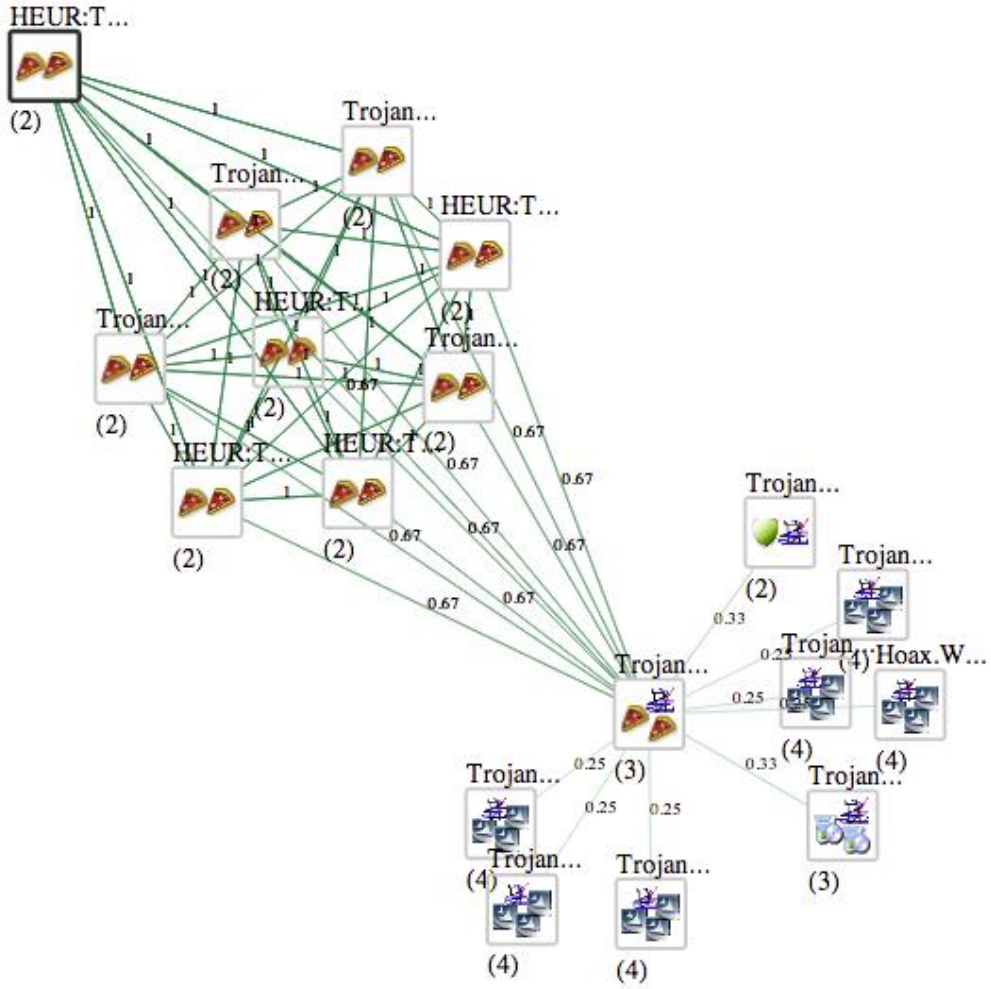

Figure 5. 2-hop visualization

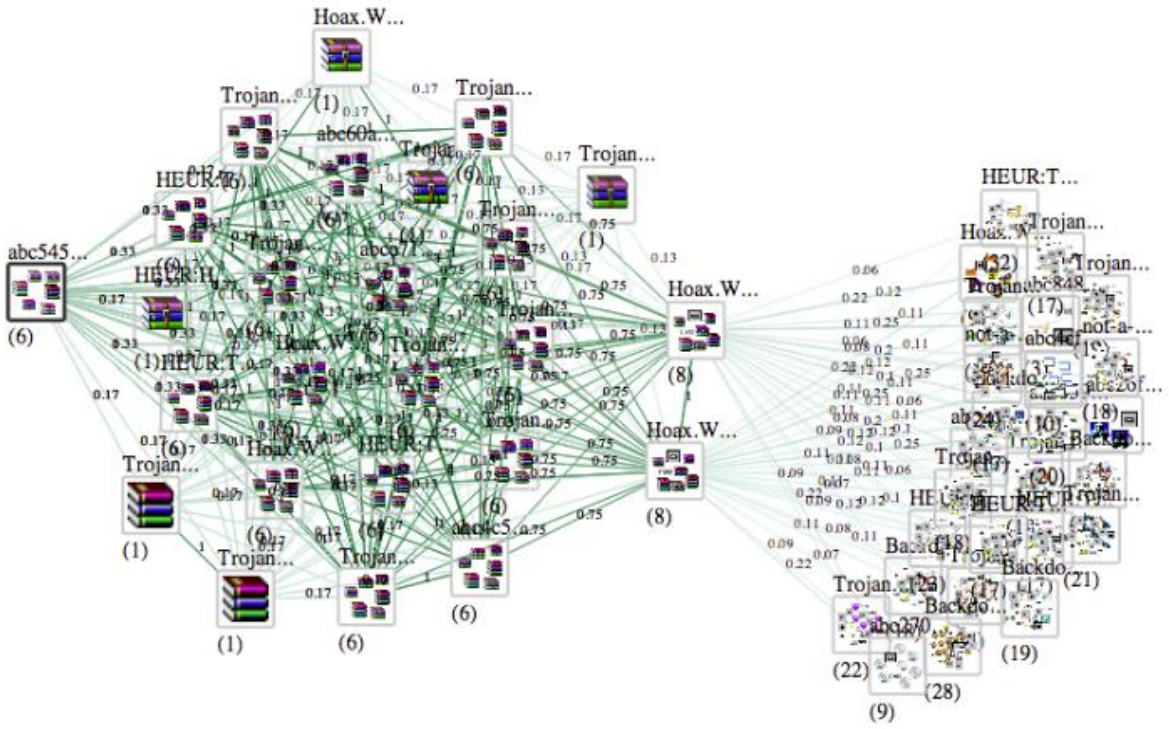

Figure 6. Second hop visualization of malware sample at larger scale.

In Figure 6, as in Figure 5, you can see two "clouds" of nodes. The first cloud (on the left) contains samples which are mostly single hops from the focal sample (on the far left) and the second cloud (on the right) contains second hop samples which have a very weak similarity score with a few of the single hop samples. The colors of the edges between the two clouds are a very faded shade of green, helping to quickly communicate to the user the fact that the malware samples in the second cloud have relatively few images in common with the single hop malware samples.

3. EVALUATION

To evaluate our method we assessed the accuracy of our algorithm for retrieving malware binaries that contain similar sets of images, and then engaged a set of human subjects to evaluate the overall usefulness of our method. In this section we give results for both of these evaluations and then offer a discussion of the strengths and weaknesses of our prototype tool.

3.1 Accuracy Evaluation

As described above, identifying malware binaries that contain similar sets of images requires that we identify image pairs that, while identical for practical purposes (say, two visually similar Internet Explorer icon images) are nonetheless encoded in different screen resolutions, colorations, or aspect ratios. Because the method that we propose above, based on the average hash feature representation and Euclidean distance metric, is not perfect, we chose to evaluate its ability to correctly identify malware samples with truly similar sets of images.

We evaluated our method using the following data. To test our hypothesis that most malware samples have at least one embedded image, we randomly sampled from a dataset of 2 million malware samples we received in late 2013 from a large United States federal agency. These samples, all Windows Portable Executable format malware, likely were active on the Internet up until late 2013, as the team we received the samples from is actively collecting and curating this malware database. We chose to use these samples to estimate the frequency of malware samples that have embedded images within malware datasets because they were randomly sampled from the largest malware dataset we had and we thereby assumed that results computed on this dataset would be the most meaningful.

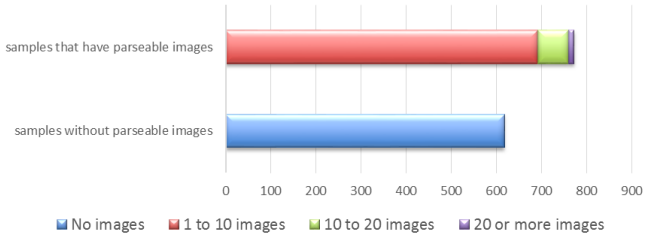

Figure 7. Chart showing the number of malware with at least 1 image (non-blue sections) versus the number of malware with no images (blue section) from a group of about 1,400 malware samples.

To perform our more detailed evaluation of our method's accuracy and utility, we used a dataset of malware samples obtained from the MD:Pro commercial malware feed that have been classified into malware families by the Kaspersky anti-virus engine [15]. We chose to use these malware samples because we had the intention of analyzing the relationships between the semantics of the images the samples contained and their anti-virus labels, although we have not yet performed this analysis and hope to report on this in future work. While the datasets we study in this paper are biased towards

malware of given origins, temporal distributions and provenance, we believe that the image analysis technique we propose here could be fruitfully evaluated against many malicious program datasets, such as mobile malware or malware targeting platforms other than Windows, as malware has the general problem of visually tricking the user into engaging with it.

To perform this evaluation we clustered by hand 200 malware binaries randomly chosen from the 200,000 used in our experiment based on their sets of images. Our criteria for placing malware in the same cluster was that at least 50% of their images needed to be near-duplicates. Given this ground truth, we then ran our malware image similarity analytics on these same malware pairs and evaluated how well our automated method reproduced this human labeling. More specifically, we used three evaluation metrics to gauge the accuracy of our system, which we list here:

1. $precision = \frac{|TP|}{|P|}$
2. $recall = \frac{|TP|}{|FN|+|TP|}$
3. $f1\text{-}score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

Where TP is the set true positives, P is the set of positives, FN is the set of false negatives, and |…| indicates the cardinality of the set.

As can be seen from these formulas, precision measures false positives (so that precision approaches zero as the number of false positives increases), recall measures false negatives (so that recall approaches zero as the number of false negatives increases) and f1-score combines both precision and recall into a global accuracy measure.

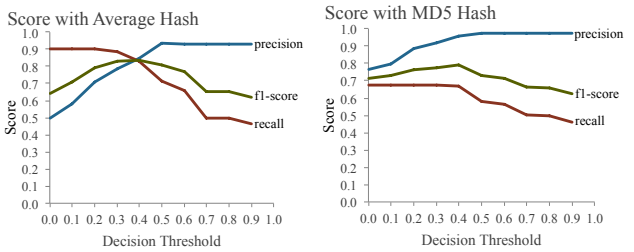

Figure 8. Precision, recall, and f1-score when using the average hash and MD5 hash algorithms to determine if two images should be matched.

For our accuracy evaluation, we evaluated our system in two ways. First, we used the average hash method described above to identify similar image pairs, tracking our system's precision, recall and f1-score as we varied our decision threshold (which decides whether or not, given malware binaries' image set similarity, those binaries should be grouped together). Second, we used the MD5 hash function to identify identical pairs of malware images and then measured our system's accuracy at retrieving malware binaries with similar sets of embedded images as we again varied our malware image set similarity threshold.

The results of these two evaluations accorded well with our expectations. As we expected, when we used MD5 matching to perform image pair matching, our system's recall was higher for most threshold values, given that many near-duplicate image pairs were deemed dissimilar, thus leading to false negatives. On the other hand, the MD5 matching method generated virtually no false positives, which is an intuitive result given that MD5 matching requires images to be identical at the byte level if they are to match.

As compared to the MD5 matching method, the average hash image matching method produced a better but still imperfect recall value over most of the test decision thresholds. This reflects the fact that the average hash function method, unlike the MD5 method, uses a genuine visual distance function between the images, such that the images can be near duplicates and still match. Near a threshold value of zero our system's recall approaches 1, but our system's precision becomes quite low (0.5). This is because at low thresholds images deemed to match are often visually dissimilar and thus lead to false positives.

Based on these evaluation data we conclude that the average hash method offers improvements over the MD5 hash method, although their accuracy is similar. The average hash method has a slightly higher f1-score value (at threshold 0.4, the MD5 method's f1-score is maximized at 0.79, whereas at threshold 0.4 the average hash method's f1-score is maximized at 0.84). At these maxima f1-scores each method has very different properties: whereas the MD5 method has a precision of 0.96 and a recall of 0.67, the average hash method has a precision of 0.85 and a recall of 0.83. We believe that because the average hash method has more even precision and recall scores in the best case that this method should be preferred. Using this method, analysts will miss few genuine similarity relationships between malware binaries, and, using our visualization, will be able to visually filter false positive malware relationships.

3.2 Analyst Feedback

To assess how malware image search functionality might help analysts and how analysts might use malware image search functionality in their computer network defense tasks, we recruited three researchers to provide feedback on our concept.

3.2.1 Participants

We recruited three cyber security researchers, which we will call P1, P2, and P3. All participants have past experience in computer network defense: P1 has 13 years, P2 has 1.5 years, and P3 has 16 years experience. All participants were male with ages ranging from 31 to 41. P1 has a bachelor's degree, P2 has a master's degree, and P3 has a PhD.

3.2.2 Experiment Design

The experiment facilitators showed the participants training material describing the malware similarity comparison algorithm. Next, the facilitators showed the prototype interface described above to the participants, and then the facilitators asked the participants five open-ended questions about malware image search functionality. Each session lasted about 20 minutes.

3.2.3 Procedure

Each participant was run in a separate session. During the training stage participants were shown a presentation describing how malware are compared based on their images. Next, the facilitators showed the prototype interface to the participants to show them real examples of malware that are similar based on the images they share. Finally, the facilitators asked the participants five questions about malware image search functionality to gather feedback about the technique.

3.2.4 Results

There were five questions the facilitators asked the participants after the participants viewed the training presentation and the prototype interface demo.

Question 1: "Would you use malware image search functionality in your work as a computer network defender? If so, how?" All participants said they would use malware image search in their work as a computer network defender. P2 said he would only use it to search for other malware that contain a very uncommon image, but the other two participants felt it would be useful as an additional tool to augment the other tools they have available to analyze malware and find similar malware samples.

Question 2: "Would the malware image search functionality change your analysis process? If so, how?" The participants did not feel like malware image search functionality would fundamentally change their analysis process, but P1 and P3 thought it would be a useful addition to existing analysis and comparison tools.

Question 3: "What questions would malware image search functionality help you answer?" P1 and P3 thought it would be useful to infer how malware is masquerading or how malware is attempting to trick users. P2 and P3 thought it would be useful for finding other malware with the same uncommon images. P3 thought it would be useful to find certain types of malware that have the same behavior, and gave the example of searching for malware that have icons related to asking victims for money.

Question 4: "How would you change the malware image search functionality to suit your needs for computer network defense?" P1 and P3 suggested building classes of images (e.g. images associated with Windows services, commercial products, etc.) so that analysts could focus on specific types of images they are looking for, or searching for malware by images that are associated with specific functionality. P2 and P3 suggested adding the ability to select an image and find all malware that have a similar image. P2 suggested modifying the similarity score function so that it does not include generic or very common images, thereby emphasizing similarity relationships between malware samples that have unique or uncommon images.

Question 5: "Do you have any other comments?" P1 suggested showing the images and malware samples in a table instead of a network diagram because he thought that might be easier to read than some networks that are densely connected. P2 commented that malware image search functionality is an interesting concept, but he is unsure whether it has a lot of use. P3 described it as clever and a "cool feature."

3.2.5 Discussion

Participants' reactions to the concept of malware image search functionality ranged from tepid to enthusiastic. P2 felt that the primary use case was to find malware with the same uncommon images, and most of his comments were aimed at that use case. However, P1 and P3 thought that malware image search functionality had at least two use cases (understanding how malware attempts to trick users, and finding malware that use uncommon images) and offered feedback to improve that functionality and expand its use cases.

The participants' feedback agrees with the use cases we had in mind while creating the malware image search functionality, but it also indicates there is future work to refine the concept and integrate it into computer network defense workflows.

4. RELATED WORK

To our knowledge our work is the first to support relational analysis of malware binaries' sets of graphical assets. However, because our method touches on multiple topical areas it relates significantly to a range of previously published work. One such body of work is the small literature on malware visualization. [4] describes a genetic sequence inspired visualization for understanding similarities and differences between malware behavioral traces, and [5] describes the application of a treemap visualization to data drawn from malware behavioral traces, arguing for the efficacy of this method in affording both comparisons between malware samples as well as the method's utility in providing insight into individual malware samples. While our work emphasizes malware comparisons it differs from this work in that it focuses on similarities and differences in malware's graphical assets. Further afield from our work, [6] describes the application of a force directed graph visualization to the problem of understanding malware dynamic traces, showing the usefulness of this visualization in revealing malware unpacking behavior. While related to our work in that it visualizes malware properties for malware analysts, this work is focused on understanding the program logic of a given malware sample.

A second related body of work is the literature that emerged in the last decade on automatic malware similarity identification and malware clustering. [7] describes a technique by which malware file byte sequences are rendered as images and then clustered using a near-duplicate matching algorithm. This technique is related in interesting ways to our method in its mapping of computer vision solutions onto relational malware analysis problems and its emphasis on the importance of analysis interpretability for analyst users. Whereas our focus is on relating malware samples by way of their shared graphical assets, the authors' focus is on grouping malware samples that share syntactic byte sequence features and then visually presenting these relationship inferences to malware analysts. Similar in inspiration, [8] describes a technique which involves clustering and visualization of malware corpora by way of their shared system call behavior relationships. Other malware clustering literature such as that described in [9], [10] and [11] gives methods for clustering malware based on application binary interface metadata features, call graph features, and dynamic behavioral trace features. Our work differs from this work in that whereas this work focuses on program logic and is non-visual, our work focuses on malware graphical assets and involves visualization.

A third body of work relevant to our own is work done on applying computer vision methods to cybersecurity detection and analysis problems. The work described in [12] and [13], for example, utilizes computer vision methods, combined with auxiliary signals (such as website whitelists and blacklists) to classify websites as either created with fraudulent intent or not, specifically looking for near-duplicate relationships between the visual features of known malicious sites and candidate sites. This work is similar to our own in its fusion of the computer vision domain with the cybersecurity domain, but focuses on websites, not malware, employs different computer vision algorithms from those we apply, and does not emphasize visualization and data exploration.

A fourth related literature describes work performed on the problem of near-duplicate image detection. This work relates to ours in that when we compare malware images we are looking for syntactic, as opposed to semantic similarities—we would like to find malware image reuse relationships rather than malware whose images shares common ontological properties (images of beaches, for example).

One example of past work on near-duplicate image detection is [14], which describes a scheme for clustering billions of images on the web based on their near-duplicate relationships. Our work differs from this past work in that we use a different feature representation for our images which we believe is more appropriate to malware graphical assets, and because we give a second-order similarity function between malware samples that produces a similarity based on the similarities and differences between sets of images.

5. CONCLUSION

In this paper we present a method for computing similarity between malware based on their sets of images. We also give a visualization and user interface to display image similarity relationships between malware and to support users in exploring these relationships.

We quantitatively evaluated our malware image similarity method against 200 ground truth malware samples and conclude that our method works well for identifying image set similarity relationships between malware samples. We also qualitatively evaluated the concept of clustering malware by image set similarity with three researchers who have experience in computer network defense. These researchers were generally interested in using this concept for malware analysis, and they confirmed our use cases. Furthermore, the researchers suggested two additional use cases.

Our work in this paper on malware image set comparison forms the basis for several opportunities for future work:

1. Query by example image. This has at least two use cases: (1) finding all malware that contain a given signature image or uncommon image and (2) finding malware that have an image similar to a set of query images, e.g. to find malware that manipulate users in the same way or have the same command and control interface.

2. Group malware by their images. This could provide analysts with an overview of images extracted from malware in a malware corpus and also allow analysts to browse malware by the image classes contained in the malware.

6. ACKNOWLEDGMENTS

This work was funded by the Defense Advanced Research Projects Agency (DARPA). We thank our user study participants for their time and valuable insight. The views expressed are those of the author(s) and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

7. BIBLIOGRPAHY

[1] O. Liljeblad, “icoutils.”

[2] M. Muja and D. G. Lowe, “Flann, fast library for approximate nearest neighbors.” 2009.

[3] M. Muja and D. G. Lowe, “Fast matching of binary features,” in Computer and Robot Vision (CRV), 2012 Ninth Conference on, 2012, pp. 404–410.

[4] J. Saxe, D. Mentis, and C. Greamo, “Visualization of shared system call sequence relationships in large malware corpora,” in Proceedings of the Ninth International Symposium on Visualization for Cyber Security, 2012, pp. 33–40.

[5] P. Trinius, T. Holz, J. Gobel, and F. C. Freiling, “Visual analysis of malware behavior using treemaps and thread graphs,” in Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on, 2009, pp. 33–38.

[6] D. A. Quist and L. M. Liebrock, “Visualizing compiled executables for malware analysis,” in Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on, 2009, pp. 27–32.

[7] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: visualization and automatic classification,” in Proceedings of the 8th International Symposium on Visualization for Cyber Security, 2011, p. 4.

[8] K. Blokhin, J. Saxe, and D. Mentis, “Malware Similarity Identification Using Call Graph Based System Call Subsequence Features,” in Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on, 2013, pp. 6–10.

[9] G. Wicherski, “pehash: A novel approach to fast malware clustering,” in 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2009.

[10] J. Kinable and O. Kostakis, “Malware classification based on call graph clustering,” J. Comput. Virol., vol. 7, no. 4, pp. 233–245, 2011.

[11] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, “Scalable, Behavior-Based Malware Clustering.,” in NDSS, 2009, vol. 9, pp. 8–11.

[12] S. Afroz and R. Greenstadt, “Phishzoo: Detecting phishing websites by looking at them,” in Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on, 2011, pp. 368–375.

[13] M. Dunlop, S. Groat, and D. Shelly, “GoldPhish: using images for content-based phishing analysis,” in Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on, 2010, pp. 123–128.

[14] T. Liu, C. Rosenberg, and H. A. Rowley, “Clustering billions of images with large scale nearest neighbor search,” in Applications of Computer Vision, 2007. WACV’07. IEEE Workshop on, 2007, p. 28.

[15] Frame4, “MD:Pro Malware Feed.” [Online]. Available: http://www.frame4.net.