SEEM: A Scalable Visualization for Comparing Multiple Large Sets of Attributes for Malware Analysis

Robert Gove[1], Joshua Saxe[1], Sigfried Gold[2], Alex Long[1], Giacomo Bergamo[1]
[1]Invincea Labs, [2]Zachary Piper
{robert.gove, josh.saxe, sigfried.gold, alex.long, giacomo.bergamo}@invincea.com

ABSTRACT

Recently, the number of observed malware samples has rapidly increased, expanding the workload for malware analysts. Most of these samples are not truly unique, but are related through shared attributes. Identifying these attributes can enable analysts to reuse analysis and reduce their workload. Visualizing malware attributes as sets could enable analysts to better understand the similarities and differences between malware. However, existing set visualizations have difficulty displaying hundreds of sets with thousands of elements, and are not designed to compare different types of elements between sets, such as the imported DLLs and callback domains across malware samples. Such analysis might help analysts, for example, to understand if a group of malware samples are behaviorally different or merely changing where they send data.

To support comparisons between malware samples' attributes we developed the Similarity Evidence Explorer for Malware (SEEM), a scalable visualization tool for simultaneously comparing a large corpus of malware across multiple sets of attributes (such as the sets of printable strings and function calls). SEEM's novel design breaks down malware attributes into sets of meaningful categories to compare across malware samples, and further incorporates set comparison overviews and dynamic filtering to allow SEEM to scale to hundreds of malware samples while still allowing analysts to compare thousands of attributes between samples. We demonstrate how to use SEEM by analyzing a malware sample from the Mandiant APT1 New York Times intrusion dataset. Furthermore, we describe a user study with five cyber security researchers who used SEEM to rapidly and successfully gain insight into malware after only 15 minutes of training.

Categories and Subject Descriptors

[**Security and privacy**]: Intrusion/anomaly detection and malware mitigation – *Malware and its mitigation*

[**Human-centered computing**]: Visualization – *Visualization application domains*

General Terms

Security, Human Factors.

Keywords

Computer Security, Malware, Sets, Venn diagrams, Visualization.

VizSec '14, November 10 2014, Paris, France
Copyright 2014 ACM 978-1-4503-2826-5/14/11...$15.00
http://dx.doi.org/10.1145/2671491.2671496

1. INTRODUCTION

The huge volume of unique malicious software (*malware*) gathered in today's computer security malware repositories are connected through a dense web of shared attribute relationships. For example, malware variants often borrow code from one another, use the same command and control servers, and use the same graphical images to trick users into executing them.

Over the last decade, researchers and practitioners have recognized that performing similarity analysis and clustering on pieces of malware (*malware samples*) is an important research problem because it indicates shared provenance and reduces analyst workload. Researchers have proposed various statistical and machine learning approaches to automatically compare malware samples (e.g. [5]). However, these approaches are limited by a lack of associated visualization techniques that help analysts understand *why* these approaches claim that malware samples are similar or of the same software lineage.

Visual similarity analysis could reveal the complex similarities and differences between a specific sample, which we call the *focal sample*, and the rest of the malware in the corpus, which we call the *comparison samples*. With that knowledge, analysts could leverage previous reverse engineering work performed on the focal sample's older "relatives," thereby accelerating and adding information to the reverse engineering process. Alternatively, analysts may realize that a new malware sample is genuinely new, i.e. not simply a polymorphic variant of previously observed malware. Set visualizations could be useful for understanding similarities and differences between malware if we consider *malware samples* to be like *sets*, malware *attributes* (e.g. specific DLLs, like `kernel32.dll`, or function calls, like `ReadFile()`) to be like set *elements*, and *categories* of attributes (e.g. all the imported DLLs or all the function calls) to be like *subsets*. (See Figure 1.) However, as we describe in Section 2, existing set visualizations have scalability issues or are not suited to comparing multiple categories across malware samples. Specifically, they do not support comparing large numbers of sets, or they do not support comparing sets with large numbers of elements, or they do not support comparing elements between sets based on the category of the element.

As an alternative to these visualization methods, we introduce the Similarity Evidence Explorer for Malware (SEEM), which we describe in Section 3. SEEM is a scalable visualization tool for simultaneously comparing a large corpus of malware across multiple sets of attributes (such as the sets of printable strings and function calls). SEEM's novel design partitions malware attributes into meaningful categories to compare across malware samples, and further incorporates comparison overviews and dynamic filtering to allow SEEM to scale to hundreds of samples while allowing analysts to compare thousands of attributes between samples.

We show the utility of SEEM by first demonstrating in Section 4 how to generate insight about a malware sample from the Mandiant APT1 New York Times intrusion set, followed by a usability study in Section 5 where we evaluated SEEM with five participants. We find that novice SEEM users can quickly learn SEEM and use it to generate insights about malware. Comments from the participants confirm our design decisions and that SEEM addresses real-life malware analysis challenges.

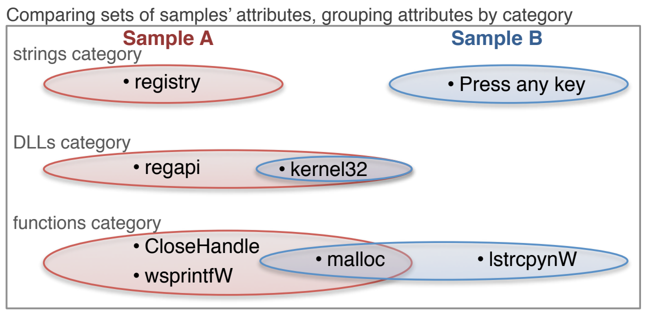


Figure 1. Some comparison tasks require analysts to compare malware sample attributes (set elements) by their categories (subsets), such as comparing the printable strings, imported DLLs, or function calls from different samples.

2. RELATED WORK

Because SEEM can be considered a tool for comparing sets, here we describe related work on set visualizations, as well as related work on malware visualizations.

Euler and Venn diagrams are among the most common techniques for displaying overlaps among sets. These diagrams can show the elements that belong to each set by drawing the elements inside the contours, but this suffers scalability issues [25] and becomes cluttered when there are lots of elements. Some variations use area to indicate the number of elements in each set instead of drawing each element [6, 14, 26], but this lacks the ability to see which elements are in each set.

UpSet [16] scales well to large numbers of elements and it is designed to visualize the intersections between many different sets. However, UpSet does not directly support comparing an individual set to other sets, nor is it clear that UpSet allows users to compare large numbers of sets.

LineSets [1], Bubble Sets [7], and Kelp diagrams [9], and KelpFusion [17] are visualizations that rely on set elements being pre-positioned (e.g. on a map or in a force-directed network layout) and then drawing lines or contours to indicate set membership. However, the published examples in those papers show less than 100 elements distributed across less than 10 sets; therefore, it is not clear if these approaches would scale to thousands of elements distributed across hundreds of sets.

Radial Sets [2] and Set'o'grams [11] scale well to dozens of sets with thousands of elements, but they do not support comparing multiple sets across multiple categories simultaneously (e.g. they could support comparing different sets of imported DLLS, but not comparing different malware samples and the sets of imported DLLs, printable strings, and function calls from each of the malware samples). Also, since Set'o'grams require each element to have its own column, Set'o'grams would have difficulty scaling to thousands of elements because they could not all fit on a single screen.

Parallel Sets [3] and Elastic Lists [23] allow users to explore categorical data and a single set of elements by its different categories, but these visualizations do not compare multiple sets by different categories.

Re-orderable matrices [4] can show the elements that belong to each set, as in Kim *et al*. [15], but cannot show an overview when there are thousands of points because the matrix becomes too large to show on one screen.

There are many examples of automatic malware comparison research [5, 21], but we are not aware of many visualizations designed to support the malware comparison process. Quist and Liebrock [20], Conit *et al*. [8] and cantor.dust [10] visualize individual malware samples to aid the reverse engineering workflow. Trinius *et al*. use treemaps and thread graphs to perform comparative analysis on malware with the goals of identifying malicious software and classifying malicious behavior [24], but our goal is to understand the similarities and differences of malware samples.

Saxe *et al*. [21] use multidimensional scaling to visualize similar malware samples. Their work incorporates a panel that allows users to highlight and select samples based on their attributes (e.g. callback domains). This approach does not directly show users how many attributes are associated with each malware sample, nor does it directly show how many attributes are in common between any pair of samples.

Other approaches represent binary files as an image that can then be compared to images of other binary files [18] [13], but this does not expose the attributes of the binaries to users.

3. SYSTEM DESCRIPTION

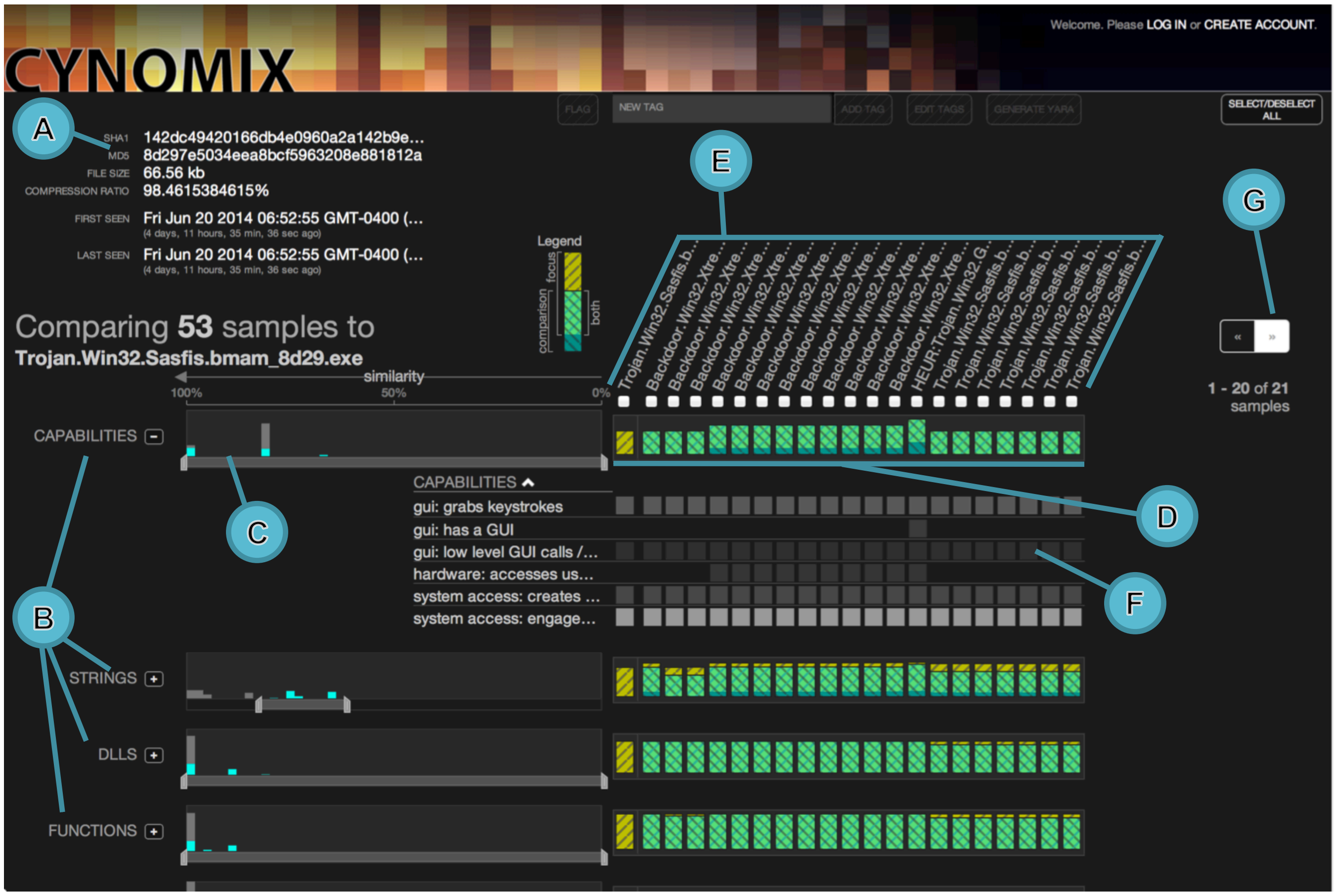
Figure 2 shows the SEEM interface. The top left of SEEM shows basic information about the focal sample: its SHA1 and MD5 hashes, the file size, compression ratio, and the dates the sample was first and most recently uploaded to the system.

3.1 Visualizations

SEEM uses categories to visually compare the focal sample to the comparison samples. Currently there are nine categories:

1. Predicted capabilities.
2. Printable strings.
3. DLL imports.
4. External function calls.
5. Image resources.
6. Tags applied to samples by system users.
7. IP addresses.
8. Hostnames.
9. Registry keys.

Each category has three visualizations: (1) A similarity histogram to give an overview of how similar the comparison samples are to the focal sample, (2) an ordered list of Venn diagrams to visualize the number of attributes and the overlap in attributes between the focal sample and each comparison sample, and (3) a matrix showing the relationship between malware samples and attributes (see Figure 2). Each comparison sample is compared to the focal sample on each category, allowing analysts to simultaneously see how the samples are similar to the focal sample with respect to each category.



SHA1	142dc49420166db4e0960a2a142b9e...
MD5	8d297e5034eea8bcf5963208e881812a
FILE SIZE	66.56 kb
COMPRESSION RATIO	98.4615384615%
FIRST SEEN	Fri Jun 20 2014 06:52:55 GMT-0400 (... (4 days, 11 hours, 35 min, 36 sec ago)
LAST SEEN	Fri Jun 20 2014 06:52:55 GMT-0400 (... (4 days, 11 hours, 35 min, 36 sec ago)

Comparing **53** samples to
Trojan.Win32.Sasfis.bmam_8d29.exe

CAPABILITIES
- gui: grabs keystrokes
- gui: has a GUI
- gui: low level GUI calls /...
- hardware: accesses us...
- system access: creates ...
- system access: engage...

Figure 2. The SEEM interface. Basic information about the sample (SHA1 hash, file size, etc.) is on the top left (A). Underneath (B) are nine categories (four are shown here), each of which has a similarity axis (C), a list of Venn diagrams (D) for each sample in view (E), and a relationship matrix (F) that users can open by clicking the "+" button next to the category name. The relationship matrix for capabilities is colored based on the confidence of the prediction that the sample has the given capability. Users can paginate through the samples in the Venn diagram list and relationship matrix with previous/next buttons (G).

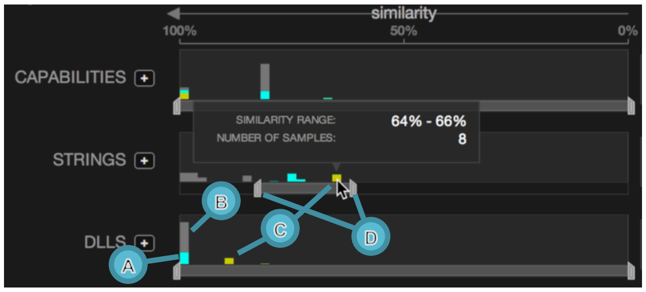
3.1.1 Similarity Histogram

The similarity histogram (see Figure 2C and Figure 3) shows an overview of how similar the comparison samples are to the focal sample, and includes a range slider (Figure 3D) to filter samples that are outside of the specified similarity range. Combined, these allow SEEM to compare an arbitrary number of different samples to the focal sample. Similarity is measured using the Jaccard index, where A and B are the set of attributes in a given category from the focal sample and a comparison sample, respectively:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Where $|A|$ is the number of elements in set A, and $J(A,B) = 1$ if $|A \cup B| = 0$. Each category is visualized in a separate histogram, allowing analysts to quickly understand how the comparison samples are distributed across all of the categories. The histogram's *x*-axis is the Jaccard index, where the bars drawn towards the left represent comparison samples that have high Jaccard index with the focal sample, and bars drawn towards the right represent comparison samples with low Jaccard index with the focal sample.

A range slider at the bottom of each histogram helps analysts to filter through arbitrarily large numbers of malware samples. Any sample that is outside of the range for one category is also filtered out of the other categories as well (see Figure 2).



SIMILARITY RANGE: 64% - 66%
NUMBER OF SAMPLES: 8

Figure 3. Similarity histograms for predicted capabilities, printable strings, and DLL imports. Bar height indicates the number of comparison samples, and the horizontal position indicates how Jaccard index with the focal sample. Bar has three colors: cyan bars (A) are samples that have not been filtered out, and gray bars (B) are the total number of comparison samples. Bars become yellow (C) when users hover over them, and yellow bars appear in the other similarity histograms to indicate where the comparison samples in the hovered bar appear in the other similarity histograms. Hovering on a bar also displays a tooltip with the number of samples in the bar. Users can adjust the range sliders (D) for each histogram to filter out samples that are not within a certain similarity range.

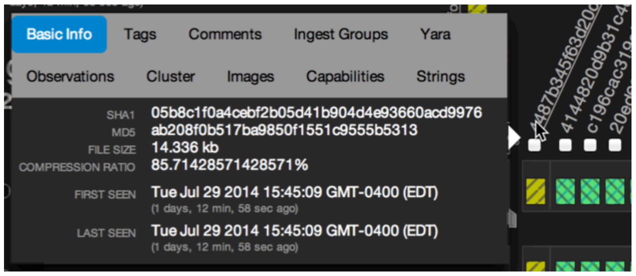
The histogram bars are displayed using 3 colors. The heights of the cyan bars (Figure 3C) represent the comparison samples that were not filtered out, and the heights of the gray bars (Figure 3B) represent the total number of comparison samples (both filtered and unfiltered). This filtering action is also reflected in the Venn diagram list and relationship matrix visualizations.

To see where the samples in one bar appear in the other histograms, users can hover on a bar in the similarity histogram, and SEEM displays a tooltip that shows users the number of samples represented by the bar and the Jaccard index (expressed as a percent) of the samples in the bar. Hovering over a bar on one category draws yellow bars on the other categories to show how those samples are distributed on other categories (see Figure 3C).

3.1.2 *Venn Diagram List*

The Venn diagram list (Figure 2D) shows pages of malware samples. Each column in the Venn diagram list is a sample. The angled text at the top of the column is an antivirus engine label, the sample filename if no antivirus engine label is available, or the sample's SHA1 hash if there is no antivirus label or filename. The comparison samples are sorted in descending order by average Jaccard index across all nine categories. Clicking the sample name will refocus SEEM with the clicked sample as the focal sample. If a user hovers over the sample name, a tooltip pops up that has additional information about the sample, such as file size and compression ratio (see Figure 4).

The paginated view shows the 20 comparison samples most similar to the focal sample, plus the focal sample itself, where each sample is a column. The first column is the focal sample and is always in view, regardless of which page of samples users are on or which filtering operations users perform. Users can click the right and left arrow buttons to navigate to the next and previous pages of samples. This list is updated based on filtering performed using the range sliders on the similarity histograms.



SHA1 05b8c1f0a4cebf2b05d41b904d4e93660acd9976
MD5 ab208f0b517ba9850f1551c9555b5313
FILE SIZE 14.336 kb
COMPRESSION RATIO 85.71428571428571%
FIRST SEEN Tue Jul 29 2014 15:45:09 GMT-0400 (EDT)
(1 days, 12 min, 58 sec ago)
LAST SEEN Tue Jul 29 2014 15:45:09 GMT-0400 (EDT)
(1 days, 12 min, 58 sec ago)

Figure 4. SEEM pops up a new tooltip with additional information when users hover over a malware sample's name.

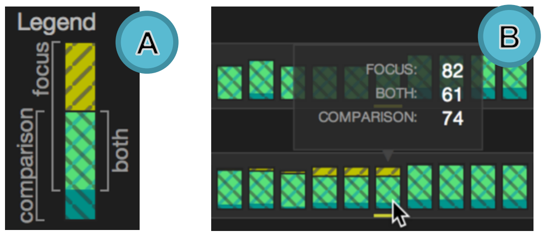


Figure 5. (A) The legend for Venn diagram design. The orange section at the top represents the attributes only in the focal sample. The cyan section at the bottom represents attributes only in the comparison sample. The green section in the middle represents the attributes in both the focal sample and the comparison sample. (B) Hovering over a Venn diagram reveals a tooltip showing the number of attributes in the focal sample, the comparison sample, and the intersection.

The Venn diagrams show the types of similarity (overlap, strict subset, disjoint) in each category. See Figure 5A for the Venn diagram legend.

The height of the Venn diagrams is proportial to the number of attributes in that category; therefore if one Venn diagram is shorter than another, then that Venn diagram has fewer distinct attributes in it. This allows analysts to compare two Venn diagrams in a category and be able to compare the number of elements in each set. If a sample has no attributes in that category, the height of its rectangle in the Venn diagram is zero. For example, if the focal sample and all comparison samples do not have any extracted IP addresses, then the Venn diagrams in the IP address category all have no height. If the focal sample does not have any extracted IP addresses, but a comparison sample has three extracted IP addresses, then the focal sample component of the Venn diagram has no height but the comparison sample's component of the Venn diagram does have height.

Hovering on a Venn diagram shows users a tooltip that displays the number of attributes in the focal sample, the number of attributes in the comparison sample, and the number of attributes in the intersection between the two (see Figure 5B).

3.1.3 *Relationship matrix*

After examining the similarity histograms and the Venn diagrams, analysts may wish to know exactly which attributes in a category belong to the samples in view. To do this, analysts click the "+" button next to the category name to view all of the attributes in that category (Figure 2F and Figure 6). For example, analysts can click the button next to "Capabilities" to view the capabilities relationship matrix that shows all of the capabilities that appear in any of the comparison samples.

This matrix can be sorted by approximate frequency of the attribute (e.g. how many samples in the database have the same printable string), by lexicographical order, or by occurrence in a specific sample. The relationship matrix for the printable strings category has an additional column that indicates which printable strings provide evidence for a given predicted capability, if any. Sorting helps users navigate the matrix and allows the visualization to scale to large numbers of elements while still being useful to users.

If the category has confidence associated with it (e.g. the confidence of a capability prediction) then the matrix cells are colored ranging from dark gray, representing low confidence, to light gray, representing high confidence.

3.2 Linked Views

The similarity histograms are linked to the Venn diagram list, allowing users to see if samples in a histogram bar are currently in view in the Venn diagram list, and vice versa. When users hover on a bar in a similarity histogram, the samples represented by that bar highlight in the Venn diagram list. Likewise, when users hover on a Venn diagram, the corresponding bar on each similarity histogram highlights to indicate the Jaccard index of that comparison sample from the focal sample.

3.3 Sample Annotation

Users can select samples (either the focal sample or comparison samples) by clicking the checkboxes next to their names. Next, users can type a tag name and apply it to the selected samples. SEEM then updates the "tags" category so that all the visualizations reflect the new tag.

3.4 Implementation

SEEM is built into Cynomix [12], a web-based system for analyzing large-scale malware corpora. SEEM is implemented in HTML, CSS, and JavaScript using the AngularJS and D3 JavaScript libraries. The back-end architecture of Cynomix consists of a web server built using the Tornado framework and a MongoDB database that is kept in sync with an Elasticsearch index for fast faceted search.

The database stores the cluster for each sample, along with the nearest neighbors for each malware sample (i.e. malware samples that are at least 50% similar). The database also stores several attributes of the malware samples, including predicted capabilities (using the capability detection described by Saxe *et al*. [22]), printable strings, DLL imports, external function calls, tags applied to the sample by Cynomix users, IP addresses, hostnames, registry keys, and image resources. Predicted capabilities have an associated score that indicates how much evidence exists to support the prediction. IP addresses, hostnames, and registry keys are all extracted from the malware samples by running a regular expression on the printable strings in the malware samples. Cynomix collects these attributes through static analysis; dynamic analysis is planned for future versions.

4. EXAMPLE ANALYSIS

To give an example of how SEEM can be used to analyze a malware sample, we chose a malware sample from the Mandiant APT1 dataset as the focal sample to analyze (SHA1 hash 05b8c1f0a4cebf2b05d41b904d4e93660acd9976). There were 31 other samples similar to the focal sample, which defined the comparison samples used by SEEM. This example illustrates a real-world use case: After a hacking incident, analysts need to analyze all suspicious files found on the infected computers and will begin with one sample to analyze.

By looking at the histograms and the Venn diagram list we noticed that none of the samples had any images, tags, IP addresses, hostnames, or registry keys. This allowed us to quickly focus on comparing capabilities, strings, DLLs, and functions.

Initially we decided to focus on the most similar samples. First we adjusted the capabilities histogram slider to keep the two bars of most similar samples, then the strings histogram slider to filter out samples that do not have similar capabilities and strings (see Figure 7). By looking at the Venn diagram list we saw that five samples are extremely similar to the focal sample. We also saw four other samples that are very similar to the focal sample with the same DLL imports but a few differences in capabilities, strings, and functions. All of these samples are also from the APT1 dataset. We noticed that seven samples have the "plays audio" capability, which we thought was interesting, so we applied the tag "look for audio resources" to annotate those seven samples about additional analysis we would like to perform outside of SEEM. We also applied the tag "analyze with 05b8c1f0a4cebf2b05d41b904d4e93660acd9976" to the five most similar samples so that anyone who analyzes those samples might benefit from reusing the analysis performed on the focal sample.

Then we started looking at the samples that had different strings by resetting the capabilities histogram slider and by changing the strings histogram slider to filter out samples with very similar strings. We sorted the strings relationship matrix by strings that occur in the focal sample to see which strings were different in the comparison samples and we noticed that some string differences were in spelling. So then we sorted the strings relationship matrix by lexicographical order and noticed lots of misspellings and typographical errors (see Figure 7). Many differences between the malware samples were due to spelling and typographical changes. This could indicate new malware versions with fixes.

Using SEEM we quickly identified the most similar samples and understood how similar they are to the focal sample. We also noticed that many differences in printable strings were due to different spellings of the same strings. Using these insights, we annotated the samples to help us with the malware analysis triage process and to see evidence that many of these samples may be alternate versions of each other.

5. EVALUATION

To evaluate SEEM, we conducted a user evaluation so we can (1) better understand SEEM's usability, (2) evaluate whether users can use SEEM to understand similarities and difference's between malware, and (3) identify future work for improving SEEM.

5.1 Participants

We recruited five participants who have malware analysis experience. The participants have 2 to 14 years experience in cyber security. All participants were male, and their ages ranged from 29 to 50 years old. Our goal was to recruit enough participants to discover the primary usability concerns, and research suggests that five participants is sufficient [19].

5.2 Experiment Design

Evaluations were performed on a 15-inch MacBook Pro with a Retina display, 16 GB of RAM, and a 2.6 GHz Intel Core i7 processor. We ran SEEM inside Chrome browser version 35. We used the APT1 dataset described in Section 4, with sample 05b8c1f0a4cebf2b05d41b904d4e93660acd9976 chosen as the focal sample the participants should analyze.

5.3 Procedure

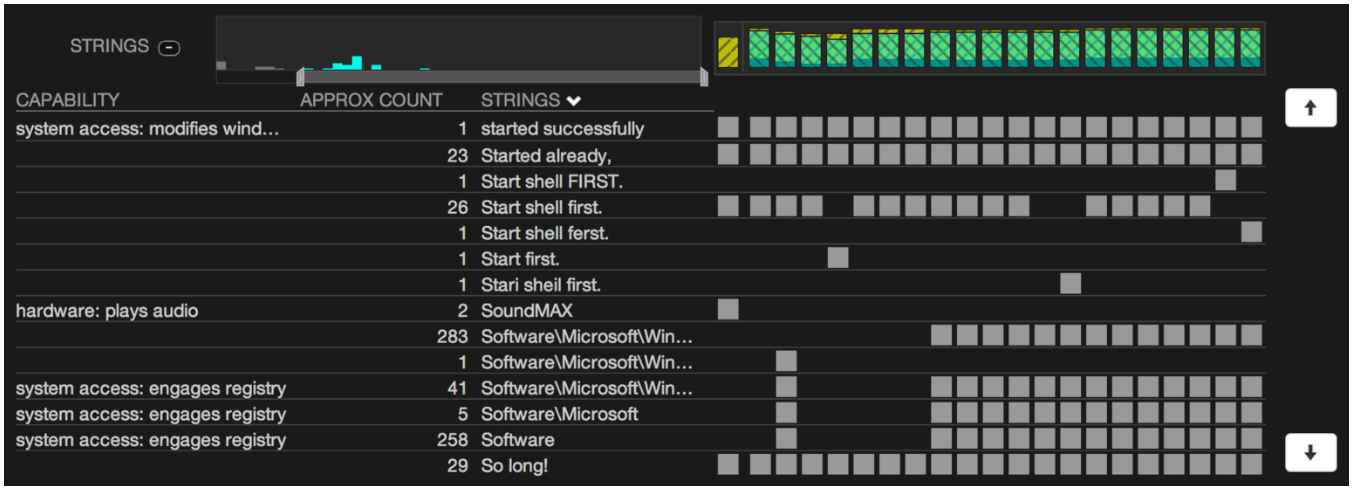
All participants took part in the evaluation in separate sessions, each lasting about 40 minutes. We offered participants a $15 USD gift card to Starbucks as compensation. Each session consisted of three consecutive phases: A training phase, an experimental phase, and a debriefing interview. Prior to the session, each participant had seen one to three demonstrations of SEEM.

During the training phase, participants read descriptions that explained SEEM's visual components and interactive widgets, then participants performed four training tasks. Participants were not allowed to proceed to the experimental phase until they correctly completed all training tasks. In the experimental phase, participants began by performing three tasks, and then participants freely explored the dataset for 10 minutes. Participants were asked to think aloud during the experimental phase. Afterwards, participants provided verbal feedback in the debriefing phase. The order of the tasks in the training phase was fixed, but the order during the experimental phase was different for each participant to reduce ordering bias.

5.4 Results

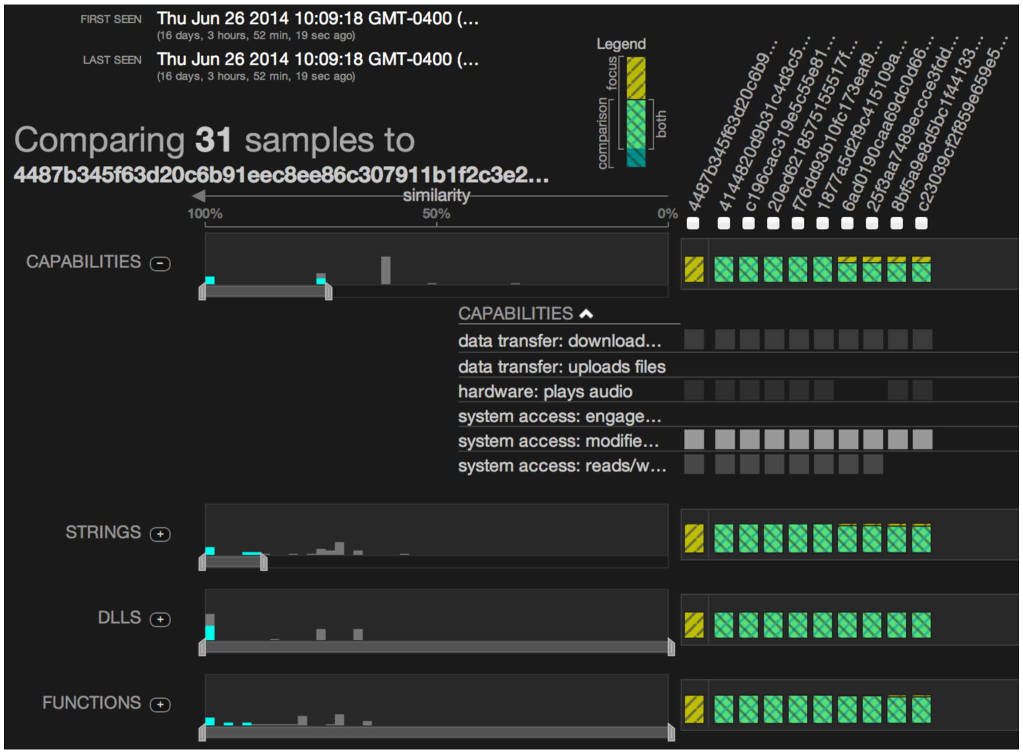
5.4.1 Timed Tasks

For Task 1 ("How many samples have the same capabilities as the focal sample?"), all participants correctly answered the question in 8 to 26 seconds. One participant answered the question by visually inspecting the Venn diagrams and correctly identifying how many samples have the same capabilities. The rest of the participants adjusted the range slider on the capabilities category to filter out the samples without the same set of capabilities as the focal sample and then counted the remaining samples that were not filtered out.



CAPABILITY	APPROX COUNT	STRINGS
system access: modifies wind...	1	started successfully
	23	Started already,
	1	Start shell FIRST.
	26	Start shell first.
	1	Start shell ferst.
	1	Start first.
	1	Stari sheil first.
hardware: plays audio	2	SoundMAX
	283	Software\Microsoft\Win...
	1	Software\Microsoft\Win...
system access: engages registry	41	Software\Microsoft\Win...
system access: engages registry	5	Software\Microsoft
system access: engages registry	258	Software
	29	So long!

Figure 6. The relationship matrix showing malware samples that have five different variations on the string "Start shell first."


FIRST SEEN Thu Jun 26 2014 10:09:18 GMT-0400 (...
(16 days, 3 hours, 52 min, 19 sec ago)
LAST SEEN Thu Jun 26 2014 10:09:18 GMT-0400 (...
(16 days, 3 hours, 52 min, 19 sec ago)

Comparing **31** samples to
4487b345f63d20c6b91eec8ee86c307911b1f2c3e2...

CAPABILITIES
data transfer: download...
data transfer: uploads files
hardware: plays audio
system access: engage...
system access: modifie...
system access: reads/w...

Figure 7. Analying a sample from the APT1 dataset. Five samples are extremely similar and bear analysis in a lower-level tool such as IDA Pro to determine their precise differences. Four other samples have the same DLL imports as the focal sample and also similar sets of capabilities, strings, and function calls.

Task 2 asked participants to answer whether or not the samples that import the same DLLs as the focal sample also have the same predicted capabilities and printable strings as the focal sample. Participants took 15 seconds to 1 minute 8 seconds to answer this question. All participants answered correctly.

Two participants performed this task by adjusting the range slider on the DLLs category to filter our samples without the same DLLs, and then inspecting the histograms on the capabilities and strings categories to see that some samples do appear similar but many samples are also very dissimilar to the focal sample in their capabilities and strings.

One participant used the DLLs histogram slider to filter out samples with dissimilar sets of DLLs, then looked at the Venn diagrams for the remaining samples to see that many samples have additional capabilities and strings not present in the focal sample, and some also lack capabilities and strings present in the focal sample.

Another participant began by adjusting the DLLs histogram slider as the other participants did, and then adjusting the capabilities and strings histogram sliders to filter out samples with dissimilar sets of capabilities and strings. The participant noticed that these last two filtering actions reduced the number of samples in view, and correctly concluded that many of the comparison samples do not share the same set of capabilities and strings as the focal sample.

The other participant also used the DLLs histogram slider to filter out dissimilar samples, and then opened the capabilities and strings relationship matrices to see whether the samples had the same capabilities and strings.

Task 3 caused some confusion in the way the question was worded ("Of the samples that do not have the same set of DLLs as the focal sample, describe how the DLLs are different in those samples."), which may have caused the task completion times to be longer (times ranged from 48 seconds to 2 minutes and 33 seconds). At first one participant thought the question was asking about DLL function calls before he asked for clarification.

All but one participant used the DLLs histogram slider to filter out samples that have the same DLLs as the focal sample, and then opened the DLL relationship matrix to examine which samples import which DLLs.

The other participant did not use the range slider and immediately started using the DLL relationship matrix while flipping from the first page of samples to the second page to examine all of the samples and using the Venn diagrams to visually check which comparison samples have the same set of DLLs as the focal sample.

5.4.2 Free Exploration

Participants often used the predicted capabilities to get a general idea of the samples, indicating that users are interested in automatic capability detection.

The participants made several observations about the data while exploring it with SEEM. Many of the observations were made possible because SEEM displays all of the samples' attributes simultaneously, allowing users to see all of the attributes of the samples within the cluster.

One participant discovered the same five very similar samples we noted in Section 4. He used the range sliders to filter out samples that are not similar. He also hovered over the sample names to get additional information, and found that all the samples have the same file size, providing further evidence that the samples are extremely similar. This is a very important observation that we were hoping users could make by using SEEM.

All five participants noticed that none of the samples had any associated images, tags, hostnames, IP addresses, or registry keys. They accomplished this by opening the relationship matrix for each category and seeing that it was empty. This is the same observation we made in Section 4.

One participant noticed several strings that he felt indicated that the samples were correctly clustered together, such as unique looking error messages and text with misspellings and grammatical errors.

All participants looked at the DLLs, function calls, and strings categories to guess at the capabilities of the focal sample and comparison samples. Participants made several other observations about the capabilities of the samples, either from the capabilities category or by inferring from the DLLs, function calls, and printable strings categories: One participant commented that the predicted capabilities seem like standard malware capabilities. One participant noticed that the focal sample has the capability "modifies Windows services" along with all of the comparison samples. That participant, as well as another participant, found several printable strings and function calls that provide evidence that those samples modify Windows services.

While examining the DLLs category, one participant commented that it was unusual that none of the samples appear to import winsock.dll despite the fact that some samples import urlmon.dll and some samples have the "downloads files" predicted capability. This is something he said he would want to explore deeper if he were to conduct further analysis.

While looking at the comparison samples, one participant noticed a comparison sample that is similar to the focal sample but has an additional predicted capability: uploads files. The participant found this interesting, and suggested that the next step in a deeper analysis would be to see if that comparison sample communicates with the focal sample to exfiltrate data from the victim's computer.

5.4.3 Debriefing

All participants appreciated SEEM's design and were very positive about its usability and utility, with comments such as "[the developers] have done a good job," and "I feel like [SEEM] is pretty useful." Two participants also said that they would like to use SEEM in conjunction with a disassembler like IDA Pro to help direct where to look in the samples.

Two of the participants mentioned that they especially liked the range sliders because it helped the participants narrow their focus to samples within a given similarity range.

One participant commented that SEEM would be very helpful for analyzing a sample that is similar to previously analyzed samples.

Participants made extensive use of the relationship matrix, but made several requests to improve it. Three participants requested some way to search the relationship matrices with some sort of plain text or regular expression search to quickly find certain printable strings or function calls. Two participants felt like they had trouble navigating the relationship matrix and wanted a better way of tracking how big the matrix was and how far up or down they had scrolled. Two participants also wanted additional functionality to filter and sort the relationship matrix.

Finally, one participant suggested functionality we had already be considering: the ability for users to arbitrarily specify which samples to compare to the focal sample. This would support the use case where users have a sample from an intrusion set that they want to compare to other samples from the intrusion set, but none of the samples are similar enough to be clustered together.

5.5 Discussion

All participants were able to complete the tasks successfully, confirming that novice users can learn SEEM and use it to answer analysis questions after only a few minutes. SEEM's flexible design allowed users to answer questions in a variety of ways, as indicated by the variety of approaches successfully used by the participants in Tasks 2 and 3.

Participants appreciated the range sliders because it allowed them to quickly focus their analysis on samples within a given similarity range, or combine filters across categories.

Participants confirmed the samples belonged in a cluster together, and thought SEEM would be very helpful to identify similarities between the focal sample and previously analyzed samples. This confirms that our design supports our initial use case.

The participants also suggested some enhancements during the debriefing session. Participants used the relationship matrix to investigate many of their questions, but they asked for a few improvements, such as free-text search, among various other usability improvements to the relationship matrix.

At the time of publication, we addressed some of the participants' feedback by adding text to indicate how many rows are in the relationship matrix, and by adding the ability to arbitrarily specify samples to use as the comparison samples.

6. CONCLUSION

This paper presents SEEM, a novel, scalable visualization tool for simultaneously comparing a large corpus of malware across multiple sets of attributes. In this paper we show how SEEM can be used to analyze a malware sample from the APT1 dataset to demonstrate how SEEM can be used to generate insight about malware. We also conducted a user evaluation that confirms that novice users can quickly learn and use SEEM to generate insight about a malware sample. The user study also revealed some features that users would like to see in future versions of SEEM.

Future work includes incorporating the user evaluation feedback: Adding search functionality to the relationship matrix, better sorting in the relationship matrix to show what is in common and different between samples, and the ability to visually bookmark interesting samples to help users find them.

We believe SEEM is a general approach for comparing categories of elements between sets. For example, comparing Twitter users and their sets of hashtags, followers, and followees, or comparing stores and the different products they sell broken down by the product categories.

Our main contributions are 1) a scalable system for comparing large, categorical subsets of attributes, and 2) a user evaluation of SEEM and an example usage scenario showing how this system can be used to analyze malware samples.

7. ACKNOWLEDGMENTS

This work was funded by the Defense Advanced Research Projects Agency (DARPA) award FA8750-10-C-0169. 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. We thank our user study participants for their time and insight, and Konstantin Berlin and David Slater for valuable feedback and discussion.

8. REFERENCES

[1] Alper, B., Riche, N.H., Ramos, G. and Czerwinski, M. 2011. Design study of LineSets, a novel set visualization technique. *TVCG*. 17, 12 (Dec. 2011), 2259–67.

[2] Alsallakh, B., Aigner, W., Miksch, S. and Hauser, H. 2013. Radial sets: interactive visual analysis of large overlapping sets. *TVCG*. 19, 12 (Dec. 2013), 496–505.

[3] Bendix, F., Kosara, R. and Hauser, H. 2005. Parallel sets: visual analysis of categorical data. *Symposium on Information Visualization* (2005), 133–140.

[4] Bertin, J. 1981. *Graphics and Graphic Information Processing*. de Gruyter.

[5] Briones, I. and Gomez, A. 2008. Graphs, Entropy and Grid Computing: Automatic Comparison of Malware. *Virus Bulletin* (2008), 1–12.

[6] Chow, S. and Ruskey, F. 2004. Drawing area-proportional Venn and Euler diagrams. *Graph Drawing* (2004), 466–477.

[7] Collins, C., Penn, G. and Carpendale, S. 2009. Bubble sets: revealing set relations with isocontours over existing visualizations. *TVCG*. 15, 6 (2009), 1009–1016.

[8] Conti, G., Dean, E., Sinda, M. and Sangster, B. 2008. Visual reverse engineering of binary and data files. *VizSec* (2008), 1–17.

[9] Dinkla, K., van Kreveld, M.J., Speckmann, B. and Westenberg, M. a. 2012. Kelp Diagrams: Point Set Membership Visualization. *Computer Graphics Forum*. 31, 3pt1 (Jun. 2012), 875–884.

[10] Domas, C. 2012. The Future of RE: Dynamic Binary Visualization. *Derbycon* (2012).

[11] Freiler, W., Matković, K. and Hauser, H. 2002. Interactive visual analysis of set-typed data. *TVCG*. 14, 6 (2002), 1340–1347.

[12] Gove, R., Bergamo, G., Saxe, J., Long, A. and Gold, S. 2014. Cynomix: Multi-Resolution Visualization of Malware at Scale for Insight and Triage. *Malware Technical Exchange Meeting* (2014).

[13] Han, K., Lim, J.H. and Im, E.G. 2013. Malware analysis method using visualization of binary files. *Research in Adaptive and Convergent Systems* (2013), 317–321.

[14] Kestler, H.A., Muller, A., Kraus, J.M., Buchholz, M., Gress, T.M., Liu, H., Kane, D.W., Zeeberg, B.R. and Weinstein, J.N. 2008. VennMaster: Area-proportional Euler diagrams for functional GO analysis of microarrays. *BMC Bioinformatics*. 9, 67 (2008).

[15] Kim, B., Lee, B. and Seo, J. 2007. Visualizing Concordance of Sets. *Interacting with Computers*. 19, 5-6 (2007), 630–643.

[16] Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R. and Pfister, H. 2014. UpSet: Visualization of Intersecting Sets. *TVCG*. (2014).

[17] Meulemans, W., Riche, N.H., Speckmann, B., Alper, B. and Dwyer, T. 2013. KelpFusion: a hybrid set visualization technique. *TVCG*. 19, 11 (Nov. 2013), 1846–1858.

[18] Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B.S. 2011. Malware images: visualization and automatic classification. *VizSec* (2011).

[19] Nielsen, J. 1993. A mathematical model of the finding of usability problems. *Proceedings of the INTERACT'93 and CHI'93*. (1993), 206–213.

[20] Quist, D.A. and Liebrock, L.M. 2009. Visualizing compiled executables for malware analysis. *VizSec* (2009), 27–32.

[21] Saxe, J., Mentis, D. and Greamo, C. 2012. Visualization of shared system call sequence relationships in large malware corpora. *VizSec* (New York, New York, USA, 2012), 33–40.

[22] Saxe, J., Turner, R. and Blokhin, K. 2014. CrowdSource: Automated Inference of High Level Malware Functionality from Low-Level Symbols Using a Crowd Trained Machine Learning Model. *MALCON* (2014).

[23] Stefaner, M. and Muller, B. 2007. Elastic lists for facet browsers. *DEXA* (2007), 217–221.

[24] Trinius, P., Holz, T., Gobel, J. and Freiling, F.C. 2009. Visual analysis of malware behavior using treemaps and thread graphs. *VizSec* (2009), 33–38.

[25] Verroust, A. and Viaud, M. 2004. Ensuring the drawability of extended Euler diagrams for up to 8 sets. *Diagrammatic Representation and Inference* (2004), 128–141.

[26] Wilkinson, L. 2012. Exact and approximate area-proportional circular Venn and Euler diagrams. *TVCG*. 18, 2 (Feb. 2012), 321–31.