

MalwareVis: Entity-based Visualization of Malware Network Traces

ABSTRACT

This paper presents MalwareVis, a utility that provides security researchers a method to browse, filter, view and compare malware network traces as entities.

Specifically, we propose a cell-like visualization model to view the network traces of a malware sample's execution. This model is an intuitive representation of the heterogeneous attributes (protocol, host IP, transmission size, packet number, duration) of a list of network streams associated with a malware instance. We encode these features into colors and basic geometric properties of common shapes. The list of streams are organized circularly in a clock-wise fashion to form an entity. Our design takes into account the sparse and skewed nature of these attributes' distributions and proposes mapping and layout strategies to allow a clear global view of a malware sample's behaviors.

We demonstrate MalwareVis on a real-world corpus of malware samples and display their individual activity patterns. We show that it is a simple to use utility that provides intriguing visual representations that facilitate user interaction to perform security analysis.

Categories and Subject Descriptors

H.5 [Information Interfaces and Presentation]: User Interfaces; C.2.0 [Computer-Communication Networks]: General—*Security and Protection*; I.3.8 [Computer Graphics]: Applications

General Terms

Visualization, Security

Keywords

Malware Analysis, Network Behavior Visualization

1. INTRODUCTION

Malware remains at the forefront of most security threats on the Internet. Nearly every existing malware instance relies on network communication to facilitate its attacks, such as communicating with its command and control (C&C) server, sending spam emails or operating in a fast-flux service network. Therefore, the set of network traces are commonly used to characterize a malware sample's behavior. As an essential technology that combats illegal income for the attackers, malware analysis has been playing a crucial role in supplying signatures to investigation and detection systems.

There is a growing need for visual representations in the workplace of malware analysis: new instances of "in the wild" malware samples are collected daily. In a typical analysis scenario, each malware instance is run on a controlled environment where its host-level and network-level activities are captured as host and network traces. In order to keep track of each sample, security researchers maintain large databases of network traces. Due to their sheer size, it is a daunting task to look into the details of each sample.

Current malware analysis approaches pose several challenges to the analytic process itself which complicate visualization design. First, the network traces generated from the controlled environment contain information irrelevant to understanding most malware's behavior (e.g. DHCP, ARP, SSDP packets and local host IP). The ambition to visualizing them all not only complicates the design, but also causes confusion to the analysts. Since many of these are known *a priori*, we filter irrelevant attributes during preprocessing. Second, even though we have discarded the irrelevant content, the analyst can easily lose the big picture of malware activities by focusing on individual packets. To handle this, we aggregate packets into streams separated by protocol. Specifically, we focus on DNS and TCP streams as they are often used by malware for communication. Third, each stream can have drastically different numbers of packets and such an uneven distribution presents in nearly every attribute space (e.g. total size of transmission, starting time, duration). Linearly mapping these attributes not only consumes screen real-estate, but also draws the viewer's attention away from those small but potentially important streams. Last, but most importantly, showing all relevant attributes in a single view, a vivid graphical representation of individual malware behaviors, requires an original design. In the absence of an intuitive presentation of the set of network traces, malware behavior analysis can still be tedious.

Time	Source	Destination	Protocol	Length	Info
18:14:48.82770	192.168.11.15	64.94.12.123	DNS	76	Standard query A time.windows.com
19:14:48.85518	64.94.12.123	192.168.11.15	DNS	528	Standard query response CNAME time.microsoft.akadns.net A 2
20:17:49.00605	192.168.11.15	207.46.232.182	NTP	90	NTP Version 3, client
21:17:49.56065	192.168.11.15	207.46.232.182	NTP	90	NTP Version 3, symmetric active
22:19:55.4173	192.168.11.15	192.168.11.1	DHCP	356	DHCP Request - Transaction ID 0xe2f238169
23:19:55.4173	192.168.11.1	192.168.11.1	DHCP	356	DHCP Response - Transaction ID 0xe2f238169
24:18:884463	c6:8f:f3:89:6c:95	Vmware-f7:be:ad	ARP	42	who has 192.168.11.1 Tell 192.168.11.1
25:19:884481	Vmware-f7:be:ad	c6:8f:f3:89:6c:95	ARP	42	192.168.11.1 is at 0:0:c:29:f7:be:ad
26:24:268288	192.168.11.15	224.0.0.22	IGMP	54	V3 Membership Report / Join group 239.255.255.250 for any sc
27:24:268304	192.168.11.15	224.0.0.22	IGMP	54	V3 Membership Report / Join group 239.255.255.250 for any sc
28:24:355880	192.168.11.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
29:25:054212	192.168.11.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
30:25:054213	192.168.11.15	224.0.0.22	IGMP	54	V3 Membership Report / Join group 239.255.255.250 for any sc
31:25:054223	192.168.11.15	224.0.0.22	IGMP	54	V3 Membership Report / Join group 239.255.255.250 for any sc
32:27:383316	192.168.11.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
33:27:383325	192.168.11.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
34:30:349033	192.168.11.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
35:33:049047	192.168.11.15	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
36:82:094184	192.168.11.15	64.94.12.123	DNS	75	Standard query A www.applian.com
37:83:048429	192.168.11.15	64.94.12.123	DNS	75	Standard query A www.applian.com
38:84:046619	192.168.11.15	64.94.12.123	DNS	75	Standard query A www.applian.com
39:84:179996	64.94.12.123	192.168.11.15	DNS	148	Standard query response CNAME applian.com A 209.237.2.231
40:84:219915	192.168.11.15	192.168.11.15	DNS	148	Standard query response CNAME applian.com A 209.237.2.231
41:84:219915	192.168.11.15	209.237.2.223	TCP	62	netinfo-local > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460
42:84:279752	209.237.2.223	192.168.11.15	TCP	62	http > netinfo-local [SYN, ACK] Seq=1 Ack=1 Win=5840 Len=0
43:84:278393	192.168.11.15	209.237.2.223	TCP	54	netinfo-local > http [ACK] Seq=1 Ack=1 Win=64240 Len=0
44:84:294582	192.168.11.15	209.237.2.223	HTTP	349	GET /flyplayer/version.xml HTTP/1.1
45:84:355407	209.237.2.223	192.168.11.15	TCP	54	http > netinfo-local [ACK] Seq=296 Win=6432 Len=0
46:84:355407	192.168.11.15	209.237.2.223	TCP	54	netinfo-local > http [ACK] Seq=296 Win=6432 Len=0
47:84:355988	209.237.2.223	192.168.11.15	TCP	54	http > netinfo-local [FIN, ACK] Seq=512 Ack=296 Win=6432 Len=0
48:84:355988	192.168.11.15	209.237.2.223	TCP	54	netinfo-local > http [ACK] Seq=296 Ack=513 Win=63729 Len=0
49:84:807111	192.168.11.15	209.237.2.223	TCP	54	http > netinfo-local [ACK] Seq=296 Ack=513 Win=63729 Len=0
50:84:865144	209.237.2.223	192.168.11.15	TCP	54	http > netinfo-local [ACK] Seq=513 Ack=297 Win=6432 Len=0

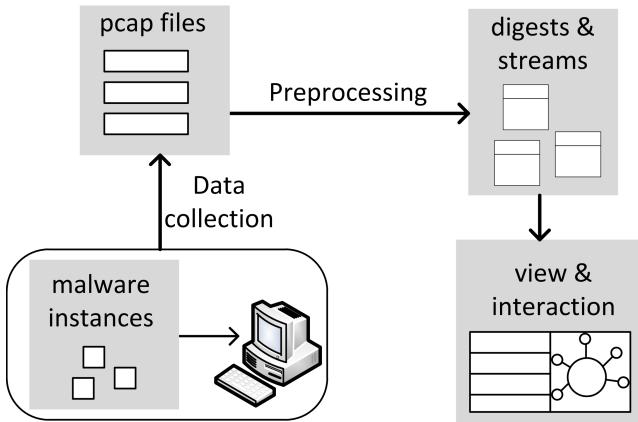


Figure 2: The overview of our malware visual analytic system. The user-end view and interaction rely on three primary modules: data collection, preprocessing and visualization.

the major innovation and advantage of MalwareVis lies in its entity-based design. The idea is to visualize the network traces associated with each malware instance as a whole. It is natural to represent overall the flows associated with each sample as one model for the purpose of identifying potentially interesting patterns. For example, Quist and Liebrock visualize the compiled executables from monitoring malicious program execution [16]. While our work focuses on malware network traces, the purpose is similar—to allow the analysts to identify particular patterns of network traffic. Based on this idea, we design an intuitive visualization model, the *cell view*, to present behaviors of malware network communications.

3. SYSTEM OVERVIEW

We show an overview of our malware visual analytic system in Fig. 2, complemented by the discussion of how malware network data are captured, processed and visualized in this section. As shown in the figure, the process consists of three primary modules: *data collection*, *preprocessing*, and *visualization*.

Data sources. First, we gather a subset of recent malware samples seen in the wild. Samples were collected from honeypots, mail filters, proxy monitors, web crawling, file sharing networks, etc. The testing set consisted of 200 MD5 unique malware instances. We analyze the malware binaries by executing them for three minutes in a virtualized dynamic malware analysis system, similar to previously described systems [6, 1, 17], and record all the network traffic into packet traces as pcap files¹. Precautions were taken to prevent the executing malware from acting maliciously; specifically, we redirected sent email to a spam trap, blocked connections to ports commonly used by exploits and blocked traffic to other local machines.

Preprocessing. Then, we parse the pcap files into a set of appropriately formatted documents which we refer to as malware entities. Each entity is of the form

¹<http://www.tcpdump.org/>

`<Digest D | ArrayList<Stream> S>`

The structure *Digest* consists of the MD5, total number of packets, total size of transmission (including overhead from the headers), total number of streams, and duration. The list of streams are extracted from the protocol information in a pcap file. Specifically, each stream is an entity itself with heterogeneous attribute value types that include host (IP or domain name), protocol (DNS or TCP), number of packets (n), size of transmission (z), relative start time (s), duration (t), and validity. A DNS stream is defined as a DNS request and its corresponding response and a TCP stream begins with a 3-way TCP handshake and concludes with successful or unsuccessful termination.

Visualization. The visualization module generates table views and cell views. Given a set of entity records, the user browses the table of digests and select one or several target entities. Then the application generates a stream table and a cell view for each selected entity. Our attribute mapping and layout algorithms run in real-time for most of the malware entities supplied. In a typical visual analysis scenario, the user interacts with the cell view by selecting a ‘cilium’ that represent a stream of interest. Details of the stream, such as its host IP and country name, are displayed at the disk panel located at the center of the cell view. The user can adjust the layout parameters for the circular timeline interactively so as to improve the clarity of the set of cilia (Sec. 4.4).

4. CELL VIEW IMPLEMENTATION

This section presents our design of the cell view. We first introduce different components of a cell view and their implementations in Sec. 4.1. Then we present in Sec. 4.3 the detailed mapping schemes that encode the various attributes to properties of simple geometric shapes. In Sec. 4.4, we present a layout algorithm inspired from fast tone-mapping technique to avoid cluttered drawing on the timeline.

4.1 Components

A cell view has three basic components: a timeline, a disk panel and a set of cilia.

Timeline. As shown in Fig. 4.1, the timeline is a directional open curve joined by an circular arc and a quadratic Bezier curve. The arc is of radius R , centered on screen and spanning an angle from δ to $2\pi - \delta$, where δ is set to a small positive value ($\frac{\pi}{12}$ in our current implementation) to differentiate the starting point from the end point. The control points of the Bezier curve are specified by δ and R in order to let the curve joins the arc with tangent continuity.

The use of circular timeline is not new (e.g. [3]) and commonly appears as a model in geology and news data visualization for presenting events that are evolving or periodic in nature. Here the motivation is to view malware network traces as an entity. We believe that using a circular timeline instead of a straight timeline better promotes this impression.

Disk Panel. The circular timeline delimits a round area which is used to display the primary key of the malware instance or details of a user-selected stream. In the viewing mode, the unique MD5 value is shown on the disk panel. In

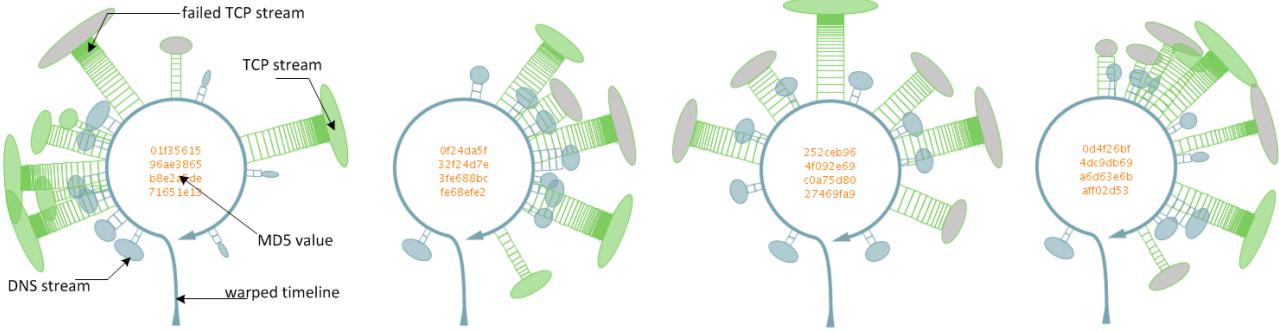


Figure 3: We show the cell views generated by MalewareVis for 4 malware instances. From left to right, the views each represent 5661, 313, 3527, 1043 records in their corresponding pcap files. We annotate on the first view (the detailed mapping scheme is explained in Sec. 4); the second view and the fourth view show similar behaviors in terms of communication patterns. Different behaviors associated with different samples are intuitive to see.

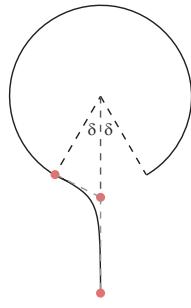


Figure 4: The timeline is an open curve joined by an arc and a quadratic Bezier curve: the red dots are the control points of the Bezier curve.

the interaction mode, the disk panel display attribute values of the user-selected stream. It is possible to show the details of a stream near the cilium representing this stream, or in another pop-up panel; nevertheless, using the disk panel is a more economical choice in terms of minimizing the total screen space necessary for the view.

Cilium. Each cilium represents a stream of network communication. A cilium has a *stem* and a *head*. The stem of the cilium is a quad strip. Each quad represents a packet and hence the longer the stem the larger is the number of packets being transmitted. The width of the stem is correlated with the duration of the stream. The head of the cilium is an ellipse whose width encodes the size of transmission. Note that in a real-world scenario, the number of packets, duration and size are highly correlated with each other. Hence, the width of a cilium's head commonly scales with the length of its stem. Outliers, such as a ‘short’ cilium with a ‘wide’ head or a ‘tall’ cilium with a ‘slender’ stem , might be indicative of interesting behavior.

4.2 Drawing

All geometric dimensions (timeline radius, cilium height, width, etc.) are relative to the screen size, hence the cell

view can scale without loss of quality. Each cilium object is associated with a drawing function (`drawCilium()`) that displays the cilium with transparency in the local coordinate system. The transformation matrix stack is used to draw a cilium in the global coordinate system:

```
pushMatrix();
translate(P.x, P.y);
rotate(a);
drawCilium();
popMatrix();
```

where P is the contact point of the cilium on the timeline determined the plan angle a and the timeline’s center O and radius R .

4.3 Attribute Mapping

As expressed earlier, we use nonlinear mapping functions to transfer attribute values to geometric properties since linear mapping may require unbounded area. The design guideline of each mapping function is to limit each geometric dimensions to be within a lower bound and a upper bound and is a monotonically increasing function of the input attribute value.

Mapping number of packets n to cilium stem height h : The stem of the cilium is a quad strip where each quad denotes a packet. The cilium height h is the sum of the set of quad heights $\{h_i\}$, which forms a geometric sequence with r as the scale factor:

$$r = \frac{h_{max} - h_{min}}{h_{max}}$$

For example, if a stream has n packets, the height of its stem is

$$h = h(n) = \sum_{i=1}^n h_i = h_{max}(1 - r^n) \quad (1)$$

Hence, h is bounded by h_{min} and h_{max} , and $h(n)$ is monotonically increasing in n .

Mapping duration t to cilium stem width w : The width w of the stem is mapped from the duration t of the stream by the following transfer function

$$w = w(t) = w_{max} - \frac{w_{max} - w_{min}}{t + 1} \quad (2)$$

This would limit the width of the stem to w_{max} and $w(t)$ is a increasing function of t .

Mapping size of transmission z to cilium head width w : Similarly, the width l of the oval head of the stem is computed from the size of the stream, denoted by z :

$$l = l(z) = l_{max} - \frac{l_{max} - l_{min}}{\log(z)} \quad (3)$$

As the sizes of transmission of different streams can be drastically different (ranges from 10^2 to 10^6 bytes), we use logarithm of z in the mapping function.

4.4 Layout on Timeline

To strengthen the visualization as a cell entity, we orient the set of streams around the circular timeline in a clockwise fashion. Each stream has a starting time, which we denote as s_i , and the position it appears on the circular timeline is determined by a_i . The relative order and the intermissions of a set of streams are important characteristics of a malware instance's behavior.

Therefore, given a set of starting times $\{s_i\}, i = 1, \dots, k$, we want to compute a set of angles $\{a_i\}$ bounded by the mappable range $[\delta, 2\pi - \delta]$. Assume that $\{s_i\}$ is sorted. s_1 is the smallest and mapped to δ and s_k is the largest and mapped to $2\pi - \delta$. As mentioned earlier, the distribution of $\{s_i\}$ is very uneven because multiple streams of communications may occur in a very short time period. Hence, linearly mapping $\{s_i\}$ to the mappable range may lead to overlappings and visual clutter, as shown in Fig. 4.4 left. In order to improve clarity while preserving the relative order as they appear in $\{s_i\}$, we present a recursive range mapping algorithm in Alg. 1. This algorithm is inspired by fast tone reproduction techniques for visualizing high dynamic range image data [7] based on the fact that the distribution of HDR data tends to be sparse. Here, the set of starting times $\{s_i\}$ is analogous to the raw HDR image data and the mappable range $([\delta, 2\pi - \delta])$ is analogous to the viewable intensity range $([0, 255])$ in the tone-mapped image. The idea of fast tone-mapping is to scale the mapping range according to the input data histogram: the time period that contains more incidences of streams is mapped to a larger range.

Algorithm Tuning. Specifically, $\alpha \in [0, 1]$ is the parameter that specifies the scaling effect: the larger α is, the more equalized the mapping is. If $\alpha = 0$, Alg. 1 is essentially a linear mapping (?), which causes visual clutter due to that multiple streams typically start together. If $\alpha = 1$, Alg. 1 is the equalized mapping where the streams appear uniformly spaced. However, the equalized mapping does not show intermissions, such as a long period of inactivity after the first DNS stream in Fig. 4.4 right.

5. USER INTERACTION

In this section, we present the user interface of MalwareVis. To start, the user can select a set of pcap files (typically a

Alg. 1 Recursive range mapping: $map(i, j, a_{min}, a_{max})$

Input: subarray indices i, j of input array s , mapping range $[a_{min}, a_{max}]$
Result: set angles in output array a

```

 $a_{mean} \leftarrow \frac{a_{min} + a_{max}}{2}$ 
if  $i \geq j$  then
     $a_i \leftarrow a_{mean}$ 
    return
end if
 $s_{mean} \leftarrow \frac{s_i + s_j}{2}$ ,
 $s_{median} \leftarrow s_{(i+j)/2}$ 
 $cut \leftarrow \alpha s_{median} + (1 - \alpha)s_{mean}$ 
 $ci \leftarrow$  index of  $cut$  in array  $s$ 
if  $ci == i$  then
     $a_i \leftarrow a_{min}$ 
     $map(ci + 1, j, a_{mean}, a_{max})$ 
    return
end if
if  $ci == j$  then
     $a_j \leftarrow a_{max}$ 
     $map(i, ci - 1, a_{min}, a_{mean})$ 
    return
end if
 $map(i, ci - 1, a_{min}, a_{mean})$ 
 $map(ci, j, a_{mean}, a_{max})$ 

```

corpus of malware network traces) and MalwareVis parses them into a collection of entities (Fig. 2). Parsing and pre-processing takes approximately 30 seconds for 95 pcap files with an average file size of 722KB. In addition to reading raw pcap files, the user can also select a set of previously saved entities for quickly initiating the application.

Table View: As shown in Fig. 5, the program then generates table views. The *pcap table* displays the entity digests, each of which is a summary of the communication sessions recorded by a pcap file. The user can browse the list of entities and search or filter them based on MD5 values and duration. The lower part of the table view is the *stream table* that lists sets of streams of a user-selected entity.

Single Cell View: After the user selects an entity, she can create a cell view of the entity by clicking the “single” button in the cell view control panel. A cell view appears nearly instantaneously. In the cell view, the user can interact with a string of cilia: the user can select a cilium by clicking the oval head of the cilium. Then the disk panel shows the detailed stream information represented by this cilium. It also queries a GeoIP database to identify the country name and flag of the host IP if it is a routable IP address. The user can also select from the stream table and its corresponding stream and cilium are highlighted in both views.

Timeline: The user can change the value of α in the timeline mapping algorithm (Alg. 1) from the slider in the cell view control panel. Notice that when sliding, the cell view redraws the set of cilia at each frame. This creates intermediate frames between the two views of different α values (Fig. 4.4). Therefore, the user can view the layout equalization processing on the timeline as an animation.

Multiple Cell Views: The user can also select multiple

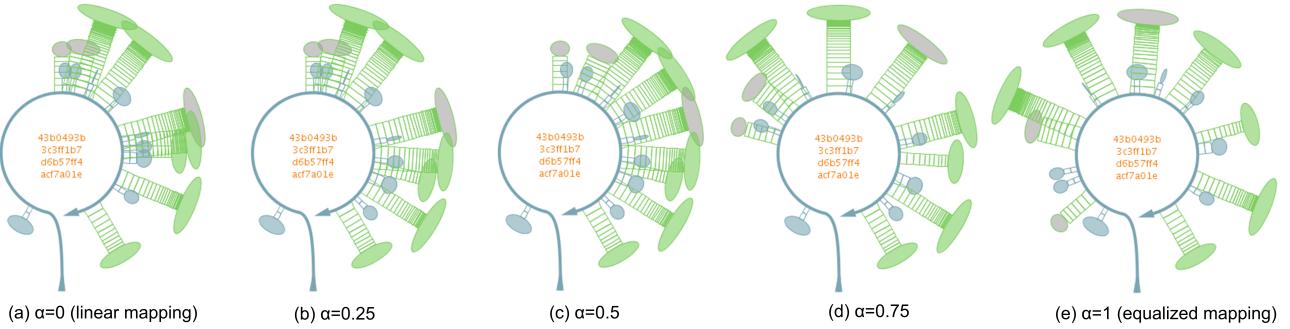


Figure 5: From left to right, we show a sequence of visualizations of the same malware instance with increasing α values: (a) linearly mapping the starting times to angles would result in visual clutter; (d) equalized mapping helps to reduce the overlapping effects.

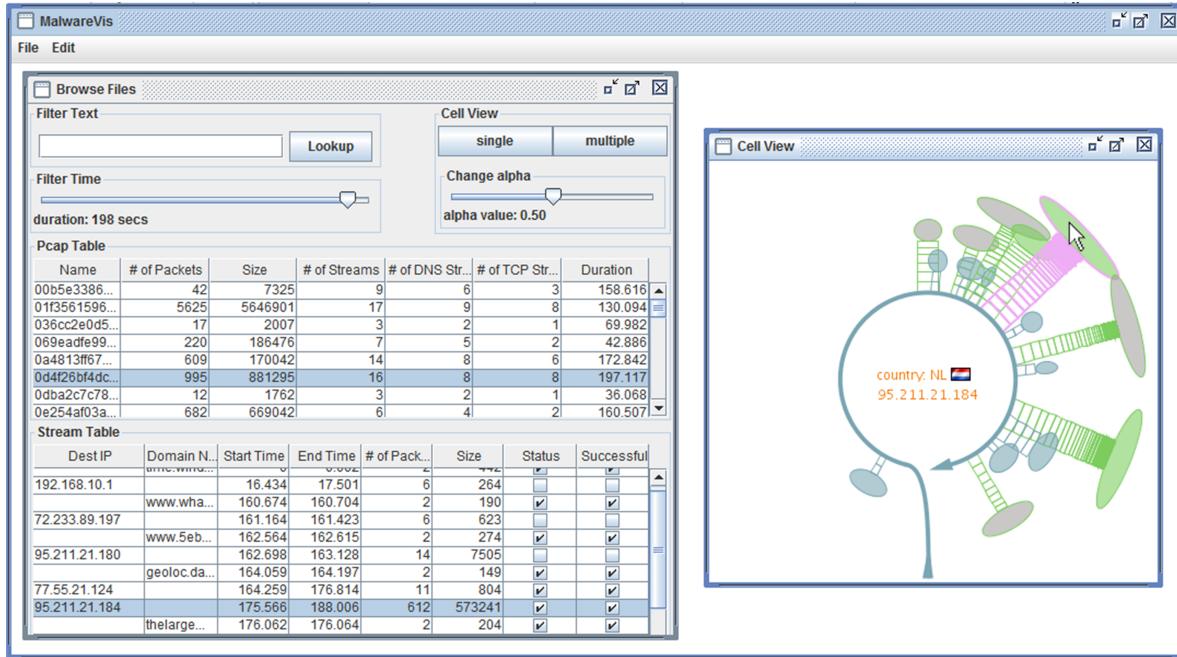


Figure 6: MalwareVis's user interface has a table view that allows the user to browse and filter malware entities and the cell view for graphical representations, interaction and comparison.

malware entities. Then, stacked cell views of a list of the selected entities can be created by clicking the “multiple” button in the cell view control panel. The user can compare the visual representations of multiple malware samples (Fig. 3) and interact with each view as one would interact with a single cell view.

6. REFLECTION AND DISCUSSION

This section discusses the project collaboration process and discusses feedback and ideas for potential future work.

Collaboration

This project has involved close collaboration between students at the graphics&visualization and the information security centers. Initially, the visualization design and the data collection processes were conducted separately. However, to build a visualization model that addresses both utility and aesthetic concerns requires close collaboration among participants. Hence in the development phase, participants met and discussed issues with data processing, visualization and user interaction on a regular basis. Convenient design and collaboration was made possible through shared repository of open source interfacing tools ([12] [10] [11]) and the Processing visualization library [8].

Feedback

We found building MalwareVis to be a rewarding experience: not only the participants with different backgrounds have learned from each other, but the users are intrigued by our design during informal user testing. From their feedback, we have concluded two major advantages of the cell view. The first advantage is that it provides a global view: a single cell view is able to represent up to 6000 flow records in a pcap file without significant visual clutter (Fig. 3). The second advantage is that it provides a graphical user interface that allows the user to analyze different malware behaviors interactively.

Future Development

During our informal user testing, a user asked if there is any relationship between different malware instances and how to visualize them. Binary and n-ary relationships do exist within a corpus of malware instances. For example, malware frequently found in honeypots actually behave as downloaders and simply download, install and execute other malicious binaries. Furthermore, malware is often packed resulting in binaries that are MD5 unique, but exhibit identical behavior. Currently, MalwareVis does not support highlighting these relationships automatically, but we believe this is a possible direction for future work.

7. CONCLUSION

Malware network traces are important characteristics of a malware sample’s behavior. To address the need of a visual representation in the workplace of malware analysis, we have developed a processing and visualization tool, MalwareVis, that allow a security analyst to filter, select, visualize, and compare malware network traces. It reads pcap files, processes them into streams and generates cell views for the user-selected malware entities interactively. In this paper,

we have described our visualization design and demonstrate the advantages of using cell views. In the future, we will explore several areas such as visualizing the relationship among a collection of malware network traces.

8. REFERENCES

- [1] U. Bayer, C. Kruegel, and E. Kirda. Ttanalyze: A tool for analyzing malware, 2006.
- [2] R. Blue, C. Dunne, A. Fuchs, K. King, and A. Schulman. Visualizing real-time network resource usage. In *Proceedings of the 5th international workshop on Visualization for Computer Security*, VizSec ’08, pages 119–135, Berlin, Heidelberg, 2008. Springer-Verlag.
- [3] M. C. Chuah. Dynamic aggregation with circular visual designs. In *Proceedings of the 1998 IEEE Symposium on Information Visualization*, INFOVIS ’98, pages 35–43, Washington, DC, USA, 1998. IEEE Computer Society.
- [4] G. Combs. <http://www.wireshark.org>.
- [5] G. Conti. Rumint: A pvr for network traffic and security visualization, 2006. <http://www.rumint.org>.
- [6] A. Dinaburg, P. Royal, M. I. Sharif, and W. Lee. Ether: malware analysis via hardware virtualization extensions. In *ACM Conference on Computer and Communications Security*, pages 51–62, 2008.
- [7] J. Duan and G. Qiu. Fast tone mapping for high dynamic range images. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04) Volume 2 - Volume 02*, ICPR ’04, pages 847–850, Washington, DC, USA, 2004. IEEE Computer Society.
- [8] B. Fry and C. Reas. <http://www.processing.org>.
- [9] J. R. Goodall, W. G. Lutters, P. Rheingans, and A. Komlodi. Preserving the big picture: Visual network traffic analysis with tnv. In *Proceedings of the IEEE Workshops on Visualization for Computer Security*, VIZSEC ’05, pages 6–, Washington, DC, USA, 2005. IEEE Computer Society.
- [10] <http://code.google.com/p/dpkt>. Python packet creation and parsing library.
- [11] <http://tortoisesvn.net>. Tortoisesvn.
- [12] <http://www.jython.org>. Jython.
- [13] K. Lakkaraju, W. Yurcik, and A. J. Lee. Nvisionip: netflow visualizations of system state for security situational awareness. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC ’04, pages 65–72, New York, NY, USA, 2004. ACM.
- [14] R. Marty. Afterglow: Link graph visualization for network flow data, 2007. <http://afterglow.sourceforge.net>.
- [15] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. Portvis: a tool for port-based detection of security events. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, VizSEC/DMSEC ’04, pages 73–81, New York, NY, USA, 2004. ACM.
- [16] D. Quist and L. M. Liebrock. Reversing compiled executables for malware analysis via visualization. *Information Visualization*, 10(2):117–126, 2011.

- [17] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security and Privacy*, 5(2):32–39, Mar. 2007.