MalwareVis: Entity-based Visualization of Malware Network Traces

Wei Zhuo Yacin Nadji
Graphics, Visualization and Usability Center
Colleage of Computing
Georgia Institute of Technology
wzhuo3, yacin@cc.gatech.edu

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 a 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 is organized circularly in a clock-wise fashion to form an entity. Our design takes into account of the sparse and skew 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, 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 for detection systems.

There is a growing need for visual presentations 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 in 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, network traces often contain large volumes of information (for example, network traffic generated by the operating system). Having to visualize everything in a single view not only complicates the design, but also causes confusion to the analysts. Here we focus on DNS and TCP channels for illustrating Malware samples using these protocols to perform their malicious behavior. To reduce clutter, we do not visualize less common protocols by default. Second, 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, streams can vary drastically in packet size and these uneven distributions present themselves 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. Lastly, showing all relevant attributes in a single, clear view requires an original design. In the absence of an intuitive presentation of sets of network traces, malware behavior analysis will remain tedious.

The goal of this paper is to present a visualization system and a model design that address the above challenges. Our work claims the following contributions:

- **Data collection and preprocessing.** The malware

VizSec '12, Seattle, WA, USA

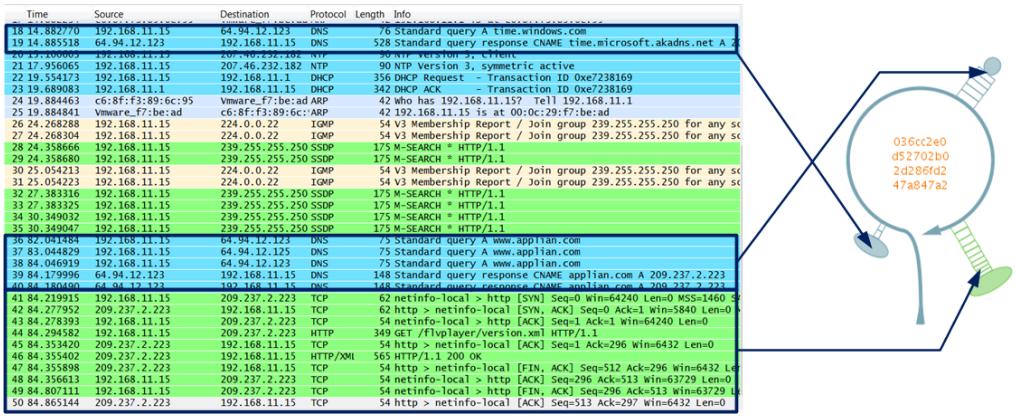


Figure 1: We illustrate our mapping scheme with a simple network packet traces generated by malware (036cc2e0): there are three streams of packets (as highlighted by rectangles in the flow records produced by Wireshark [4]) are mapped to three cilia in the cell view on the right. The user can select a cilium for detailed information about the stream.

network traces were captured by packet sniffer software in a controlled environment. The preprocessing consists of two steps: *pruning* and *bundling*. The raw network data are pruned by preserving protocols known to be used commonly by malware seen in the wild be default. Other protocols can be viewed if an analyst wishes to do so. Then we bundle the packets according to their protocols and hosts so that each stream of communication is semantically clear.

- **Interface and visualization metaphor.** We design an intuitive view, the *cell view*, to represent a set of streams generated by any malware instance. A cell view consists of a *circular timeline*, a *disk panel* to display details-on-demand, and a set of *cilia* oriented clockwise along the timeline to represent the set of streams. This metaphorical structure allows the viewer to select and compare among streams as well as between malware instances.

- **Mapping and layout techniques for sparsely distributed attributes.** As explained earlier, the majority of the communication (in terms of the number of packets, transmission size) takes place within a small percentage of streams. Also, many streams may occur within a short period of time. To cope with these uneven distributions, we use geometric sequences and inverse logarithms to map discrete or continuous attribute values. We also present a recursive angle mapping algorithm that helps produce equalized layout for viewing streams that are cluttered over time.

This paper is organized as follows. Section 2 introduces several related security visualization systems. Section 3 presents an expository account of malware analysis infrastructure for the benefit of readers who are not familiar with the technique. We then introduce our design and implementation in Section4. We apply the visualization model to a corpus of real-world malware samples and Section 5 introduces user interaction scenarios with different components of MalwareVis. Finally, Section 6 discusses and reflects on the project process; Section 7 concludes the paper.

2. PRIOR ART

Most of the related network visualization systems so far has been traffic-centric with explicit focuses on each record of flows. For example, the work of McPherson and Ma [15] presents PortVis, which visualizes large network flow data collected at an Internet gateway. Their system allows interactive timeline filtering and histogram summarization on port activities. NVisionIP [13], introduced by Lakkaraju and his colleagues, focuses on the static visual representation of the IP network traffic in a 2D scatter plot named galaxy view. The other two views (the small multiple view and the machine view) complement the main view by providing bar charts of various flow counts. TNV by Goodall et al. [10] aims to visualize the source, destination and the time attributes in a single view. To allow the user to explore the area of interest in greater detail, TNV can zoom in on traffic through bifocal display techniques. Rumint [5] is another full-fledged, time-oriented network traffic analyzer that captures live traffic and supports playing back the traffic.

Apart from scatter plots histograms and bar charts, linked graphs have become popular representations for network flows. Based on force-directed graph layout techniques, Afterglow [14] is a collection of utilities for generalized communication network visualization. Recently, Blue and his colleagues presented NetGrok [2] that consists of graph and treemap views. In their graph view, the IP layer communications with the local host are visualized as a star network with other hosts organized around based on their IP addresses. Although using graphs for security visualization has increased in popularity, it is unclear to the authors how to incorporate the time-series nature of network flow data into graphs without using a complementary view.

Compared to previous traffic-centric visualization systems, 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 the overall flow associated with each sample as one model for the purpose of identifying potentially interesting patterns. For example, Quist and Liebrock visualize the compiled executables by monitoring malicious program execution [17]. While our work focus on malware

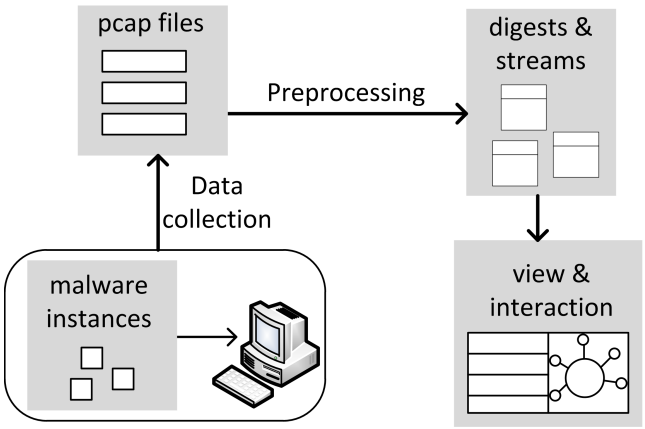


Figure 2: The overview of our malware visual analytic system. The user-end view and interaction rely on three primary modules: data collection, preprocessing and visualization.

network traces, the purpose is similar—to allow the analysts to identify particular patterns of network traffic. Based on this idea, we design a intuitive visualization model, the cell view, to present behaviors of malware network communications.

In the scope of visualizing large volumes of NetFlow data, Fischer et al.[8] presents a visual analytics system for monitoring and visualizing large networks using TreeMap; FloVis [18] presents a suite of visualization tools that display various aspects of multiple individual host behaviors as color-coded time series and show the interactions among hosts and groups of hosts.

3. SYSTEM OVERVIEW

We show an overview of our malware visual analytic system in Figure 2, and discuss how malware network data are captured, processed and visualized in this section. This process consists of three primary steps: *data collection*, *preprocessing*, and *visualization*.

Data sources. First, we gather a corpus of malware samples to use for evaluation purposes. 200 MD5 unique malware samples were randomly selected from malware collected during April 2011. We use multiple sources that provide approximately 6,000 MD5 distinct samples per day taken from low interaction honeypots, web crawlers, spam email filters, and user submissions. We restrict our test set to only malware samples that exhibit network behavior during execution. We analyze the malicious binaries by executing them for three minutes in a virtualized dynamic malware analysis system that runs malware samples in kvm [12] and records all the network traffic into packet traces as pcap files[1]. Our visualization system is not dependent on any particular dynamic analysis system and similar systems described in the literature could serve as drop-in replacements ([1] [6]), Precautions were taken to prevent the executing malware from acting maliciously; specifically, we redirect SMTP traffic to a spam trap, blocked connections to ports commonly used by exploits and blocked traffic to other local machines.

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

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

The structure *Digest* consists of a sample's MD5, the total number of packets, the total size of transmission, the total number of streams, and the packet trace's duration. The list of streams are extracted from the protocol information in the pcap file. Specifically, each stream includes attributes such as host (IP or domain name), protocol (DNS or TCP), number of packets (n), size of transmission (z), offset from the initial start time (s), duration (t), and whether the stream ended successfully or not. 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 parsed and preprocessing entities, the user browses the table of digests and selects one or more targets for analysis. Then, the application generates a stream table and a cell view for each selected entity. Our attributes 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 represents a stream of interest. Details of the stream, such as its host IP and country code, 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. 4.3).

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. 4.2 the detailed mapping schemes that encode the various attributes to properties of simple geometric shapes. In Sec. 4.3 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. The time line is an 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 work) 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 is commonly appeared 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 the interaction mode, the disk panel display attribute values of the user-selected stream. It is possible to show the details

[1] `http://www.tcpdump.org`

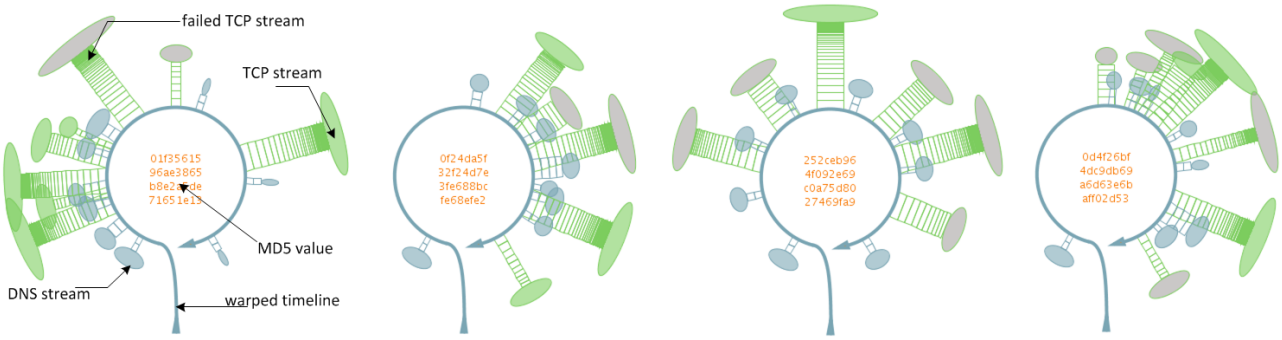


Figure 3: We show the cell views generated by MalewareVis for 4 maleware instances. From left to right, the views each represent 5661, 313, 3527, 1043 records in their corresponding pcap files. We annotate on the first view (the detailed mapping scheme is explained in Sec. 4); the second view and the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

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 communications. A cilium has a *stem* and a *head*. The stem of the cilium is a quad strip. Each quad represent 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 a 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.

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 display the cilium with transparency in the local coordinate system. The transformation matrix stack is used to draw a cilium in the global coordinate system.

4.2 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 dimension 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

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

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

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

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:

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

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.

4.3 Layout on Timeline

To strengthen the visualization as a cell entity, we clockwisely oriented the set of streams around the circular timeline. Each stream has a starting time which we denote as s_i, and the position it appears on the circular timeline is determined by its contacting angle 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 $\{s_i\}, i = 1, ..., k$, 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. 4.3 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. 4.3 right.

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
$map(i, ci - 1, a_{min}, a_{mean})$
$map(ci, j, a_{mean}, a_{max})$

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 corpus of malware network traces) and MalwareVis parses them into a collection of entities (Fig. 2). Parsing and preprocessing takes approximately 30 seconds for 95 pcap files (average file size 722KB). In addition to reading raw pcap files, the user can 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 a list of 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 list a set of streams of a user-selected entity.

Single Cell View: After the user select a 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 in a very small fraction of a second. 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. 4.3). Therefore, the user can view the layout equalization processing on the timeline as a animation.

Multiple Cell Views: The user can also select multiple 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.

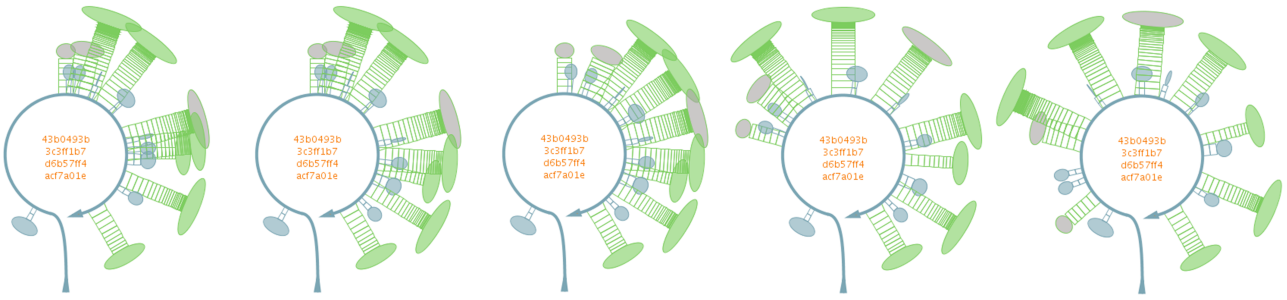
6. REFLECTION AND DISCUSSION

This section discusses the project collaboration process, the feedbacks 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 ([11] [16]) and the Processing visualization library [9].

Feedbacks. 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 as well during informal user testing. From their feedbacks, 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,



(a) α=0 (linear mapping) (b) α=0.25 (c) α=0.5 (d) α=0.75 (e) α=1 (equalized mapping)

Figure 4: 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 results in visual clutter; (d) equalized mapping helps to reduce the overlapping effects.

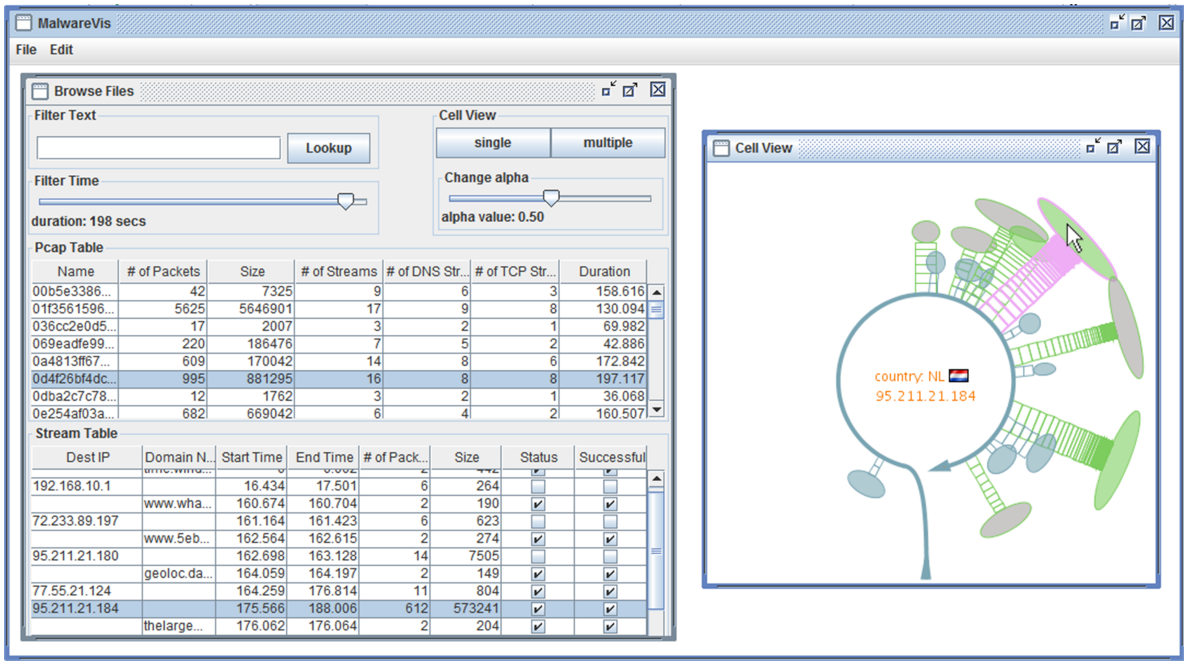


Figure 5: MalwareVis's user interface has a table view that allows the user to browse and filter malware entities and the cell view for graphical representations, interaction and comparision.

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 develop 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. ACKNOWLEDGEMENTS

We would like to thank reviewers for their comments, Prof. Jarek Rossignac and Wenke Lee, Dr. Wes Bethel and Prabhat for their suggestions, PhD student Ikpeme Erete for his help.

9. 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. www.wireshark.org.

[5] G. Conti. www.rumint.org, rumint: A pvr for network traffic and security visualization. www.rumint.org, 2006.

[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] F. Fischer, F. Mansmann, D. A. Keim, S. Pietzko, and M. Waldvogel. Large-scale network monitoring for visual analysis of attacks. In *Proceedings of the 5th international workshop on Visualization for Computer Security*, VizSec '08, pages 111–118, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] B. Fry and C. Reas. www.processing.org.

[10] 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.

[11] Jython. `http://www.jython.org`.

[12] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: The kernel-based virtual machine for linux, 2007.

[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.sourceforge.net, afterglow: Link graph visualization for network flow data. afterglow.sourceforge.net, 2007.

[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] P. packet creation and parsing library. `http://code.google.com/p/dpkt`.

[17] D. Quist and L. M. Liebrock. Reversing compiled executables for malware analysis via visualization. *Information Visualization*, 10(2):117–126, 2011.

[18] T. Taylor, D. Paterson, J. Glanfield, C. Gates, S. Brooks, and J. McHugh. Flovis: Flow visualization system. In *Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security*, CATCH '09, pages 186–198, Washington, DC, USA, 2009. IEEE Computer Society.