Visualization Techniques for Malware Behavior Analysis

André R. A. Grégio[a,b], Rafael D. C. Santos[c]
[a]Information Technology Research Center (CTI/MCT), Campinas, SP, Brazil;
[b]University of Campinas (Unicamp), Campinas, SP, Brazil;
[c]Brazilian Institute for Space Research (INPE/MCT), São José dos Campos, SP, Brazil

ABSTRACT

Malware spread via Internet is a great security threat, so studying their behavior is important to identify and classify them. Using SSDT hooking we can obtain malware behavior by running it in a controlled environment and capturing interactions with the target operating system regarding file, process, registry, network and mutex activities. This generates a chain of events that can be used to compare them with other known malware. In this paper we present a simple approach to convert malware behavior into activity graphs and show some visualization techniques that can be used to analyze malware behavior, individually or grouped.

Keywords: Security data visualization, malicious software, dynamic analysis, information systems security

1. INTRODUCTION

There are a majority of computer systems and devices accessing the Internet nowadays, as it provides all sort of services. These systems are subject to malware exposition that can lead them to loss of integrity, confidentiality or availability. Malware can be defined as a set of malicious applications or codes, such as viruses, worms, trojans and bots, whose intent is to attack system in order to disrupt them, steal sensitive, financial information or even to use them as a disguise in other attacks, with directed target or not.

Malware attacks can be logged when they compromise a system in basically two ways: by network dumps related to the network activities performed by a compromised machine and by tracing all actions (relevant for security purposes) this malware executes during the period it exploits a machine and keeps running on the infected system. Those logs can be attained by dynamically analyzing malware samples to obtain its behavioral profile, i.e., running them inside a controlled environment and monitoring its execution, capturing the actions done in the operating system and the network.

Due to the great amount of malware samples and variants arising every day, there are also a massive data generated by the automated analysis of them. This data needs to be stored and, mainly, analyzed in an adequate fashion in order to provide some useful information that can be used in countermeasures, defense or at least in the identification of a menace. However, analyzing this data in its raw form can be impractical to human analysts or can not generate presentable results depending on which tools were used to extract information of it.

An effective way to find relevant information in great amounts of data with several dimensions is seeing pictures corresponding to it[1], i.e., applying visualization techniques. The area of data visualization applied to security is still recent, so we still have a large range of tools and applications to explore, some of them undeveloped. In addition, many problems require ad-hoc solutions that must be integrated to specific systems and, in the majority of cases, there aren't solutions ready to use which attend all needs.

In this paper we propose a simple approach to convert malware behavior into activity graphs and show some visualization techniques that can be used to analyze malware behavior, individually or grouped. The remaining of the paper is divided as follows: in Section 2, we present a visualization review, showing some concepts and techniques in general. In Section 3, we provide a brief summary of some research and development done in the field of security data visualization. In Section 4, we discuss our methodology to capture malware behavior, then to process it in order to apply visualization techniques on it. The results obtained are also shown in this Section. Finally, in Section 5 we present the final considerations about this work.

Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X, edited by Edward M. Carapezza, Proc. of SPIE Vol. 8019, 801905

2. VISUALIZATION REVIEW

2.1 Visualization concepts

Edward Tufte, Professor Emeritus of Political Science, Statistics, and Computer Science of the Yale University, writes in his classic book *The Visual Display of Quantitative Information*[1] that "…modern data graphics can do much more than simply substitute for small statistical tables. At their best, graphics are instruments for reasoning about quantitative information. Often the most effective way to describe, explore, and summarize a set of numbers – even a very large set – is to look at pictures of those numbers. Furthermore, of all methods for analyzing and communicating statistical information, well-designed data graphics are usually the simplest and at the same time the most powerful". This is a very good definition of the role of visualization on data analysis: see "pictures of the numbers" may explain much about what the numbers measure or represent than plain statistical measures.

Visualization is a task easily performed by humans: the human visual system is able to detect patterns, exceptions, trends, relationships among visually noticed objects, even if it's not possible to describe these phenomena in natural language. This ability allied to the use of computers to prepare, organize and visualize data turns possible data exploration in new, efficient and interesting ways.

Data visualization, in particular technical-scientific data such as systems security data, can be used to accomplish several objectives related to analyzing these data[2]:

- **Exploratory Analysis**, using known data without a defined hypothesis about phenomena that can occur in this data. Usually involves the visual search for trends, exceptions, structures, etc. in the data. The result of exploratory analysis may be the hypothesis definition.
- **Confirmation Analysis**, using known data and hypothesis about phenomena related to this data. Through visualization the hypothesis can be accepted or rejected.
- **Presentation**, using the data with an objective to demonstrate these data, data-related phenomena or hypothesis. One should define an adequate presentation technique which allows for easy interpretation.

2.2 Visualization techniques

There are several data visualization techniques, since the simplest and generic ones, such as area, pie, bar, pizza, lines and dots graphics which are usually available in electronic spreadsheets, until more complex and specific ones, such as volume slicing in 3D to present bi-dimensional images used in nuclear magnetic resonance image visualization.

Visualization techniques can be grouped in categories, but some techniques can belong to more than one category and some other can belong to any of them. As there are plenty of visualization techniques we will cover just a subset of them, some based in tutorials and class notes by Daniel Keim[2].

Geometric techniques allows data visualization through transformations (reorganization, projection) of its attribute values. One of the most known geometric technique is the scatter-plot matrix[3], which presents a bi-dimensional matrix with data plots where each combination between two dimensions is represented each one by an exclusive plot. This kind of visualization is shown in Figure 1, that represents a comparison of some suspicious and normal network traffic attributes[4]. In this kind of graphic we can observe direct and inverse correlation between attributes (two by two), attributes values distribution, among other phenomena that can suggest hypothesis about this data.

Another geometric technique that allows visualization of correlations among attributes, patterns and exceptions is named parallel coordinates[5,6]. Here, an axis is created for each attribute, then organized in a parallel and equidistant way and, for each data to be visualized we trace a polygonal line that crosses the axis in the height of the corresponding value. An example is given in Figure 2, using the same data from Figure 1. In this kind of graphic it is possible to infer metrics to separate network traffic which abuses an institution security policy, as we can clearly observe the differences between suspicious and normal traffic in some attributes axis.

Other geometric techniques are Kohonen Self-Organizing Maps (SOM)[7], based on reorganization of multidimensional data in less dimensions, Prosection Views and Hyperslice[2], which are similar to scatter-plots, but give to the user the support to select a data bi-dimensional region in one of the matrix cells and also to drill-down.

Problems can occur when using geometric techniques if there is a huge amount of data to be visualized. Depending on the order in which the lines are drawn there will be overlaps that hide some lines and prevent the visualization of part of data. Techniques such as transparency or jitter can be used to minimize this kind of problem.

Icon-based techniques can represent multidimensional data as icons whose characteristics correspond to data attribute values. One icon-based technique is shape coding[8], where the multidimensional values are mapped in a small retangular graphic which is used as a marker in a bi-dimensional graphic whose dimensions can be spacial or temporal. This kind of technique helps in visually identifying exceptions. A similar technique is used in VisDB application[9], which uses colors and spiral organized icons. Icon-based techniques can allow quick visual interpretation through comparing among other icons in order that one identify similarities and exceptions, but the mapping of attribute values into icons requires constant use of legends that can confuse the user.

Pixel-based techniques are similar to icon-based as both use small sets in a spacial arrangement to represent multidimensional values and to organize this sets in greater arrangements that may represent the dataset spacial/temporal dimensions. The attributes values are depicted as colored pixels, usually in a color map that helps in the identification of equal and different values. Other pixel-based techniques variants involve temporal ordering of pixels using lines and different curves to fill the bi-dimensional space, such as Peano-Hilbert, Z-Curves, etc.

Hierarchical techniques partition multiple dimensions of data in subspaces that can be visualized in 2 or 3 dimensions. One of most known hierarchical technique is treemap[10], in which a bi-dimensional area is filled with small polygons where their area is proportional to importance regarding the visualization. Other features such as colors or textures can be added to denote additional information. Another well-known hierarchical technique is the dendogram, a kind of graphic that divides a relatively small set of data in partitions and shows groupings and distances among represented data.

Graph-based techniques show relations (edges) among objects (vertices). This graphic representation can be used to present patterns and values associated to relationships, such as proximity, intensity, correlation. There are several visualization techniques that use graphs and the study of algorithms to position vertices and edges in a way that is better to visualize an entire data set is an active research area. In Section 4 we present an example related to graph-based visualization.

Tridimensional techniques make use of 3D graphics to visualize data that are organized in scenarios and they are very effective when one needs to interact with the data. This way, the user can select a region or subset of data to visualize in a focused manner or to change the scenario's point of view. These techniques require a reasonable amount of computational resources or specific hardware so it can be used adequately, in special when regarding huge amounts of data.

Maps are visualization components universally known and can be used as background in graphics that contain geographic information – regions or coordinates. They can, for instance, show a current trend by presenting a visualization example of a geographically-located phenomenon such as malicious accesses to a set of sensors with markers proportional to the quantity of accesses.

3. SECURITY DATA VISUALIZATION

Using logs collected from operating systems, network devices and security applications we can visualize interesting events in a more practical manner than to read a textual file. The application of visualization techniques to security events can provide useful information to help identifying suspicious activities and responding to a incident in a timely fashion. In this Section we will show some approaches to visualize security data related to malware.

Regarding network activities generated by malware, a map-based technique is deployed by The Shadowserver Foundation[11]. IP addresses from monitored botnet controller servers are converted to geographical coordinates and depicted in a world map, where as more servers appear in the same position, greater is the diameter of a point in the map.

In 2007, the CERT.br[12] deployed the SpamPots Project[13] to gather data about compromised machines connected to Internet to send spam. The collected spam data were analyzed to find behaviors in order to identify patterns in spammers activities[14]. To do it, they used data mining techniques to separate a huge amount of spam data into campaigns – message sets which share frequent features and which have few infrequent features to obfuscate the actual message's subject.

Karim et al.[15] describe methods to build malware phylogeny models that can allow one to study malicious code evolution. Besides the authors approach is not directly related to visualization, they use in their paper a technique composed by dendogram and annotation that can be replicated and adapted to other hierarchical-based applications.

Bilar[16] analyzes the struct of malware and goodware callgraphs regarding some properties, such as amount of normal internal function calls, amount of calls to external functions statically (libraries) or dynamically (imports) and amount of thunks (usually function envelopes). The results are presented in a statistical way and with several scatter-plots where each point color is proportional to the executable file size.

Trinius et al.[17] have also used visualization to enhance comprehension of malicious software behavior. They used treemaps and thread graphs to display the actions of the executable and to help the analyst identify and classify malicious behavior.

Fischer et al.[18] show visualization and analysis techniques from NetFlow protocol data. Here, a treemap variant is used to indicate connections among external systems and internal ones in the monitored network. In their work they show the connections done by a 120 zombies botnet with their University systems during an attack occurred in 2008. According to the authors, it was a distributed attack that remained undetected by University's security systems on that time.

4. METHODOLOGY AND RESULTS

Before starting to visualize malware behavior, we must have a collection of malware samples that represents current threats and that actually are being spread through Internet. This collection, as well as a pre-classification step to label the malwae samples were accomplished in a previous work where we set up few distributed malware collectors that are gathering malware samples since 2008[19]. In this Section we will briefly explain how we built a malware behavior monitoring tool with other fellow researchers – BehEMOT[20] – to dinamically analyze samples, how to pre-process behavioral profile data and some techniques we applied to visualize malware behavior, the shortcomings and results obtained.

4.1 Malware behavior monitoring

In order to retrieve malware behavior so we can visualize it, we need to monitor its execution to capture its security relevant actions performed in the operating system and in the network. This task can be accomplished using dynamic analysis, which consists basically of running a malware sample in a controlled environment. Thus, to dynamically analyze malware, we used a slightly modified version of BehEMOT to present not only high-level activities performed by malware, such as file write and delete, process creation and termination, registry reads and writes, mutexes and network operations and so on, but also the system calls called by the malware while doing these actions.

There are several ways to intercept system calls from an executable[21,22] and we chose the technique of System Service Dispatch Table (SSDT) Hooking, which operates at kernel level and allows the modification of the execution flow[23] and the answers returned to the programs[24]. It is implemented through a MS Windows driver that executes at “ring 0” and it has the advantage of not modifying the code of the monitored malicious file, avoiding integrity checks detected by some malware. In Table 1 we present the actions currently monitored by BehEMOT. Besides those, we also capture network traffic related to the malware sample execution, obtaining, at the end of four minutes of running in an emulated environment with Qemu[25], the activities performed by this sample and its child-processes regarding the operating system and the network.

Table 1. OS-related actions that can be captured by the dynamic analysis monitor while running malware samples.

TYPE	OPERATION
FILE	OPEN, READ, WRITE, DELETE
PROCESS	CREATE, TERMINATE, OPEN
REGISTRY	CREATE, READ, WRITE, DELETE
MUTEX	CREATE, RELEASE
NET	CONNECT, SEND, DISCONNECT, RECEIVE

4.2 Visualizing malware behavior

Along with the information shown in Table 1, we have three more pieces of information: the network traffic associated to a malware sample execution, the system calls performed by it – ZwCreateProcess, ZwWriteFile, ZwSetValueKey and so on – and the identification labels provided by submitting the sample to VirusTotal[26].

Using the aforementioned data we can apply different visualization techniques and observe if the results are promising, useful or confusing. Also, we will mention how to convert each kind of data in a simplistic way to use it within the tools. At first, we visualize simple relationships among different malware using JUNG API[27]. We selected approximately 200 samples from our collected malware database and submitted them to VirusTotal. Only positive identification results were stored and associated to the malware sample.

Converting this data to use with JUNG was done by the following way: graph's vertices corresponds to each malware descriptor, while the edges among descriptors have an associated weight which is calculated based on the degree of agreement among antiviruses labeling (for each pair of malware we count how many times antiviruses identify them as being from same category).

As a test to see if they are grouped correctly based on the labels, we colored in red vertices corresponding to malware which is suggested to be part of Tsunami backdoors family[28]. In Figure 1 this graph can be viewed in a layout where vertices are grouped based on their edge values, creating visible clusters – all malware identified as being from Tsunami family were isolated, except for two of them. We are developing a dynamic version of this tool to allow identification of malware which are classified (in a supervised way) in a category but present features from other categories.

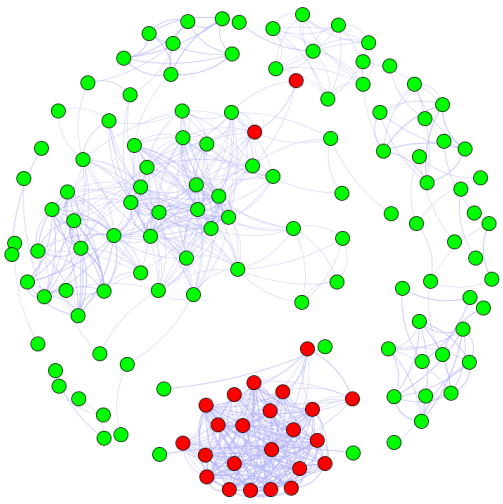

Figure 1. Visualization of malware similarities based on antivirus labeling using JUNG API with grouping.

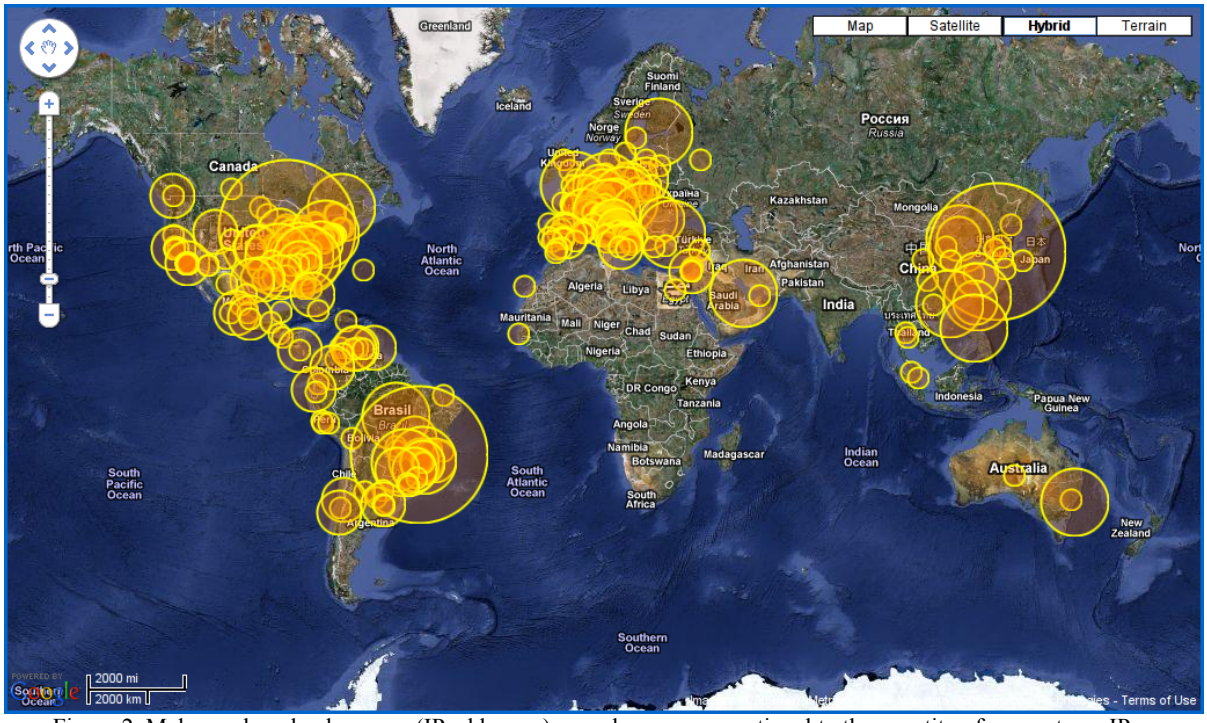

Figure 2. Malware download sources (IP addresses) – markers are proportional to the quantity of access to an IP.

Regarding the operating system behavior, we have time series data representing both system calls that were called by the sample and the actions performed in the system. A work-in-progress we are developing is the categorization of malware according to the system calls performed while running it. Initially, we analyzed this data in an unordered way, i.e., we sorted it for all unique syscalls present in a execution log and then count their frequencies. In Figure 3 we present a small example of malware sample visualization through syscalls. In this example, developed in Java, more darker the line, more this specific syscall appeared in the log, where each line denotes a different syscall.

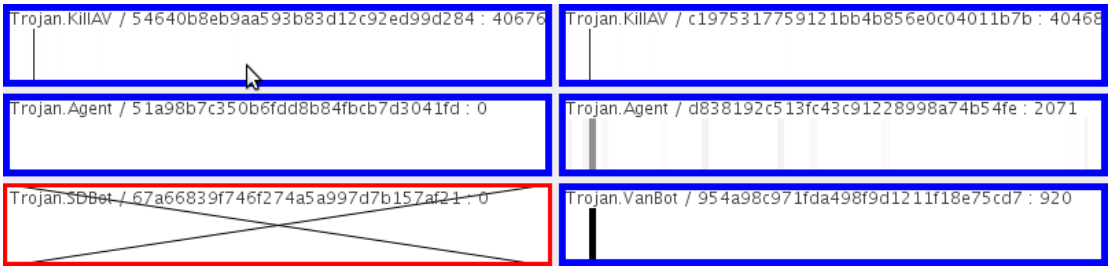

Figure 3. Malware behavior through syscalls incidence; darker lines represent more incidence, lighter ones, less.

Our program reads a textual log file, processes it and prints the graphic. If it reads a file whose malware sample execution didn't log anything, the graphic is blank. If there is no log file or it is corrupted, the graphic is marked with a red 'X'. Here, the objective was to try to see similar behaviors among malware from the same family, but this approach failed. This happened because just the incidence of a determined syscall is not enough to differentiate good programs from bad ones, we need them ordered then. Thus, in this example, it can be confusing to differentiate malware and also to categorize it as a suspicious binary or not, as the large incidence of a small group of syscalls can indicate a normal behavior of binaries in general. Also, we can miss important actions looking for the darker lines while a truly malicious event can be occurring and being missed in lighter lines.

Malware time series events can also be visualized in simple x-y plots, where the x axis can represent the time and the y axis some information about the event. The time information on the x axis can be either the absolute time of occurrence of an event, the relative time of occurrence (counted from a particular initial value) or a simple sequence that implies the order of occurrence of the events. The event information can be plotted using several different methods, often specifically tailored for a particular purpose. The height on the y axis can be used to represent event occurrence amount, severity or intensity if such information is known, or to allow the discrete representation of the types of events. Decorations such as different marks for additional event characteristics can also be used to allow representation of more data dimensions than just two. Additional graphical elements may also be used, but one should always take care not to overload the plot with too much graphical content, which may confuse the user and hinder his/her ability to draw quick conclusions from the plot.

One example of x-y plot used to represent malware time series events is given in Figure 4. It shows a tool developed by the authors that parses a malicious behavior file with malware events ordered by action timestamp and plots it in a simple, interactive interface. This tool plots the whole time series in two panels: on the top panel all points on the series are plotted, with the x axis representing the order in which the events occurred and the y axis representing the action associated with the event (which are not in any particular order). Each event is also plotted as a dot in a particular color, which represents the process who performed a specific action. For example, if the malware associated process created two other processes and interacted with a different already running process, we have four different colors in the graphic. Nor all possible monitored actions have to be performed by a malware, so the y axis varies accordingly to what was present in the behavioral trace captured.

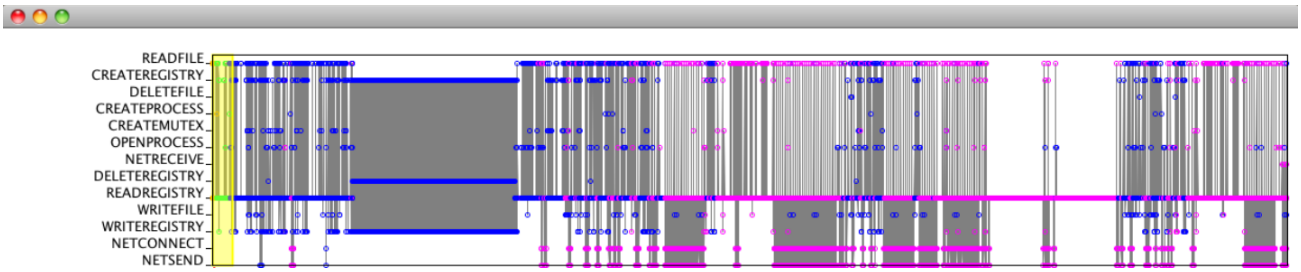

Figure 4. Malware behavior execution time series. This figure depicts the top panel of our tool, were a user is able to walk through it using the yellow magnifier.

The plot created by this tool is also interactive – since there are many events on the top part of the plot it is hard to see exactly which event followed which, so a region of interest (presented under a translucent yellow region on the plot) can be defined by the user, which can change its position by dragging the region with the mouse. The region of interest is shown, enlarged, on the bottom panel of the plot (Figure 5), with the x and y axis and plot colors representing the same information as in the top part of the plot (Figure 4). The bottom part of the plot also shows, in its background, regions in a gray hue corresponding to a basic estimate of variability of operation types around a point on the plot – darker backgrounds around events indicate that there were several different operations around it, while lighter backgrounds indicate that around that particular event there were events similar to it (in regard to operation type).

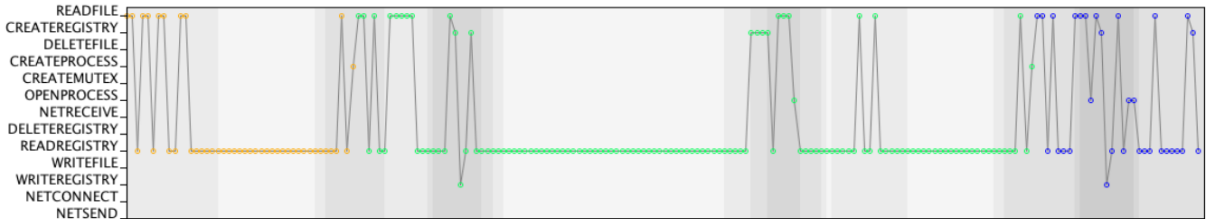

Figure 5. The bottom panel is the magnified view of Figure's 4 yellow region. Here, we can see that the malware sample (yellow dots) starts and then creates a new process (green dots) which created a new one (blue).

5. CONCLUSION

Much attention has been given to malicious software analysis, as it is the major current threat to information systems. Malware behavioral analysis can provide useful information to help in the identification, classification and even in prevention or remediation processes.

This plenty of information generated by analysis systems can be visualized in order to aid an analyst to perform the aforementioned processes or to understand better certain patterns by visual ways. In this paper, we presented some visualization concepts and techniques, cited how they are being used in security-related data, discussed how to use this data to do your own visualization tools and showed approaches we applied to develop our tools and visualize malware behavior data.

There are several other techniques that can be applied to malware related network or operating system data in order to categorize it, for instance, we can use parallel coordinates in network flows or sessions traffic to see differences among suspicious and normal data. It's just a matter of what attributes to chose and if they are good enough to provide useful information.

Also, we showed how to use maps that can be useful to analyze attack trends or visualize sites that are hosting malicious code; and graphs that can be applied to establish relationships among samples, form common shared features groups and families clusters.

On the other side, we presented a tool to visualize system calls accordingly to the incidence they occur, raising a visualization side-effect: are we looking for the right piece of data or the less frequent (and less apparent) data could provide more sensitive information?

Finally, we presented an interactive timeline tool that allows an analyst to observe a malware sample behavior in the order the actions were performed. This way it is easy to one understand what happened in a compromised system step-by-step and, when comparing to another sample, see whether they behave in a similar fashion or not.

Concluding, one must chose the adequate techniques and tools to perform security visualization in a way that it is not a burden greater than to just look at the textual logs. It is also required an adequate preprocessing of the raw data so it can be in fact visualized, providing useful information, not just a heavy processing, graphical confusion stuff.

REFERENCES

[1] Tufte, E. R., [The Visual Display of Quantitative Information], Graphic Press, (2001).
[2] Keim, D., "Visual Data Mining. Tutorial," Proc. 23rd International Conference on Very Large Data Bases, (1997).
[3] Cleveland, W. S., [Visualizing Data], Hobart Press, (1993).
[4] Grégio, A. R. A., "Aplicação de Técnicas de Data Mining para a Análise de Logs de Tráfego TCP/IP," Masters dissertation in Applied Computing at INPE - Brazilian Institute for Space Research (2007).
[5] Inselberg, A., "The plane with parallel coordinates," The Visual Computer 1(2),69-91 (1985).
[6] Inselberg, A., [Parallel Coordinates - Visual Multidimensional Geometry and Its Applications], Springer, (2009).
[7] Kohonen, T., [Self-Organizing Maps], Springer, (1997).
[8] Beddow, J., "Shape Coding of Multidimensional Data on a Mircocomputer Display," Proc. of the First IEEE Conference on Visualization, 238-246 (1990).
[9] Keim, D. A., Kriegel, H-P., "Using Visualization to Support Data Mining of Large Existing Databases," Proc. IEEE Visualization '93 Workshop, (1993).
[10] Shneiderman, B., "Tree Visualization with Tree-maps: A 2-D Space-Filling Approach," ACM Transactions on Graphics 11, 92-99 (1991).
[11] www.shadowserver.org
[12] www.cert.br
[13] www.cert.br/docs/whitepapers/spambots
[14] Calais, P. H., Pires, D. E. V., Guedes, D. O., Meira Jr., W., Hoepers, C., Steding-Jessen, K., "A campaign-based characterization of spamming strategies," Proc. of Fifth Conference on E-mail and Anti-Spam, (2008).

[15] Karim, M. E., Walenstein, A., Lakhotia, A., Parida, L., "Malware phylogeny generation using permutations of code," Journal of Computer Virology, 1(1), 13-23 (2005).
[16] Bilar, D., "On callgraphs and generative mechanisms," Journal of Computer Virology, 3(4), 163-186 (2007).
[17] Trinius, P., Holz, T., Gobel, J., Freiling. F. C., "Visual analysis of malware behavior using treemaps and thread graphs," Proc. of International Workshop on Visualization for Cyber Security - VizSec, 33-38 (2009).
[18] Fischer, F., Mansmann, F., Keim, D. A., Pietzko, S., Waldvogel, M., "Large-Scale Network Monitoring for Visual Analysis of Attacks," Proc. of 5th International Workshop on Visualization for Computer Security - VizSec, 111-118 (2008).
[19] Grégio, A. R. A., Oliveira, I. L., Santos, R. D. C., Cansian, A. M., Geus, P. L., "Malware distributed collection and pre-classification system using honeypot technology," Proc. of SPIE 7344, 73440B-73440B-8 (2009).
[20] Fernandes Filho, D. S., Grégio, A. R. A., Afonso, V. M., Santos, R. D. C., Jino, M., Geus, P. L., "Análise Comportamental de Código Malicioso através da Monitoração de Chamadas de Sistema e Tráfego de Rede," Proc. of Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - SBSEG, (2010).
[21] Garfinkel, T. and Rosenblum, M., "A virtual machine introspection based architecture for intrusion detection," Proc. Network and Distributed Systems Security Symposium, 191-206 (2003).
[22] Father, H., "Hooking Windows API-Technics of Hooking API Functions on Windows," CodeBreakers J. 1(2), (2004).
[23] Kong, J., [Designing BSD Rootkits], No Starch Press, (2007).
[24] Hoglund, G. and Butler, J., [Rootkits: Subverting the Windows Kernel], Addison-Wesley Professional, (2005).
[25] Bellard, F., "QEMU, a fast and portable dynamic translator," Proc. of the USENIX Annual Technical Conference, 41-41 (2005).
[26] www.virustotal.com
[27] jung.sourceforge.net
[28] www.viruslist.com/en/viruses/encyclopedia?virusid=47843