MalViz: An Interactive Visualization Tool for Tracing Malware

Vinh The Nguyen
Computer Science Dept
Texas Tech University
Lubbock, Texas, USA
vinh.nguyen@ttu.edu

Akbar Siami Namin
Computer Science Dept
Texas Tech University
Lubbock, Texas, USA
akbar.namin@ttu.edu

Tommy Dang
Computer Science Dept
Texas Tech University
Lubbock, Texas, USA
tommy.dang@ttu.edu

Keith S. Jones
Psychology Department
Texas Tech University
Lubbock, Texas, USA
keith.s.jones@ttu.edu

ABSTRACT

This demonstration paper introduces *MalViz*, a visual analytic tool for analyzing malware behavioral patterns through process monitoring events. The goals of this tool are: 1) to investigate the relationship and dependencies among processes interacted with a running malware over a certain period of time, 2) to support professional security experts in detecting and recognizing unusual signature-based patterns exhibited by a running malware, and 3) to help users identify infected system and users' libraries that the malware has reached and possibly tampered. A case study is conducted in a virtual machine environment with a sample of four malware programs. The result of the case study shows that the visualization tool offers a great support for experts in software and system analysis and digital forensics to profile and observe malicious behavior and further identify the traces of affected software artifacts.

CCS CONCEPTS

• **Security and privacy → Systems security**; *Intrusion/anomaly detection and malware mitigation*; Software and application security;

KEYWORDS

Malware visualization, dynamic analysis, digital forensics

ACM Reference Format:
Vinh The Nguyen, Akbar Siami Namin, Tommy Dang, and Keith S. Jones. 2018. MalViz: An Interactive Visualization Tool for Tracing Malware. In *ISSTA '18: ISSTA '18: International Symposium on Software Testing and Analysis, July 16–21, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3213846.3229501

1 INTRODUCTION

In the digital age, a vast amount of sensitive information stored in computers and data servers has been increased dramatically. These sugary resources are favourite targets for many attackers. A lot of malicious software, so called malware, has been developed for variety of reasoning such as harming users, computers, networks, or infrastructures in general [13]. When the target computer is infected by a piece of malware, the resources and data stored in the system are tampered without consensus of the user.

ISSTA '18, July 16–21, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5699-2/18/07…$15.00
https://doi.org/10.1145/3213846.3229501

In order to understand the pattern and behavior of a malicious program, two general analysis approaches are often used: 1) static analysis, and 2) dynamic analysis. Static analysis involves analyzing binary signatures of the malware without executing it; whereas, dynamic analysis observes the behavior of the running malicious code in a controlled environment. As the number of samples increases drastically, static or dynamic analysis of every single signature becomes overwhelming and a daunting task. It is salient to have a supporting tool that helps facilitate the process of analysis and assisting in comprehending the suspicious behavior of the malicious software as much as possible.

2 EXISTING APPROACHES

There are several good and useful tools in the literature to accelerate analysis. Wagner et al. [16] surveyed the available tools, and then compared and categorized them based on a number of criteria. These existing tools enable users to delve deeper into processes and system Application Program Interface (API) calls and are capable of detecting potentially malicious software. The survey then proposed a Knowledge-Assisted Visual Malware Analysis System (KAMAS) [17] for analyzing complex data in dynamic malware analysis. Kim et al. [8] proposed a method for classifying and detecting malware based on dynamic behavior through API-call sequences and thus utilizing sequence alignment algorithms. Their experiment was conducted in a realistic settings to avoid mal-functioning malicious code. Several steps are involved to extract API calls which then can be fed into Multiple Sequence Alignment (MSA) algorithms for analysis. The final result can categorize malware into some classes.

Kim et al. [9] used the breath first search algorithm to track the flow of malware behaviors in static analysis. The unusual actions were visualized into graphs consisting of edges and vertices. This approach is very interesting since it is a static analysis-based approach and does not require users to run the actual code.

Each of these tools analyzes a malware program from different perspectives. However delving into every single process for analysis purposes is a very tedious job and time consuming. In addition, it is hard to detect a malware executed in a short time when it terminates itself (i.e., a process is created and then terminated). Traditional Window Task Manager does not retain the terminated processes unless the log file event is investigated. On the other hand, going through hundreds or even thousands of events is a challenging task. It is important to have a better methodology to support this task. An interactive visualization tool can help to a great extent.

3 CONTRIBUTION

Visualization is the process of representing data visually and has been utilized in many research in malware analysis through basic representations, such as scatter plots [11], matrices [6], and

byte plots [5]. However, these studies mostly focus on visualizing known malicious programs and inspecting a certain aspect of their patterns and behaviors. To the best of our knowledge, there is no visualization technique, and in particular with interactive features, for analyzing unknown processes, the system calls they make and their relations.

To fill this gap, this demonstration paper introduces *MalViz* that enables users to view and observe unusual processes and system calls for malware analysis. Malware inspectors are able to quickly detect malicious processes and suspicious patterns. Our approach is different from the existing techniques in terms of time dependencies and the relationship between processes and API-calls. In particular, *MalViz* enables users to interact with processes and threats captured in the given memory dump.

4 THE *MALVIZ* APPROACH

MalViz[1] is developed using JavaScript and in particular the D3.js library [2]. The primary goal of *MalViz* is to create an interactive visual analytic tool that presents system and security experts a high level view of running processes along with their corresponding API-calls and interactions with some other kernel and application-based processes. The tool enables users to investigate the behavior of a malware program over a period of time such as 1) the time-line it starts and remains active, 2) the processes that are triggered and their timing, and 3) the actions that are performed. To meet this goal, this paper proposes several features that are implemented in *MalViz*:

- **Overview Display (F1)**. Display overview of events' distribution.
- **Details-On-Demand (F2)**. Present details on demand, including retrieving *live* malicious patterns based on response result from VirusTotal [14].
- **Chronological Ordering (F3)**. Order processes chronologically and topologically to avoid over-plotting.
- **Library Calls (F4)**. Show the pattern of libraries called by each process.
- **Anomaly Detection (F5)**. Detect anomalies of events from visualization.

4.1 The Format of the Processing Dataset

The dataset template and format processed by *MalViz* can be captured and exported from the Sysinternals Process Monitor [1] toolkit, as a Comma Separated Value file (CSV). The Process Monitor generates thousands of Window's events every second and generates a log file. For the purpose of malware analysis and ease of visualization, the events that are mostly related to malware activities [1] can be further filtered. The filtered events can be classified into four categories:

- **Process** including Process Create and Load Image;
- **File** including CreateFile, WriteFile, and SetRenameInformationFile;
- **Registry** including RegCreateKey, RegDeleteValue, and RegDeleteKey; and
- **Network** including UDP Send, UDP Receive, TCP Receive, TCP Connect, and TCP Send.

[1]For a short demo video of *MalViz* visit https://github.com/Alex-Nguyen/Malviz

These filtered events then need to be exported to a CSV dataset that contains the following information: the time the event is triggered, the process ID, the process name, the operation made by the underlying process, the location of the operation and the additional information of each process. For our case study, these events were captured after the malware was executed for 5 minutes.

5 THE *MALVIZ* ARCHITECTURE

MalViz consists of three main panels as depicted in Figure 1 where 1) Box A contains the *Overview Panel*, 2) Box B presents a list of processes along with API-calls ordered by the time when they have been triggered, and 3) Box C displays the libraries and APIs called by each process.

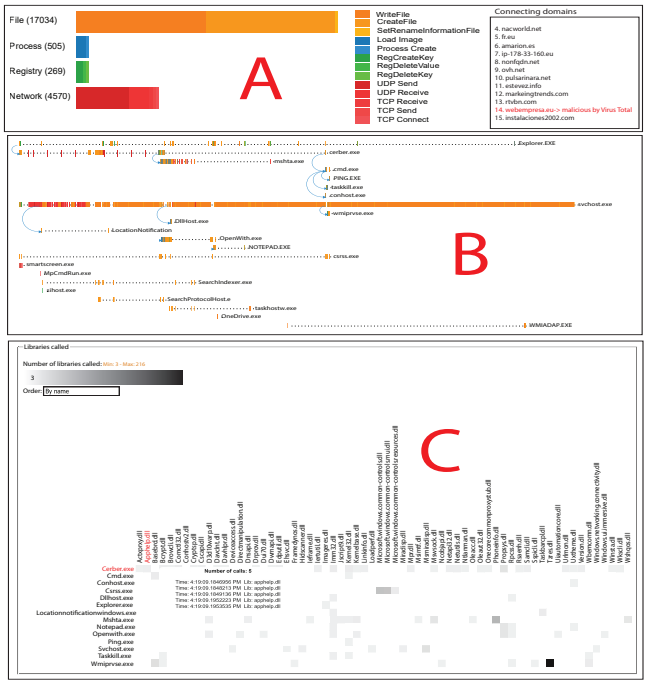

Figure 1: User interface of *MalViz*: A) Overview Panel, B) Detail Panel, and C) Supporting Panel.

5.1 Box A: The Overview Panel

The Overview Panel shows the distribution of 13 operations grouped into four categories (the visualization feature **F1**). Categories are color-encoded. Operations with each category are distinguished by saturation. The wider (and darker) bars indicate that these type of operations are more frequently occurred in the system. The bar charts can also be used to filter operations (described in Section 6). The color legend of process operations is displayed on the middle of the Overview Panel. On the right side of the Panel, a list of domains that the running malware tries to connect is shown. These domains are sent to the VirusTotal [14] service for lively checking of their maliciousness domains. *MalViz* integrates the API offered by VirusTotal to automate the process. The result is color-encoded for easy detection, i.e., malicious domains are encoded as red color.

5.2 Box B: The Detail Panel

The Detail Panel presents the interactions along with the time when each event is triggered. The time axis is aligned horizontally from left to right. This design is widely used when visualizing time series data [18]. Each API-call is represented by a thin vertical bar at its time stamp. Due to the limited screen space, *MalViz* sets the opacity of each API call (vertical bar) to $0.5px$ to help viewers recognize the intensity of its activities. That is, when there are many API-calls at the consecutive time stamps, the vertical stripes become clearly visible. This visualization technique helps users to quickly identify which process makes a lot of API-calls in a short period of time. By default, each API-call will be set with the same height. However, if an API-call tries to connect with a malicious Internet domain, longer bars are plotted. This presentation makes malicious processes stand out (visualization feature **F5**).

Another possible approach to detect anomalies is based on process relations (i.e., one process is initialized by another process). The parent-child relationship is a common behavior usually exhibited by a typical malware program. *MalViz* uses arcs diagrams [4] to encode this relationship. Moreover, *MalViz* organizes the layout so that processes are vertically sorted via a topological ordering algorithm (i.e., "*toposort*") in order to avoid introducing longer arcs (by putting related processes next to each others). When sorting multiple processes of the same parent (or child processes), *MalViz* gives the priority to the earlier-initialized processes. This also helps to minimize edge-crossings (visualization feature **F3**).

A pop-up window is also introduced in this Panel to further display the details of each API-call on demand (visualization feature **F2**). Furthermore, the response results obtained by checking the domain and produced by VirusTotal are appended to this panel.

5.3 Box C: The Supporting Panel

This panel highlights the libraries called by each process (the visualization feature **F4**). The purpose of this panel is to help users quickly detect the unusual patterns of libraries called by each process. A process may call multiple libraries or a library can be loaded by multiple processes. *MalViz* uses an interactive heat-map matrix to represent these multi-relationships as it avoids cluttering compared to node-link diagrams. The value (appears when mouse over) in each cell represents the number of calls by a process. To draw the user's attention, this value is encoded by color opacity which ranges from white to black. The white color means the value has smallest calls whereas the black color carries the largest API-calls.

6 USER INTERACTIONS

MalViz enables users to interact with the visually represented components through mouse over and mouse click.

6.1 Mouse Over

a) Overview Panel. When users mouse over a rectangular box, a tooltip will be popped up to display the type of underlying operation along with the number of API-calls made by the operation.

b) Detail Panel. Similar interactions are also allowed when the user mouses over each API-call (i.e., detailed information is provided). Additional information is also appended to the details, when an API tries to connect to a malicious Internet domain. This information is retrieved from VirusTotal which contains some useful information such as: Harmless, Malicious, Suspicious, Undetected - the number of external tools classified the target domain as harmless (or clean), malicious, suspicious and undetected, respectively. For convenient, malicious data are highlighted as bold fonts with red color. Users can investigate more detailed information of the domain by clicking on the VirusTotal link which will navigate to the original result.

c) Supporting Panel: The tooltip displays all API-calls and the time they have been invoked.

6.2 Mouse Click

a) Overview Panel: When users click on this box, its corresponding API-calls will be highlighted in the Detail Panel. Doing so, other API-calls will be faded out. This interaction allows discovering any API-call patterns of each operation.

b) Detail Panel: When users click on an API, the Detail Panel highlights only its related processes.

c) Supporting Panel: Mouse click is performed within four selections and one slider:

- The sort by `name` selection allows the user to sort the matrix by process and library names in an ascending order,
- The sort by `Diff Libraries called` selection sorts the matrix by the number of different libraries called by each process descending,
- The sort by `frequency` selection descending orders matrix by the number of libraries called; and
- The sort by `similarity` selection sorts the matrix based on similarity values of each cell

Narrowing down the result is also supported by filtering the number of libraries called.

7 CASE STUDY

To evaluate the usefulness and effectiveness of *MalViz*, we conducted two case studies with seven participants, including two faculty members who are experts in Security and Privacy and five graduate students majoring in Computer Science. The purpose of this study was to (1) assess whether the user could detect a malware program within a limited time , (2) find interesting patterns of malicious processes, and (3) receive feedback from user's experience to improve *MalViz*. Before using the visualization tool, all users were introduced the basic functionality of *MalViz* along with some explanations of the common behaviors of malware patterns. The Malware programs for research analysis were retrieved from Malware DB [15]. We conducted the experiment in a virtual environment (i.e., VirtualBox) running Windows 10. Three data-sets were used in this study:

- The first data set contains no malware program, it helps users understand the normal activities of the Operating System as depicted in Figure 2 (A),
- The second data set contains one running malware (ransomeware Cerber [10]) as shown in Figure 2 (B) and
- The last data set contains three malicious programs (Kelihos [12], NJRat [3] and WannaCry [7]) as shown in Figure 2 (C).

Each user interacted with each visualized dataset for five minutes. In the first case study, as depicted in Figure 2(B), we expected to see some interesting patterns on File and Networking activities.

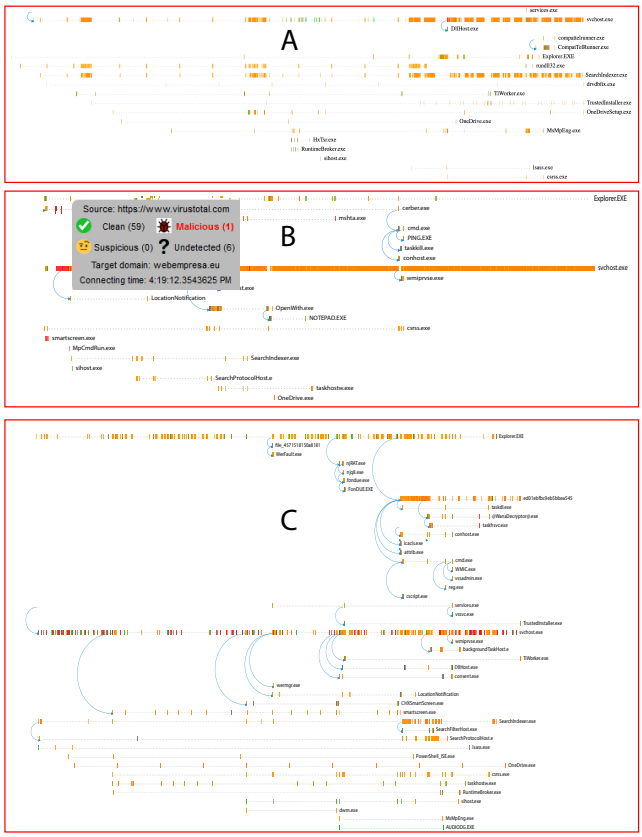

Figure 2: Visualization of behaviors of none (Box A), one (Box B) and three (Box C) malware.

The participants were able to quickly recognize unusual process, which was *cerber.exe*, because of its three indicators:

- The first behavior was the number of processes created in the chain within a short period of time (*cmd.exe*, *conhost.exe*, *taskkill.exe*, *PING.EXE*),
- The second cue was a series of API-calls making together in some time interval (the opacity of the rectangle is clearly visible and the gaps between them are equally distributed), and
- Finally, the rectangular with higher heights indicates activities relating to the malicious domains.

It can be shown from the Figure 2 (B) tooltip box that, out of 66 anti-virus services, one of them reports the target domain as malicious (red color). It can also be seen that system calls made by this malware mostly related to Network activities

In the second experiment, we randomly picked three pieces of malware and ran them simultaneously as shown in Figure 2 (C). With the gained experience from the previous use cases, the participants could quickly detect three chains of processes created which were originated from *Explore.EXE*.

These malicious programs behaved differently and did not call the same APIs. The first malware (from left to right) created only one process while the second and the third malware generated a lot of new processes. We can also notice that the number of libraries called also increases in both volume and variety as we run more malware programs.

User's feedback: After finishing the experiment, we gathered user's feedback to improve *MalViz*, including adjusting the bar of the process to higher the malicious domain was detected, reorganizing the layout (or toposort) and including more information of the tooltip in the Supporting Panel (show event's time and sorting)

8 CONCLUSION AND FUTURE WORK

In this demonstration paper, we introduced *MalViz*, a graphical tool that allows researchers, especially in security and malware analysis field, to gain a better understanding about malware behaviors based on the relationship between processes in a timely manner, and malicious domains notification.

The usefulness of the application is evaluated through two case studies where non-experts in security were able to find some typical interesting processes. Our application can also be able to visualize the interesting pattern of libraries called by each process and thus gives hints to experts for further analysis.

ACKNOWLEDGEMENT

This project is funded in part by a grant (Award#: 1516636) from National Science Foundation.

REFERENCES

[1] Moti Bani. 2016. Process Monitor for Dynamic Malware Analysis. (2016). https://blogs.technet.microsoft.com/motiba/2016/05/04/process-monitor-for-dynamic-malware-analysis/.
[2] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3 Data-Driven Documents. *IEEE Trans. Vis. Comput. Graph.* 17, 12 (2011), 2301–2309.
[3] New Jersey Cybersecurity Communications Integration Cell. 2017. NJRat. (2017). https://www.cyber.nj.gov/threat-profiles/trojan-variants/njrat.
[4] Tuan Nhon Dang, Nick Pendar, and Angus G. Forbes. 2016. TimeArcs: Visualizing Fluctuations in Dynamic Networks. *Computer Graphics Forum* (2016). https://doi.org/10.1111/cgf.12882
[5] John Donahue, Anand Paturi, and Srinivas Mukkamala. 2013. Visualization techniques for efficient malware detection. In *Intelligence and Security Informatics (ISI), 2013 IEEE International Conference on*. IEEE, 289–291.
[6] KyoungSoo Han, Jae Hyun Lim, and Eul Gyu Im. 2013. Malware analysis method using visualization of binary files. In *Proceedings of the 2013 Research in Adaptive and Convergent Systems*. ACM, 317–321.
[7] Emi Kalita. 2017. *WannaCry Ransomware Attack: Protect Yourself from WannaCry Ransomware Cyber Risk and Cyber War*. Independently published.
[8] Hyunjoo Kim, Jonghyun Kim, Youngsoo Kim, Ikkyun Kim, Kuinam J Kim, and Hyuncheol Kim. 2017. Improvement of malware detection and classification using API call sequence alignment and visualization. *Cluster Computing* (2017), 1–9.
[9] Jihun Kim and Jonghee M Youn. 2017. Dynamic Analysis Bypassing Malware Detection Method Utilizing Malicious Behavior Visualization and Similarity. In *Advanced Multimedia and Ubiquitous Engineering*. Springer, 560–565.
[10] Malwarebytes Labs. 2018. Ransom.Cerber - Malwarebytes Labs. (2018). https://blog.malwarebytes.com/detections/ransom-cerber/.
[11] DongHwi Lee, In Soo Song, Kuinam J Kim, and Jun-hyeon Jeong. 2011. A study on malicious codes pattern analysis using visualization. In *Information Science and Applications (ICISA), 2011 International Conference on*. IEEE, 1–5.
[12] MalwareTech. 2017. The Kelihos Botnet. (2017). https://www.malwaretech.com/2017/04/the-kelihos-botnet.html.
[13] Michael Sikorski and Andrew Honig. 2012. *Practical malware analysis: the hands-on guide to dissecting malicious software*. no starch press.
[14] Hispasec Sistemas. 2018. VirusTotal Public API v2.0. (2018). https://www.virustotal.com/en/documentation/public-api/[Accessed date: Feb 14, 2018].
[15] Yuval tisf Nativ. 2016. theZoo aka Malware DB. (2016). http://thezoo.morirt.com/.
[16] Markus Wagner, Fabian Fischer, Robert Luh, Andrea Haberson, Alexander Rind, Daniel A Keim, Wolfgang Aigner, R Borgo, F Ganovelli, and I Viola. 2015. A survey of visualization systems for malware analysis. In *EG Conference on Visualization (EuroVis)-STARs*. 105–125.
[17] Markus Wagner, Alexander Rind, Niklas Thür, and Wolfgang Aigner. 2017. A knowledge-assisted visual malware analysis system: Design, validation, and reflection of KAMAS. *computers & security* 67 (2017), 1–15.
[18] Martin Wattenberg. 2005. Baby Names, Visualization, and Social Data Analysis. In *Proc. IEEE Symp. on Information Visualization*. 1–7.