Visualization of Processes and DLL Usage

Connor Leavesley
Rochester Institute of Technology
Rochester, NY, USA

ABSTRACT

This paper introduces a visual analytic technique for analysis of process actions through process monitoring using collected host logs. The goals for this paper are to investigate the relationships between processes and process actions over time on a system, and to support security operations professionals in recognizing suspicious and unusual activity based on process actions. A case study is conducted using print-outs of the visualized processes with a sample of four malicious processes. The results of the case study show that the visualization technique offers benefits to security experts in process analysis to find malicious processes and see how they interact with the system.

Author Keywords

visualization; security; process; processes; dynamic link library; DLL

CCS Concepts

•**Security and privacy** → **Usability in security and privacy;**

INTRODUCTION

The number of processes that exist on computers continues to increase as operating systems and applications get more complex and numerous. These processes can do any number of things, from helping run aspects of the operating system, browsing the internet, or editing documents. Many of these processes have to interface with the operating system kernel using dynamic link libraries, or DLLs for short. DLLs host shared code, and cut down on the size and bulk of programs. They are easy to monitor for malicious activity and announce the presence of malicious programs. In response, malware tries to use as little DLLs as possible to stay undetected.

In the past, analysts have examined the DLLs that a process uses by looking at lists from logs. However, this requires analysts to extract patterns from text, a task that can be difficult to do when dealing with unknown processes. By visualizing the correlation between processes and the DLLs they use in a heat map, security analysts can easily see what DLLs are commonly used in conjunction with each other, and the difference between benign and malicious DLL usage. With this, it would become far easier to identify clusters of used DLLs to establish malware type.

CSEC659, December 4, 2020, Rochester, NY, USA

© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-6708-0/20/04. . . $15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

RELATED WORK

Samy et al. [5] did an analysis of feature categories for malware visualization. Four feature categories were derived: dynamic features, static features, hybrid features, and application metadata. Five visualization classifications were also introduced: host/server, internal/external, port activities, attack pattern, and routing behavior. Each had a list of visualization techniques that would work well with that data set. According to the chart, color maps, radial panels, scatter plots, or parallel coordinates would be the best options for integral processes. The choice of features to visualize are important for an effective system. Any visualization has to effectively choose the most important features to show to the security practitioner. In addition, the choice of manner to display these features is equally important. If important features are shown in an efficient and effective manner, it will be of great use to analysts.

There have been several tools to visualize aspects of computer system processes. Microsoft, for instance, has the Sysinternals Process Monitor for monitoring processes on a host system. It provided a user with a much more detailed breakdown of parent and child processes and related information about the process binary. It also added color coding and indentation to give information about a process with a quick glance. This method, versus the traditional method, "task manager", quickly established Process Monitor as a far better option for finding strange processes on a system [1]. Fink et al. [2] proposed a method for correlating network traffic between processes. On one side the authors had a list of client processes, and on the other there was a list of server processes. Processes that were connected to one another had a line between them that was color coded by network state. This method provided a way to view process interaction with the network interaction and the interaction with other processes. Though the system was easy to read, it lacked the scalability required for more than 39 processes on the screen at any given time. Both these visualization techniques have effectively correlated processes to a property of those processes. In the case of Process Explorer, it visualizes child processes with great effectiveness. In the case of Fink's system, processes are correlated to network action. These methods are useful and offer insight in how to display process properties for other applications.

Further research has been done in visualizing malware process features. Kim et al. [4] visualized API call sequences to help analysts detect malware. API call sequences are visualized as strands of DNA, with each strand assigned to its respective host process. This system gives the analyst a view of the call chain of specific processes, giving further insight into what the processes are doing. Trinius et al. [7] took a similar approach by using tree maps and thread graphs to visualize malware behavior and API calls. The tree map served as a graphical representation of the percentage each API call out of the total API calls for each sample. The thread graphs served to give analysts a idea of which APIs were called over time. However, these graphs can only follow one execution path of a sample, which means it lacks scalability. Andre and Santos [3] graphed system call incidences to represent malware behavior in addition to using tree maps. Saxe et al. [6] developed visualization for malware using shared system call sequence relationships. The individual system calls were placed on a map of system calls. Different system calls would be highlighted at once to represent a sequence, and the samples that used that sequence would be presented to an analyst to help discover patterns between samples. These systems display important features of malware processes that are useful to an analyst. However, they lack scalability and fail to strip out unimportant features that end up cluttering the system, or visualize too much and end up overwhelming the system.

RESEARCH QUESTIONS

Processes need DLLs to interact with the operating system. This proves an excellent way to monitor processes for malicious activity. Those DLLs can easily point to if a process is malicious or not, provided the analyst has some insight into what those DLLs do and what can happen when they are used in a certain way or order. A visualization system that visualizes the correlation between processes and DLL usage should clearly show which major DLLs are used and which are not, and be able to be compared to other programs to extract patterns. Malware will have different DLL usage compared to regular programs, and therefore will likely stand out from benign programs. With the ability to glance at a visualization, analysts should be able to gather what is out of the ordinary faster than looking at regular text-based systems. As such, this paper seeks to answer the following research questions:

1. **How can process and DLL usage correlation be visualized in a way that is intuitive to an analyst?**
2. **Can analysts extract patterns to determine which processes are malicious?**
3. **How much time can visualization save in system analysis?**

METHODOLOGY

This research seeks to measure the effectiveness of visualizing process and process DLL use. As such, the research design follows an experimental structure. First, we will describe the design of the visualization. Second, we will describe the method of subject discovery and the subjects that took part in the study. Third, we will describe the materials used for the study. Fourth, we will describe the methods used to run the study. Finally, we will describe the problems and limitations that arose during the study.

Design

The visualization system follows the concept of a heat map. If a DLL is used by a process, it is marked as yellow on the map, else it is marked as purple. This allows for a high degree of distinction between present and absent DLLs. On the Y-axis is the list of processes gathered from the system; on the X-axis is the combined set of DLLs those processes call. The DLL names are printed at a 45-degree slant to increase compactness and readability. The visuals were made using Python's plt library, and the DLLs were extracted using the strings command line utility from processes running on a Windows 10 virtual machine. Ten benign processes were chosen for visualization: csrss, services, mpcmdrun, msdtc, dllhost, dwm, conhost, spoolsv, wininit, and iexplore. The four malicious processes used were Worm_Vobfus.smm2, Win64.Trojan.GreenBug, and two versions of StringPity. To prevent participants from using the process name to determine maliciousness, the process name was obfuscated in the visualization with a number. This ensured that participants only determined maliciousness based on the used DLLs.

Subjects

Subjects were gathered through the information security community at the Rochester Institute of Technology (RIT). Participants were asked if they were willing to take part in a fifteen-minute study on process to process DLL correlation. The participating subjects were three computing security students from RIT. Participant One was in their final year of graduate study, with expertise in secure programming and Linux subsystems. Participant Two was also in their final year of graduate study, with expertise in penetration testing and Linux subsystems. In addition, both of these participants had one to two years of real-world experience in information security. Participant Three was in their second year of undergraduate study with no particular area of in-depth knowledge.

Materials

To conduct the study, two images were shown, each corresponding to a different stage of the study. A computer is needed to display the images. The first image had the DLL usage of eleven process. Ten of the processes were benign, and the eleventh process was StrongPity. The second image had the DLL usage of thirteen processes, with ten being benign and three being StrongPity, Worm_Vobfus.smm2, and Win64.Trojan.GreenBug.

Methods

Before beginning the study, a brief description and overview of the system was orally given to the participant. This overview described the layout of the visualization and its purpose. The actual study is split up into two parts: the introduction phase and the malware detection phase. In the first phase, a visualization of ten benign processes and one malicious process are presented to the participant and they were asked to find the malicious process. The participant was free to ask questions about the system and how to use it to clear up any confusion.

Figure 1. A subset of the first data set visualization.

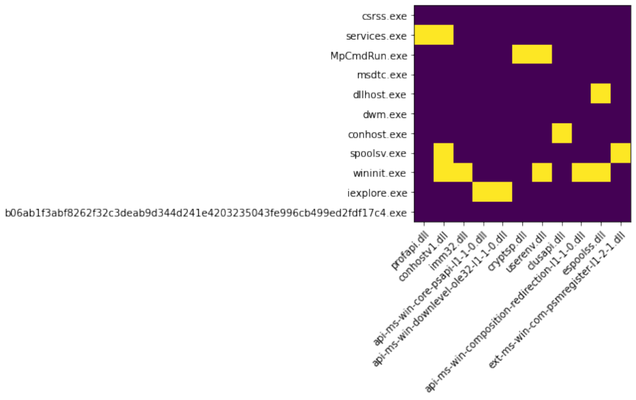


Figure 2. First data set visualization.

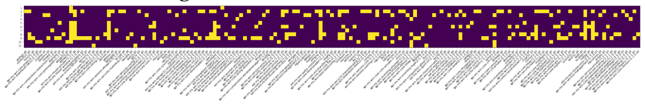


Figure 3. A subset of the second data set visualization.

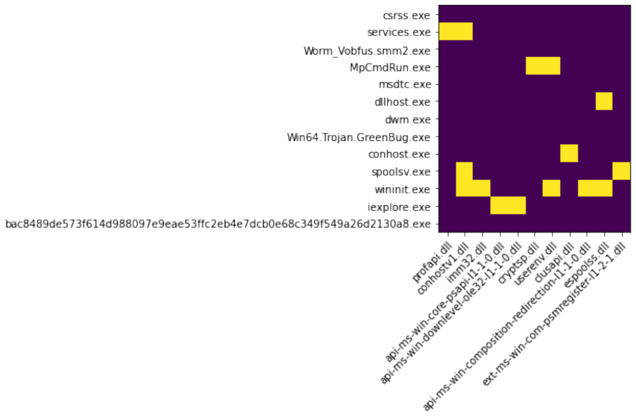
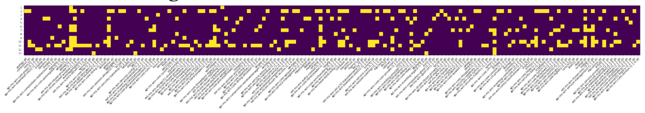


Figure 4. Second data set visualization.



Once the participant made their choice, they were asked for their reasoning and their thoughts on the design of the system. In the second phase, the participant was shown a visualization of ten benign processes and three malicious processes and were asked to identify the malicious processes. This time, the participant was timed while they conduct their analysis and make their decision. They were then asked to explain their reasoning.

Problems

Participants were quite confused by the description of the system before it was presented. When the first phase was presented to them, there appeared to be a disconnect between what they envisioned and what was presented to them. In addition, as the study was split into two parts, each requiring the user to find malware, the users had no chance to get comfortable with the system in a non-stressing situation. For future work on this visualization technique, a mock-up with no malware samples should be created so that participants can ask questions about the system and its design without feeling the need to find a malware sample as well.

RESULTS

The conducted study had three participants, all with differing levels of expertise in the computing security field. In the first part of the study, two of the three participants were able to find malware in the visualization shown to them. When asked about their reasoning, two participants drew correlation between a process's lack of DLL usage and its probability in being malware. In addition, all the participants relied on DLLs they were familiar with and DLLs that had the terms "security" or "kernel" in the name. All participants narrowed down the selection to two to three processes out of the initial eleven before making their decision. All three participants included the actual malware sample in their narrowed range.

# of Malware Samples in Narrowed Range in Part 1

Participant	Narrowed Range	# of Malware Samples
1	3 Samples	1 Samples
2	3 Samples	1 Samples
3	2 Samples	1 Samples

There were two main issues with the design pointed out by participants when asked about the look and feel of the visualization system: difficultly in tracing back a square in the map back to a process, and that there were too many DLLs used in the visualization. Due to the number of DLLs used in the visualization, participants had to trace back from one side of the visualization to the other due to a lack of delineation between lines. Other issues brought up by participants were that the axes were not labeled, that it was difficult to read text at a slant, there were DLLs that were not important to malware investigation, and that the DLLs were not in alphabetical order.

In the second phase, two of the three participants found two of the three malware samples, while the remaining participant only found one. No one found all three samples. Participants were able to narrow down the choices between four or five samples out of the original thirteen before they made their choice. At least two of the actual malware samples were included in that narrowed range, with two participants having two and one having all three. In addition, all three participants found the same sample first, sample #13.

# of Malware Samples in Narrowed Range in Part 2

Participant	Narrowed Range	# of Malware Samples
1	5 Samples	3 Samples
2	4 Samples	2 Samples
3	5 Samples	2 Samples

Participants spent nearly the same amount of time on part two, with all participants taking between four to five minutes. On average, participants spent 4 minutes and 28 seconds before they were confident they had found all three samples. On average, participants found 0.67 malware samples in the first phase, and 1.67 samples were found in the second phase.

Participant # of Correct Choices and Time

Participant	# in Part 1	# in Part 2	Part 2 Time
1	1/1	2/3	4:23
2	0/1	1/3	4:58
3	1/1	2/3	4:03
Avg.	0.67	1.67	4:28

DISCUSSION

In this section we will discuss the results of the study and conduct a critical reflection of the data gathered. First, we will interpret the results and what they mean. Second, we will review the impact for security practitioners. Third, we will conduct a critical reflection of the methodology. Finally, we will suggest refinement of the method and areas of future research.

Interpretation

It is evident that a heat map is slightly effective at assisting analysts with general knowledge of the security field at finding malicious processes. Participants were able to narrow down potential malware samples using the visualization, resulting in 1.67 pieces of malware being found on average. In seemingly random data, 1.67 samples found is far higher than expected. Participants had two to three probable samples in the first part and four to five in the second. In part one, the malware sample was in all three participants narrowed down sample range. In part two, all three participants had at least two of the malware samples in their narrowed down range. While the final decisions still came down to guesses by the participants, they showed they were able to greatly narrow down the possibilities using the system.

When asked about the look and feel of the visualization, participants complained that they felt that far too much data was being shown at once. While this may be the case, the quantitative data gathered shows that users were able to gather some patterns out of the visualization, resulting in not needing to trace back specific DLL uses too often. While the way to delineate between the lines would be useful in specific instances and would likely improve the results, due to how users have shown that they will look for patterns in the data, the addition of a delineation to take the focus away from the pattern searching.

Impact

Analysts have to understand a lot of data about processes on a system to determine what is malicious and what is not, including what processes are expected on a system and what DLLs, when used in conjunction with each other, indicate malicious actions. This data is mixed together in host logs from dozens to hundreds of systems, making the search for malware a needle in a haystack. The proposed system was shown to assist in finding malware samples from a mix of processes. With the ability to see all the data in the sample place and pick out known patterns of DLL usage of malware, analysts can quickly pick out suspected malicious processes and investigate them closely elsewhere.

Critical Reflection

The high time to decide and low accuracy are likely due to a number of environmental and system design factors. Of the participants, two were highly experienced in Linux subsystems and malware design, and one had general knowledge of both Linux and Windows. The data presented to the participants utilized Windows only DLLs, and did not represent the knowledge base of the participants. If analysts who were experienced in Windows subsystems and malware detection had participated, they likely would have done far better. In addition, the participants complained that the data was too spread out, that they had to scroll back and forth to line up data, and it was difficult to follow data back to a process or DLL. This can be remedied by making the individual heat boxes smaller to make the map smaller, and adding delineation lines to help follow lines back to the label. In terms of the data in the visualized data, it was not randomized between phases in the study. This resulted in all but three of the lines in the second image being the same as the first. This is also likely the reason that all three participants said sample #13 was malicious as that was the location of the malicious sample in part one.

Future Research

The current visualization system has been shown to have shortcomings in scalability, but today's computers can have hundreds of processes and DLLs on the system. Without the ability to visualize processes and process data from host machines, visualization techniques will suffer. In addition, a lot of DLLs are not important to malware investigations as many do not relate to system permissions and actions. Future research should focus on making visualizations scalable, particularly in techniques that are applicable in multiple different forms of visualization and further work into what DLLs are important to malware investigations that can be used in visualization.

CONCLUSIONS

In this paper we have discussed how to visualize processes to process DLL usage in a way that is intuitive to analysts searching for malware. First, we conducted a literature review of related work in the area of process visualization. Second, we proposed the research questions this paper sought to answer. Third, we covered the design of the proposed visualization solution and the structure of a study to test its effectiveness. Finally, we discussed the results of the study and their implications.

The number of processes that exist on computers continues to increase time goes on. All these processes need to use DLLs, and those DLLs are key indicators to a process's maliciousness. Analysts need tools that allow them to quickly discern information about a process and it's DLLs without having to digest large amounts of texts and data to understand what is going on. The key to this is visualization. Visualization is incredibly important for today's security analysts, and it will only become more so in the future.

REFERENCES

[1] 2020. Process Monitor v3.60.

[2] G. A. Fink, P. Muessig, and C. North. 2005. Visual correlation of host processes and network traffic. In

IEEE Workshop on Visualization for Computer Security, 2005. (VizSEC 05). 11–19.

[3] André R. A. Grégio and Rafael D. C. Santos. 2011. Visualization techniques for malware behavior analysis. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense X*, Edward M. Carapezza (Ed.), Vol. 8019. International Society for Optics and Photonics, SPIE, 9 – 17. DOI: http://dx.doi.org/10.1117/12.883441

[4] H. Kim, J. Kim, Y. Kim, I. Kim, K. J. Kim, and Hyuncheol Kim. 2017. Improvement of malware detection and classification using API call sequence alignment and visualization. *Cluster Computing* 22 (2017), 921–929.

[5] Ganthan Narayana Samy, Pritheega Magalingam, A. Ariffin, Wafa Mohd Khairudin, Mohamad Firham Efendy Md Senan, and Zahri Yunos. 2018. Analysis of Feature Categories for Malware Visualization. *Journal of Telecommunication, Electronic and Computer Engineering* 10 (2018), 1–5.

[6] Josh Saxe, David Mentis, and Chris Greamo. 2012. Visualization of Shared System Call Sequence Relationships in Large Malware Corpora. In *Proceedings of the Ninth International Symposium on Visualization for Cyber Security (VizSec '12)*. Association for Computing Machinery, New York, NY, USA, 33–40. DOI: http://dx.doi.org/10.1145/2379690.2379695

[7] P. Trinius, T. Holz, J. Göbel, and F. C. Freiling. 2009. Visual analysis of malware behavior using treemaps and thread graphs. In *2009 6th International Workshop on Visualization for Cyber Security*. 33–38.