DaViz: Visualization for Android Malware Datasets

Tomás Concepción Miranda, Jean-François Lalande, Valérie Viet Triem Tong, Pierre Wilke

DaViz: Visualization for Android Malware Datasets

Tomás Concepción Miranda, Jean-François Lalande, Valérie Viet Triem Tong, Pierre Wilke
CentraleSupélec, Inria, Univ Rennes, CNRS IRISA, Rennes, France

***Abstract*—With millions of Android malware samples available, researchers have a large amount of data to perform malware detection and classification, specially with the help of machine learning. Thus far, visualization tools focus on single samples or one-to-many comparison, but not a many-to-many approach. In order to exploit the quantity of data from various datasets to obtain meaningful information, we propose DaViz, a visualization tool for Android malware datasets. With the aid of multiple chart types and interactive sample filtering, users can explore different application datasets and compare them. This new tool allows to get a better understanding of the datasets at hand, and help to continue research by narrowing the samples to those of interest based on selected characteristics.**

I. Introduction

Nowadays malware researchers have millions of applications available to study. In the Android malware research community, AndroZoo provides over ten million APK files, Android's application package, available to download, including benign applications and malware obtained mainly from Google Play. Other repositories, such as VirusShare [1] and Malpedia [2], provide only malware as this is their main focus. Researchers then choose applications from these repositories and build their experiment. They mainly perform the following types of experiment: reverse engineering to understand a sample's behavior, malware detection, malware family classification. To help with these tasks, visualization tools provide ways to show relevant information about one application or a dataset of applications. In this paper we present DaViz, a visualization tool for representing Android malware datasets. This document is structured in the following way: after the state of the art in Section II, we describe our tool in Section III and a use case in Section IV. Lastly, we make our conclusions and hints for future works in Section V.

II. State of the Art

In this section, we review the literature on works presenting visualization methods for malware analysis. We start with those aimed towards the Android environment, then with more recent tools for x86, and lastly methods for visualizing malware datasets.

Jain et al. [3] propose a method to visualize the program's `.dex` file into a color 2D image. The method aims to help recognize patterns in the bytecode file, by allowing the user to match colors and structures while comparing with other applications. Jenkins et al. [4] propose a tool to better understand inter-component communication and intents in an application. Using dynamic analysis, the tool outputs a graph of calls between components. Santhanam et. al [5] present a tool to visualize artifacts of an Android program, in order to help an analyst studying an unknown application if it violates confidentiality, integrity or availability. These artifacts can be control flow graphs, system flow graphs, information flow graphs, or a combination of these results. By looking at the interactions operated by the program, the analyst can discern more easily if the application contains a possible malicious behavior. Lalande et. al [6] propose a tool for representing the dynamic execution of an application with a replay option. It allows to visualize the system flow graph, which represent everything the application touches (a taint analysis), allowing to see the interactions the program performs during execution. Although a precise understanding of a single application is their main goal, these tools are not suitable for comparing many applications at the same time when more malware appear day by day.

The next papers are other recent, non-Android specific malware analysis visualization tools. MalViz [7] is a malware tracing visualization tool. It allows the user to see the interaction between processes and API calls. The interface shows the types of operations performed by the program, a time-lapse of the interactions with their triggers, and the libraries used by each process. Case studies were conducted with Windows system programs and malware to show behavior differences between these types of programs. The SymNav [8] tool was designed to show the execution tree until the ”malicious behavior”, abstract these execution steps, and then compare them to other samples. By comparing the execution steps, analysts get a glance of the type (and family) of malware they have at hands.

In addition to visualize a single sample, other visualization tools focus on a set of applications. The following papers try to represent a collection of applications, by either presenting information of every element it has, by showing similarities between subsets of elements, or both. Saxe et al. [9] present a tool that shows similarities between applications in a malware dataset, according to their system call sequences. Semantic sequences of system calls in logs are calculated, and then compared between the different applications in the dataset. Gove et al. [10] propose a tool to compare sets of attributes of malware sample to a dataset. Using histograms, Venn diagrams and matrices, the tool show how similar the sample is to a dataset. It helps reverse engineers speed up their work with new applications, and analysts to discern whether new samples are based on other older samples, or if they are different applications. Event though these solutions allow to represent sets of programs, other kinds interactions and characteristics are available to exploit them for different tasks in malware research.

III. DaViz

In this section we will present our malware datasets visualization tool called DaViz (which stands for Dataset Visualization), the types of chart it can produce, and the tasks users can accomplish with it.

A. *Functionality*

DaViz allows users to browse datasets using different types of charts. These charts are created by specifying the type of chart and the characteristics users want to show. These characteristics where calculated using Droidlysis [1], a "property extractor for Android apps": for each application it outputs different boolean characteristics from signatures in the code, alongside other ones from the application's Manifest file (like permissions, intents, activities, etc.).

Different charts can be created using the multiple characteristics available:

- **Venn (Euler) charts**: These charts allow to see the set relationships between the different characteristics, where the size of each set corresponds to the number of applications with that characteristics. Intersections between sets correspond to the number of applications with the characteristic in common. It also allows to see the integration of the chosen characteristics: if two of them are mutually exclusive, they will appear separate from each other.
- **Bar charts**: These charts show the size relationship between bars. Each bar represent a single characteristics and its size represents a value such as: the number of applications with the corresponding characteristic, the average size, and others. Users specify both abscissa and ordinate before creating the chart. In the case of characteristics such as size and the application's date year, users specify the range for each bar.
- **Heat charts**: These charts show the magnitude between two dimensions with a color hue. In our case, users specify the different characteristics for the abscissa and the ordinate, and lastly the value for each intersection (number of applications having both characteristic, the average size, etc.).

B. *Interactivity*

Interactivity is one of the key features of DaViz: once a chart is created, users can create more charts that will appear from left to right of the first chart and an interaction performing a selection on one of the charts will affect the charts on the right. The intent behind this behavior is to let users selects sub-parts of the datasets based on filters applied to the characteristics and see the results on the right diagrams. For example, if the first chart is a Venn diagram and the user selects one set (that represent one characteristic), all the elements of next charts will only contain applications that have this characteristic. Users then are able to manipulate the charts by selecting the characteristics they are interested to see.

C. *Associated tasks*

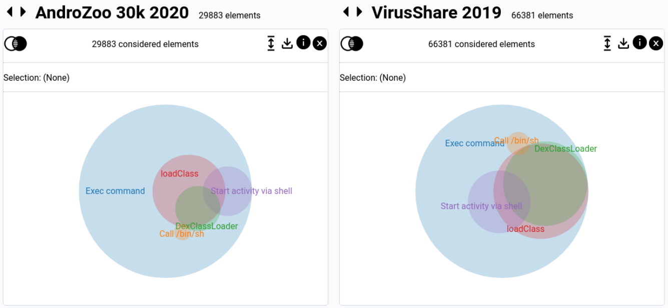
DaViz allows to accomplish two lain tasks: dataset exploration and dataset comparison.

The first task, dataset exploration, has a "discovery" and "exploration" goal according to the different goals defined by Munzner [11]. By exploring the datasets, users can gain new knowledge, verify hypothesis, or generate new sub datasets from the data at hand.

The second task, dataset comparison, has a "comparison" goal [11]: once users choose the datasets they are interested in, charts can be created to show differences between datasets. Multiple datasets can be compared at once, but a one by one comparison allows a more focus approach by concentrating in only two datasets at a time.

IV. Use case

In this example we show a comparison between two datasets: VirusShare 2019 a malware dataset of 66,381 applications and AndroZoo 30k 2020, a 29,883 sample from AndroZoo extracted in 2020. Two charts are created: a Venn diagram and a bar chart.


AndroZoo 30k 2020 29883 elements

29883 considered elements

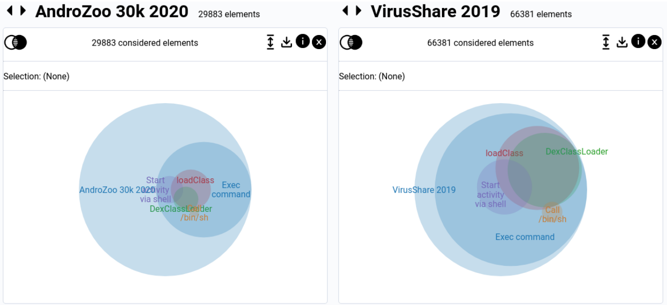
Selection: (None)

VirusShare 2019 66381 elements

66381 considered elements

Selection: (None)

Fig. 1a. A Venn diagram in DaViz


AndroZoo 30k 2020 29883 elements

29883 considered elements

Selection: (None)

VirusShare 2019 66381 elements

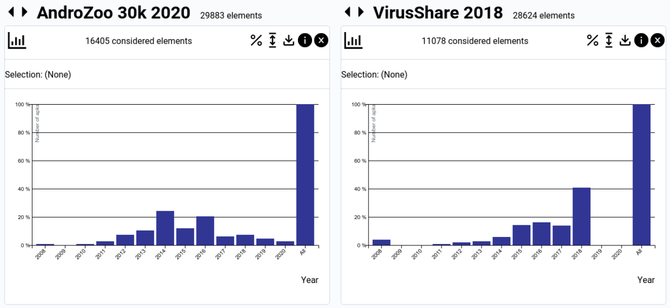
66381 considered elements

Selection: (None)

Fig. 1b. The same Venn diagram with the whole dataset included

In the Venn diagram, shown in Fig 1a, we can see the that "Exec command" is the biggest set (the characteristic present the most in comparison to the others), with almost all other sets inside it, meaning that "Exec command" is always present in an application if these other are. We can see a difference in the intersection of "DexClassLoader" and "loadClass" between the two datasets: in AndroZoo 30k 2020 there is less use of both characteristics simulteously than in VirusShare 2019. Notice

[1] Droidlysis' github page



AndroZoo 30k 2020 29883 elements

16405 considered elements

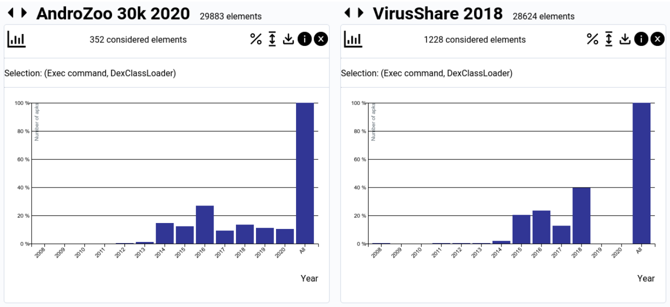
Selection: (None)

VirusShare 2018 28624 elements

11078 considered elements

Selection: (None)

Fig. 2a. A bar chart in DaViz



AndroZoo 30k 2020 29883 elements

352 considered elements

Selection: (Exec command, DexClassLoader)

VirusShare 2018 28624 elements

1228 considered elements

Selection: (Exec command, DexClassLoader)

Fig. 2b. The same bar chart with filtered elements

that the sets' sizes represent the proportion between them in one dataset, which can be misleading for comparison. In order to fix this, users can compare the sets with the whole dataset respectively. By selecting the scale option, the previous Venn diagram changes to Fig 1b. Now we can see that the ''Exec command'', although is the most present characteristic among the others selected, is not as present as in the malware dataset, where almost every application use it. The user can conclude that the VirusShare dataset contains mostly malware that try to execute code more frequently compared to the general case of applications in AndroZoo, and further inspection may be performed.

The bar chart, shown in Fig 2a shows the distribution of applications in the dataset by year. By default, it will show all possible applications with a date, no filter is applied (Fig 2a). The user can, for example, select in the Venn diagram to only show in the bar chart those applications with ''Exec command'' and ''DexClassLoader'' by clicking the in the Venn diagram the intersection corresponding to these two characteristics. The bar charts for both datasets will update, shown in Fig 2b, to show the same distribution but for application containing both characteristics chosen in the Venn diagram. We can see that, for AndroZoo 30k 2020, 2016 is the year with most applications with this selection (Exec command'' and ''DexClassLoader'') while for VirusShare 2019 is 2018. The trend is that there are more applications in this last dataset containing the select characteristics than in the other dataset. To perform a better comparison, users can change the scale of the ordinate to show percentage, and then show the dataset total.

V. Conclusion and Future Works

In this paper we presented DaViz, a dataset visualization tool for exploration and comparison. Using different techniques, like filtering and reordering, users can gain an insight of the malware datasets at hand, and compare datasets to understand their differences and similarities. The tool, being a work-in-progress, can still be modified and improved. Future work will concentrate in adding new functionalities, like automatic selection of features that maximizes the difference between two datasets, and adding other types of charts. Lastly, a future goal would be to perform a user experience test. It will allow to compare how users perform a number tasks using our tool against the performing the same tasks using another baseline visualization tool.

References

[1] VirusShare, "VirusShare.com - Because Sharing is Caring," https://virusshare.com/.

[2] D. Plohmann, M. Clauss, S. Enders, and E. Padilla, "Malpedia: A Collaborative Effort to Inventorize the Malware Landscape," *The Journal on Cybercrime & Digital Investigations*, vol. 3, no. 1, 2018. [Online]. Available: https://journal.cecyf.fr/ojs/index.php/cybin/article/view/17

[3] A. Jain, H. Gonzalez, and N. Stakhanova, "Enriching reverse engineering through visual exploration of Android binaries," in *Proceedings of the 5th Program Protection and Reverse Engineering Workshop*, ser. PPREW-5. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2843859.2843866

[4] J. Jenkins and H. Cai, "Dissecting Android Inter-Component Communications via Interactive Visual Explorations," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 519–523.

[5] G. R. Santhanam, B. Holland, S. Kothari, and J. Mathews, "Interactive Visualization Toolbox to Detect Sophisticated Android Malware," in *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2017, pp. T–4 (1–8).

[6] J.-F. Lalande, M. Simon, and V. Viet Triem Tong, "GroDDViewer: Dynamic Dual View of Android Malware," in *The Seventh International Workshop on Graphical Models for Security*. Virtual Conference, France: Springer, Jun. 2020. [Online]. Available: https://hal-centralesupelec.archives-ouvertes.fr/hal-02913112

[7] V. T. Nguyen, A. S. Namin, and T. Dang, "Malviz: An interactive visualization tool for tracing malware," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 376379. [Online]. Available: https://doi.org/10.1145/3213846.3229501

[8] M. Angelini, G. Blasilli, L. Borzacchiello, E. Coppa, D. C. DElia, C. Demetrescu, S. Lenti, S. Nicchi, and G. Santucci, "SymNav: Visually Assisting Symbolic Execution," in *2019 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2019, pp. P–9 (1–11).

[9] J. Saxe, D. Mentis, and C. Greamo, "Visualization of Shared System Call Sequence Relationships in Large Malware Corpora," in *Proceedings of the Ninth International Symposium on Visualization for Cyber Security*, ser. VizSec '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 3340. [Online]. Available: https://doi.org/10.1145/2379690.2379695

[10] R. Gove, J. Saxe, S. Gold, A. Long, and G. Bergamo, "SEEM: A Scalable Visualization for Comparing Multiple Large Sets of Attributes for Malware Analysis," in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, ser. VizSec '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 7279. [Online]. Available: https://doi.org/10.1145/2671491.2671496

[11] T. Munzner, *Visualization Analysis and Design*, ser. AK Peters Visualization Series. CRC Press, 2015. [Online]. Available: https://books.google.de/books?id=NfkYCwAAQBAJ