MANIAC: A Man-Machine Collaborative System for Classifying Malware Author Groups

Eujeanne Kim
Hanyang University
Seoul, Republic of Korea
eujeanne@agape.hanyang.ac.kr

Sung-Jun Park
Hanyang University
Seoul, Republic of Korea
sjpark@agape.hanyang.ac.kr

Seokwoo Choi
The Attached Institute of ETRI
Daejeon, Republic of Korea
seogu.choi@gmail.com

Dong-Kyu Chae*
Hanyang University
Seoul, Republic of Korea
dongkyu@hanyang.ac.kr

Sang-Wook Kim*
Hanyang University
Seoul, Republic of Korea
wook@hanyang.ac.kr

ABSTRACT

In this demo, we show MANIAC, a **MAN**-machIne collaborative system for malware **A**uthor **C**lassification. It is developed to fight a number of "*author groups*" who have been generating lots of new malwares by sharing source code within a group and exploiting evasive schemes such as polymorphism and metamorphism. Notably, MANIAC allows users to intervene in the model's classification of malware authors with high *uncertainty*. It also provides effective interfaces and visualizations with users to achieve maximum classification accuracy with minimum human labor.

CCS CONCEPTS

• **Human-centered computing → Visualization systems and tools**; • **Security and privacy → Software and application security**.

KEYWORDS

Malware authors; malware classification; interactive classification

ACM Reference Format:
Eujeanne Kim, Sung-Jun Park, Seokwoo Choi, Dong-Kyu Chae, and Sang-Wook Kim. 2021. MANIAC: A Man-Machine Collaborative System for Classifying Malware Author Groups. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3460120.3485355

1 INTRODUCTION

Recently, a number of attacks have been made via malwares. They are growing in number more than ever in recent years, resulting in serious damages to home users, businesses, and governments. In addition, we note that there are many "groups" of malware authors such as APT (Advanced Persistent Threat) groups attacking governments or institutions according to the orders from specific countries [9, 11], and few efforts have been made to develop a defensive system against the author groups of malwares. The authors in the same group are suspected of sharing source codes for malwares and modifying the codes to create new malwares [3]. As such, new types of malwares can be easily developed and rapidly spread from a group of malware authors while avoiding the detection of existing anti-malware systems [9].

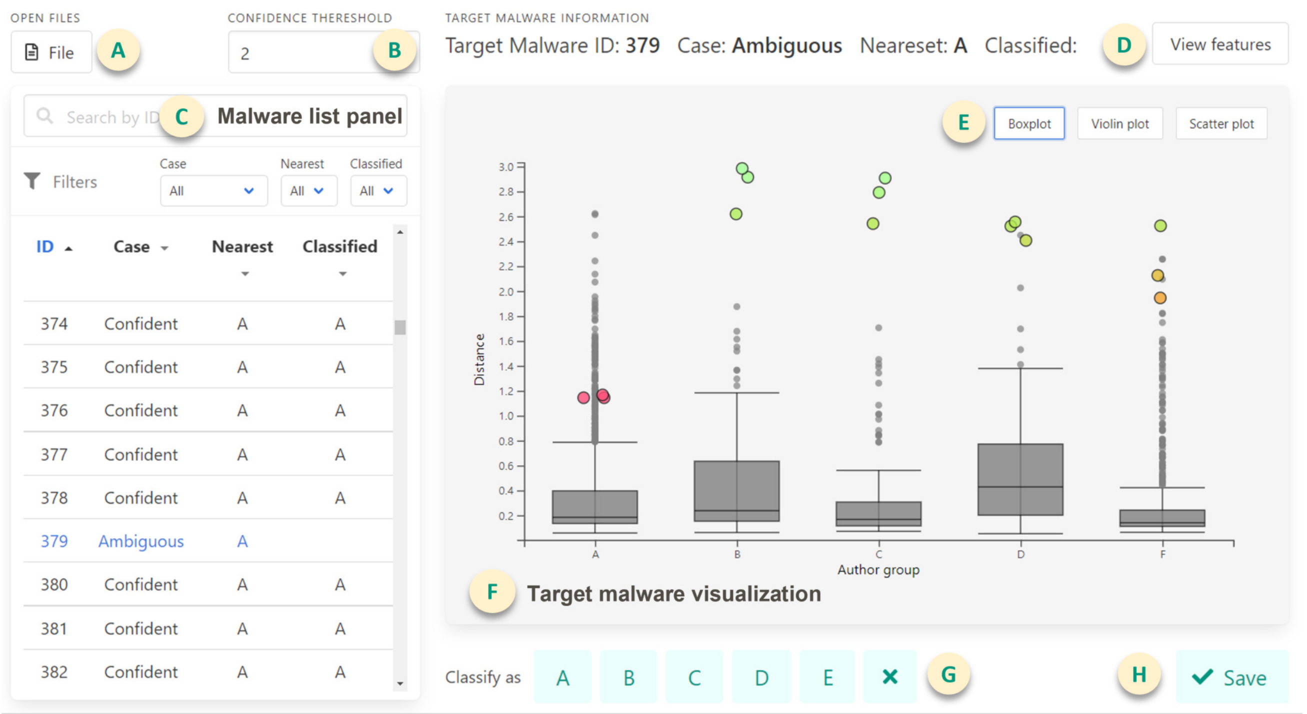
As an effective safeguard against the threat of malware spread by the author groups, we developed MANIAC, which stands for a **MAN**-machIne collaborative system for malware **A**uthor **C**lassification. It is a web-based system providing users with a graphical user interface (GUI) to enable them to upload their malwares in question to identify their author group and intervene in the classification process with the aid of MANIAC's interfaces and visualizations. MANIAC is for anyone who wants to identify the author groups of her malwares, but it is best suited for those working in the field of cybersecurity who have domain knowledge about malware. MANIAC includes the following distinct features compared with the existing malware classification tools: (1) It tackles the important problem of identifying author groups of malwares. (2) It allows the users to participate in the classification process to reduce the risk of misclassification of the model. (3) It provides informative visualization with the users which can lessen the labor of the users and give convincing explanations about a classification result. We confirmed the involvement of the users in the classification process results in meaningful improvement of the accuracy.

2 CLASSIFICATION SERVER

Before demonstrating MANIAC, we first introduce our *classification server* running in the backstage of MANIAC. This server takes the roles of (1) analyzing and embedding malwares uploaded by the users and (2) classifying the malwares based on the embeddings.

After the users upload the binary files of their malwares, the server performs both static and dynamic analysis on the binary to extract features. Here, we pre-defined specific features that would be important evidence of similar attack strategies, code reuse, and habitual patterns of an author group. The static features include the imports, exports, printable strings [4], and function chunks [1], and the dynamic features contain system API calls [2], network requests [7], addresses on the filesystem, registry keys, and the

*Co-corresponding authors


OPEN FILES: File

CONFIDENCE THERESHOLD: 2

TARGET MALWARE INFORMATION

Target Malware ID: **379** Case: **Ambiguous** Neareset: **A** Classified:

Filters: Case: All, Nearest: All, Classified: All

ID	Case	Nearest	Classified
374	Confident	A	A
375	Confident	A	A
376	Confident	A	A
377	Confident	A	A
378	Confident	A	A
379	Ambiguous	A	
380	Confident	A	A
381	Confident	A	A
382	Confident	A	A

Figure 1: A screenshot of MANIAC. (A): It is a button for uploading malware in question to the server. (B): It controls the confidence threshold. (C): The malware list panel summarizes the classification results returned from the server. It includes several interfaces allowing users to retrieve malwares having specific conditions using filters. (D): It presents features of the target malware. (E): It is for choosing the form of visual information. (F): This panel presents visual information of any selected malware from the (B) malware list panel. (G): It allows users to manually assign a label for ambiguous cases. (H): A button for saving the classification results to a user's local computer.

names of Mutexes [6]. Based on the static & dynamic analysis on it, each malware is represented as a set of those feature values.

Next, the server tries to learn good embeddings of malwares by using our training malware set[1] as well as the uploaded (test) malware set. The goal of this embedding is to locate malwares having similar characteristics (and thus likely to be created from the same author group) close to each other in the shared latent space. Towards this goal, following [9], the server first constructs a bipartite graph comprised of a malware part and a feature part where a malware node is connected to its extracted feature nodes. Finally, the well-known graph embedding model DeepWalk [10] is applied to the bipartite graph for learning embedding of each node (malware).

After obtaining the embedding of each malware, the server retrieves the nearest neighbor [8] of the uploaded malwares among those in the training set on the latent space. Then, it regards the label (author group) of each uploaded malware as corresponding nearest neighbor's author group, which was already labeled by our domain experts.[2] Furthermore, if the prediction is not confident, the server says "I don't know" rather than trying to predict something. This idea was motivated by our observation that there are some ambiguous cases where careful examinations by human experts are strongly required. Then the question is how to measure such confidence of the classification. First, we evaluate how the picked nearest neighbor is *close enough* to the target malware. This is evaluated according to the *distance distribution* among malwares belonging to the same author group with the chosen nearest neighbor: if the distance between the target malware and the nearest neighbor is an *outlier*, the confidence of classification is considered as low. Next, we also measure how the chosen nearest neighbor is *relatively close* to the target malware compared to other nearest neighbors belonging to the different author groups. If the distances from the target malware to (1) the chosen nearest malware and to (2) the second nearest malware in another author group are not different that much (i.e., lower than a certain threshold), then the confidence of classification is considered as low.

Consequently, if the confidence of classification of an uploaded malware from the two perspectives is low, the server considers it as an *ambiguous case* and does not try to predict something. After all the classifications are done, the server returns the results on all the uploaded malwares among which the confident cases are labeled as the predicted author groups and the ambiguous cases are marked as "Ambiguous" to the users.

[1]Each malware in the training set saved in the server is also represented as a set of the static & dynamic feature values.
[2]1,941 malware samples were collected where each one was labeled as one of five author groups 'A', 'B', 'C', 'D', and 'E'. There might exist other author groups, but MANIAC assigns one among the five labels to the (test) malwares.

3 DEMONSTRATION SCENARIOS

We now demonstrate how users can interact with MANIAC to classify their malwares. Figure 1 shows a screenshot of MANIAC. As a first step, users can select the binary files of malwares through the "File" button in the header of the GUI (see **A** in Figure 1). Then the selected files will be uploaded on the classification server and will be analyzed, embedded, and classified. After then, the users can see the classification results of the uploaded malwares in the left panel of the GUI (see **C** in Figure 1). Here, each row corresponds to each malware, and the titles of each column "Case", "Nearest", and "Classified" indicate whether the classification of this malware is confident, the author group of the nearest neighbor of this malware, and the predicted label of this malware, respectively. For example, the malware whose ID is 379 is an ambiguous case judged by the server so its column named "Classified" is empty even though its nearest neighbor is belonging to the author group 'A'.

As a next step, users can manually classify the author group of ambiguous cases. Users can sort out such cases by filtering the list whose "Case" value is "Ambiguous". After then, if users click each unclassified malware, the visualized information and the classification buttons will be presented on the right and bottom sides of the GUI (see **F** and **G** in Figure 1), respectively. Users can select the form of the visual information among Boxplot, Violin plot, and Scatter plot [5]. **F** shows the example of the box plot form, which illustrates the distance distribution (i.e., black boxes) where outliers are plotted as individual black points. The three colored points in each group are the top-3 nearest neighbors of the clicked malware from each author group. The color spectrum of the dots indicates the distance to the neighbor.

After carefully analyzing the visualized information, users can label the chosen malware by clicking one of the six buttons from 'A' to 'X' shown at the bottom, where 'A' to 'E' corresponds to the five author groups and 'X' corresponds to "unknown"[3]. Let's take a look at **F** in Figure 1, which is an example of visualized information of an ambiguous case. Here, its nearest neighbor which belongs to author group 'A' is actually an outlier based on the box-plot analysis. Also, the distances from the target malware to the nearest neighbors of each group does not differ very greatly. So it was judged as an ambiguous case by the server. However, by comparing the distance to the three nearest neighbors within each group based on the visualized information, the users will be able to realize that this malware should be classified as the author group 'A' since the distances to the three nearest neighbors of group 'A' are much closer than the distances to the three nearest neighbors of the other four groups. Thanks to the visualization, the users can make a quick decision without the time-consuming manual analysis.

In contrast, the users can still take the advantage of the visualization of confident cases. For example, some users may be worried about the fact that there could be misclassifications. Or some other users would want to inspect some malwares in their specific interest. Our visual information can assist such users by providing convincing evidence and explanations about the server's decision.

[3]Note that the users are also allowed to *give up* the prediction if it is still ambiguous even if its visualized information is provided. Such malware will later be further analyzed by manual inspection of its binary file.

Table 1: Accuracy comparisons

	micro-F1	**macro-F1**
MANIAC with human expert 1	95.60%	88.07%
MANIAC with human expert 2	95.75%	88.27%
MANIAC with human expert 3	95.92%	88.25%
MANIAC with majority vote	96.17%	89.41%
MANIAC only	93.71%	85.05%

4 PERFORMANCE EVALUATION

To demonstrate whether MANIAC is effective in classifying author groups more accurately, we employed three human experts and guided each expert to use MANIAC. Our evaluation follows the well-known *leave-one-out cross-validation* protocol; more specifically, each expert picked one malware from our dataset including 1,941 malwares, and uploaded it on the classification server. The rest 1,940 malwares were then used as a training set. Then the server returned the classification result, and if it turned out to be an ambiguous case, then each expert manually classified the uploaded malware with the help of MANIAC. The majority vote of the three experts was also tested here. After repeating 1,941 times of this task, we computed the F1-Scores for the five classes (i.e., author groups). Table 1 shows the macro-F1 and micro-F1 scores of the classification. For comparison, we also showed MANIAC's classification without collaboration with human experts. In this case, MANIAC tried to classify all the given malwares by itself and did not return any ambiguous case. We observed that MANIAC collaborated with human experts achieved much higher accuracy compared to that obtained only from MANIAC.

5 ACKNOWLEDGMENTS

This work was supported by Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2020-0-01373, Artificial Intelligence Graduate School Program (Hanyang University)). This work was also supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(*MSIT) (No. NRF-2020R1A2B5B03001960 and No.2018R1A5A7059549). *Ministry of Science and ICT

REFERENCES

[1] D. Bilar. 2007. Opcodes as predictor for malware. *International Journal of Electronic Security and Digital Forensics* 1, 2 (2007), 156–168.
[2] D.-K. Chae et al. 2013. Software plagiarism detection: a graph-based approach. In *ACM CIKM*. 1577–1580.
[3] S. Chakkaravarthy, D. Sangeetha, and V. Vaidehi. 2019. A Survey on malware analysis and mitigation techniques. *Computer Science Review* 32 (2019), 1–23.
[4] G. Costantini, P. Ferrara, and A. Cortesi. 2011. Static analysis of string values. In *International Conference on Formal Engineering Methods*. 505–521.
[5] F. M. Dekking et al. 2005. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media.
[6] M. Egele et al. 2008. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)* 44, 2 (2008), 1–42.
[7] A. Grégio et al. 2011. Behavioral analysis of malicious code through network traffic and system call monitoring. 8059 (2011), 80590O.
[8] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.
[9] J. Hong et al. 2019. Malware classification for identifying author groups: a graph-based approach. In *ACM RACS*. 169–174.
[10] B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. In *ACM SIGKDD*. 701–710.
[11] D. Plohmann et al. 2017. Malpedia: a collaborative effort to inventorize the malware landscape. *Proceedings of the Botconf* (2017).