Rule Creation in a Knowledge-assisted Visual Analytics Prototype for Malware Analysis

Johannes Schick[1], Markus Wagner[2], Niklas Thür[2], Christina Niederer[2], Gernot Rottermanner[2], Paul Tavolato[2], Wolfgang Aigner[2]
St. Pölten University of Applied Sciences, Austria
Email: [1]dm171566@fhstp.ac.at, [2]first.last@fhstp.ac.at

***Abstract*—The increasing number of malicious software (malware) requires domain experts to shift their analysis process towards more individualized approaches to acquire more information about unknown malware samples. KAMAS is a knowledge-assisted visual analytics prototype for behavioral malware analysis. It allows IT-security experts to categorize and store potentially harmful system call sequences (rules) in a knowledge database. To meet the increasing demand for individualization of analysis processes, analysts should be able to create individual rules. This paper is a visualization design study, which describes the design and implementation of a Rule Creation Area (RCA) into KAMAS and its evaluation by domain experts. It became clear that continuous integration of experts in interaction processes improves the knowledge generation mechanism of KAMAS. Additionally, the outcome of the evaluation revealed that there is a demand for adjustment and re-usage of already stored rules in the RCA.**

I. Introduction

Nowadays, domain experts have to deal with an ever increasing number of malicious software (malware) which in addition is becoming more targeted, persistent and unknown. Malwares are able to disturb computer operations and gather personal information of the system's owner without raising attention [1] [2]. When it comes to analyzing methods, there are two approaches for the identification of malware. On the one hand, software can be analyzed without actually executing it, which is called static analysis. Obfuscation techniques used by malware developers can render this task virtually impossible [3]. Dynamic analysis observes actions performed by potential malware while it is being executed in a protected environment. More precisely, analysts observe execution traces of programs; for the sake of simplicity malware analysts often reduce these traces to function calls, neglecting all other simple machine instructions. Therefore, dynamic analysis is also known as behavior-based analysis [3].

In behavior-based analysis malware analysts have to deal with large amounts of data, which can lead to a very complex analysis process: a trace of a malware sample may often comprise thousands of system calls and analysts have to find similar system call patterns within thousands of such traces. In order to simplify this process, analysts need automated approaches for finding such patterns and categorizing them as potentially harmful or harmless. However, such identification of patterns relies heavily on the analysts knowledge, which makes it impossible to automate this process completely [4]. These patterns of behaviors can be defined as a formal language using formal grammars (syntactic pattern recognition [5], [6] or for more details [7]). The task of the analyst is the development of a set of grammar rules incorporating their knowledge about (malicious) behaviors of malware samples. In this context, visual analytics (VA) is needed to support the analysts in integrating their knowledge. VA plays an essential role in supporting data analysis, since it combines data processing capabilities of computer systems with the knowledge and experience of users [8].

According to Keim et al. [9], VA also connects automated analysis techniques with interactive visualizations in order to combine different types of information and obtain understanding from complex data sets. To make reasoning out of this massive amount of data, it is necessary to include "implicit" [10] or "tacit" [11] knowledge in the analysis process. By externalizing the implicit/tacit knowledge of domain experts, it is possible to provide explicit knowledge in form of data, which is independent from the current user of the system. This extracted knowledge can subsequently be connected through interactive visualization tools [11]. In addition to these findings, Lee et al. [12] stated that visualization is necessary to analyze potential malware more effectively.

This paper provides a design study [13] dealing with the implementation and evaluation of a separate Rule Creation Area (RCA) [14] into a Knowledge-Assisted Visual Malware Analaysis System (KAMAS) [15]. In order to meet quality standards, this paper follows a problem-oriented research approach. In conjunction with this prototype, this means that the visualization and implementation of the system is performed under consideration of specific real-world problems defined by domain experts [13]. Thus, the main objectives of this research are:

- Clarify, why the implementation of a separate area for rule creation is necessary and how malware analysts can benefit from it.
- Presenting the design and implementation of the RCA into the KAMAS system with a detailed description of all involved components and functionalities.
- Conducting an evaluation of the implemented system in cooperation with malware analysis experts to proof the effectiveness of the deployed functionalities.
- Reflection of the implemented functionalities under consideration of their evaluation by real world users and the resulting future research.

II. Related Work and Background

Since there were no interactive visualization tools available which cover all requirements for malware analysts, Wagner et al. [15] developed the KAMAS prototype. With KAMAS, analysts are able to categorize function call traces in terms of their potential harmfulness and store them into a knowledge database (KDB). The KDB assists them in further observation and simplifies the analyzing process. In order to expand the range of functionalities and subsequently improve the effectiveness of KAMAS, Wagner et al. [16] suggested an interface design for the RCA [14]. The RCA allows the construction of completely new rules by using single system and API calls in the same structure as generated by the sequitur algorithm [17]. These rules can subsequently be stored in the KDB.

Knowledge Generation in VA: Thomas and Cook [18] define VA as approach to gain knowledge from massive, dynamic, ambiguous, and often conflicting data. Based on the research by Sacha et al. [19] following findings can be determined: VA uses data to draw conclusions on a specific application field and gain insights into the problem domain. On the one hand, there is the combination of perceptive skills following the capability of drawing conclusions by humans. On the other hand, there is the computing and data storage capability of machines. Both of these aspects can be combined in visual representations. Interactions with VA tools provide a possibility for analysts to detect patterns in their data, thus assisting them in verifying or falsifying their initial hypothesis. By clustering and classifying the found patterns, the outcome of their exploration can be visualized.

According to Chen et al. [10], the aim of knowledge-assisted visualization is to automate reasoning about abstracted information from a set of data. Additionally, they also mentioned that the field of knowledge-assisted visualization is still in development, since the growing amount of data requires systems to continuously adapt to these challenges. As stated by Wang et al. [11], the nature of tacit knowledge can be defined as intimate and specialized. However, by using interactive visualization it can be connected with explicit knowledge, which is processable by computers or can be stored in a database [11]. Recent research has focused even more on the role of humans in this process. In order to deal with the increasingly ambitious challenges in the field of VA, the philosophy has to shift from a ”human in the loop” philosophy to a ”human is the loop” viewpoint [20]. This new approach focuses on recognizing the workflow of analysts and consequently adapting interaction processes to the needs of the analyst.

Appliance of VA Techniques to Malware Analysis: According to Alazab et al. [21], all executable programs have the aim to perform actions using API calls. The process of malware analysis involves the observation of system call sequence patterns and the actions they cause. Both Alazab et al. [21] and Mohaisen et al. [22] emphasize classification and clustering of patterns in terms of their maliciousness or benignity as a main task for malware analysts. AMAL, a behavior-based malware analysis system by Mohaisen et al. [22] is an example for a program, which is capable of fulfilling this task. Just like KAMAS, it tries to tackle shortcomings of existing systems by combining methodologies of static and behavior-based approaches. By running malware samples in a virtualized environment, the system collects data which is subsequently used for automated classifying and clustering of samples into different malware families. However, AMAL does not provide an interactive user interface, nor does it provide the possibility to integrate externalized expert knowledge into the analysis process. Another project with similar approach to KAMAS is the visualization tool SEEM [23], which enables analysts to compare large sets of malware and their associated attributes.

As mentioned in Section I, supportive visualization is needed in order to provide a more efficient approach of analyzing potential malware samples [12]. In their state of the art report on visualization systems in the field of malware analysis, Wagner et al. [24] concluded that future systems should provide a compound of classification overviews for comparison and detail views for individual analysis.

III. Method

In general, this paper is a design study following the design principles/ideas proposed in [13], which is described as problem-orientated research approach. This includes a problem definition, the design and implementation of a visualization system which solves the problem, the evaluation of the prototype as well as a reflection about lessons learned and possible improvements [13]. The problem was defined by Wagner et al. [16] in their design study, which addresses the need for the implementation of a separate area for rule creation in the KAMAS prototype. All scientific publications directly related to KAMAS [4] [16] [15] [24] served as a basis for the general understanding of the prototype and its background.

Requirements & Features: The functionalities and interface design of the RCA were designed according to the rule building screen prototype ’CallNet’ presented in [16]. Furthermore, ’CallNet’ and its desired functionalities were already reviewed by usability experts. In order to ensure compatibility with the sequitur algorithm [17], the extracted knowledge has to be stored in a rule-based interface and structure. Based on the task definitions and the outcome of the design study, following key requirements (R) for the RCA can be defined:

- **R1 Consistency:** To ensure an effective usage of the user interface, it is necessary to provide consistent interaction techniques throughout the whole system. In this specific case, the input data for the RCA originates from another interface section of the program. Therefore, the interaction visualization should be related to the movement of data, e.g. ’Drag & Drop’ operations.
- **R2 Creation Support:** The amount of data offered by this system is particularly high. Thus, additional support in the process of rule creation is important. By giving the analyst additional interaction possibilities, e.g. automatically validated suggestions for single calls, the rule creation process can be accelerated.

Moreover the interface has to provide the possibility to switch the highlighting of these calls based on higher or lower frequency to support creating rules with individual preferences.

R3 Editing Options: The process of rule creation requires the system to allow the editing of rules at any time and to offer a quick way to restart the process. As a consequence, the analyst has to be able to reorder and delete single calls of the dropped call sequence and to reset the whole RCA to its default state.

R4 Knowledge Extraction and Extension: Finally, the newly created rule should be used to extend the spectrum of computerized knowledge in the system. Therefore, it is necessary to offer the possibility of moving rules from the RCA to the KDB. By implementing this functionality, the knowledge generation loop (see Figure 2) can be expanded, which should subsequently improve the effectiveness of the analysis process.

The features of the RCA were implemented according to the defined key requirements. In general, the design and implementation followed a user-centered design process [25]. During the development process, continuous exchange with researchers/developers of the KAMAS prototype was performed. Thus, it was possible to adjust requirements and discuss alternative solutions.

Evaluation: In order to evaluate the implemented features of the prototype, two malware analysis experts reviewed the system in the course of a semi-structured, qualitative user test. During this test, both experts had to solve different tasks, which occur in the rule creation process. The results were documented by written notes and afterwards categorized based on their importance. Afterwards, the results of the evaluation were summarized and rated in a list inspired by Nielsen's severity rating procedure [26]. With these ratings, it was possible to provide a clear overview of the most important findings as well as potentially negligible aspects.

IV. Design and Implementation

The design and functionalities of the RCA are based on the 'CallNet' prototype [16], which allows users to create rules from scratch with system and API calls. The KAMAS prototype and its implemented functionalities developed by Wagner et al. [15] served as a basis to expand the prototype's spectrum of features. The implementation resulted in the realization of the RCA (see Figure 1), which was achieved using the programming language Java.

Call Exploration: The 'Call Exploration' table (see Figure 1.3) provides a list of all system and API calls of the loaded file showing their occurrence in the file, the name and the ID of the call. In addition to the already available functionalities from [15], the possibility to drag single calls from the table to the RCA was implemented.

KDB: The KDB (see Figure 1.1) offers the possibility to save and organize rules in tree structure based concepts visualized as folder structure. Furthermore, the analyst can access information of already stored rules like the name, the assigned concept and the calls it consists of.

RCA in General: The RCA (see Figure 1.2) generally consists of three main areas. First, the analyst can drop single calls, which he previously selected and dragged from the 'Call Exploration' table into the Rule Creation Table (RCT) (see Figure 1.2.b). Secondly, above and below the RCT, the interface provides suggestions for single calls which occur either before (see Figure 1.2.a) or after (see Figure 1.2.c) the dropped system call sequence. At last, on the bottom of the RCA the analyst has the possibility to reset the whole RCA to its default state (see Figure 1.2.d) and to switch the highlighting of the call suggestions (see Figure 1.2.e).

Rule Creation Table in the RCA: After adding the first call from the 'Call Exploration' table to the RCT, an additional row gets added on the top of the table. This row makes it possible to drag the newly created rule (which contains all single calls inside the RCT) and add it to the KDB. Furthermore, the number in the second column of the RCT represents the occurrence of the newly created rule in the loaded file. If there is a need to reorder calls inside the RCT, this can be achieved by simply dragging a single call and move it to the desired position. The original call from the desired position then switches position with the dragged call. Also, single calls can be deleted from the RCT by right clicking on the desired call and using the 'Delete' pop-up. It must also be pointed out that every interaction performed in the RCT affects the occurrence column and call suggestions, since these components depend on the values inside the RCT. Even though a rule usually can contain 1 to n calls, the maximum number of calls inside the RCT was limited to eight calls in order to provide enough space for the other areas in the RCA.

Call Suggestions in the RCA: Above and below the RCT, the interface offers suggestions for calls, which can be dragged and dropped into the RCT. The suggestions above (see Figure 1.2.a) represent calls from the loaded file which occur before system call sequences with the same structure as the one inside the RCT, whereas the calls below (see Figure 1.2.c) represent calls which occur after the currently dropped system call sequence. Moreover, the font size of the call suggestions varies depending on their occurrence. By default, more frequent single calls are displayed with a bigger font. If a single call appears in multiple system call sequences of the loaded file, the font size increases by one for every found similar single call. Thus, every call suggestion displayed in the user interface is unique and the analyst gets a better overview of which single calls are more or less frequent.

Control Buttons in the RCA: Provided that the RCT contains at least one single call, a 'Reset' button (see Figure 1.2.d) is available at the bottom of the RCA. This button offers the possibility to set the whole area back to its default state. The second button (see Figure 1.2.e) is responsible for handling the highlighting of the call suggestions and is only visible when the currently dropped system call sequence offers suggestions. With the use of this button, the analyst can switch between highlighting more or less frequent call suggestions.

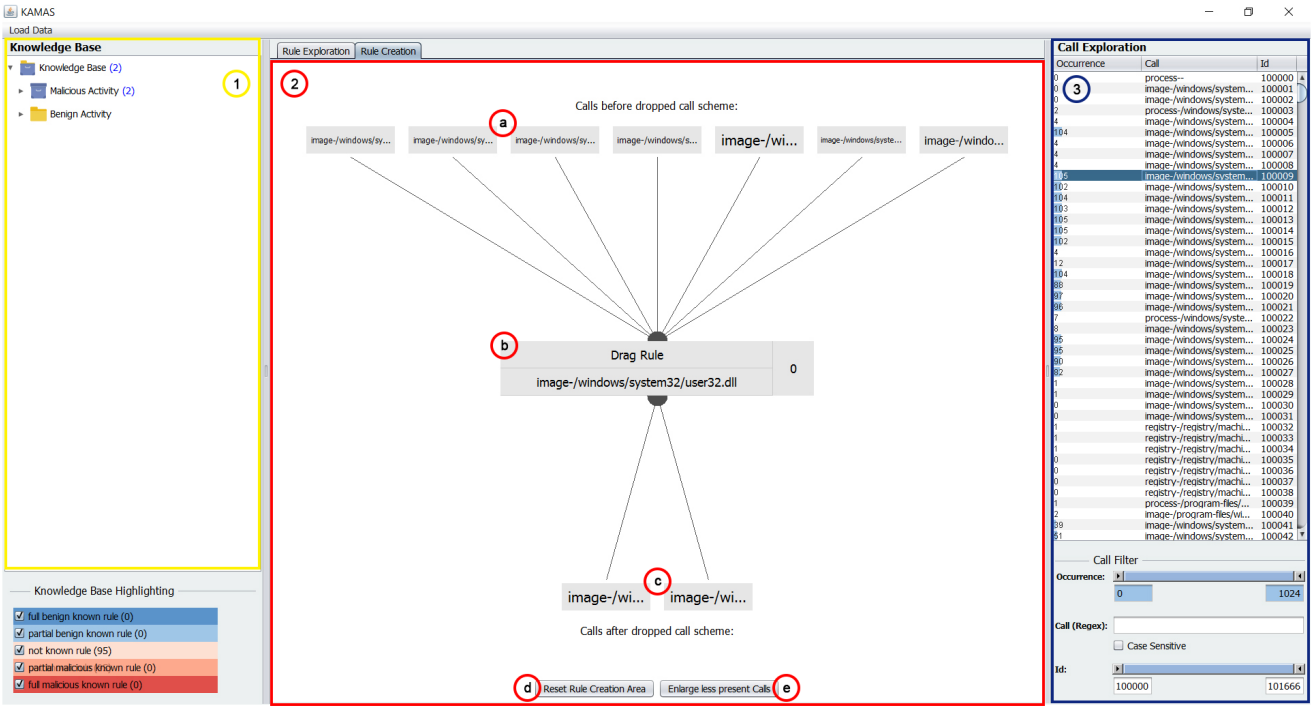

Fig. 1. User interface of the KAMAS prototype with activated RCA. 1) The KDB where newly created rules can be stored. 2) The RCA including the call suggestions before (2.a) and after (2.c) the currently dropped call sequence, the rule creation table (2.b), the button to reset the whole RCA (2.d) and the button to change the call suggestion size according to their occurrence (2.e). 3) The 'Call Exploration' table with a list of all single calls included in the currently loaded file.

Filter Pipelines for Call Suggestions: Depending on the currently dropped system call sequence, the call suggestions get validated through one general and two separated filter action pipelines. The general pipeline loops through all rules of the loaded file and eliminates every rule which does not include the exact same system call sequence as in the RCT. In the next step, the remaining rules serve as input data for the call suggestion validation which is finally displayed above and below the RCT. The first pipeline loops through every remaining rule and checks if there is another single call before the first one of the sequence. If so, this call is displayed as call suggestion above the RCT in the user interface. The same applies to the second pipeline, except that it extracts the single call after the last one of the sequence which is subsequently displayed below the RCT.

Usage Scenario: First, the analyst loads a new file into the system and KAMAS automatically provides an overview of all included single calls in the 'Call Explorer' (see Figure 1.3) as well as all preprocessed rules in the 'Rule Explorer'. The 'Rule Explorer' serves as a graphical summary and exploration area with colored highlighting of all included rules depending on the current knowledge state of the KDB. If the analyst wants to store one of these rules in KDB, this can either be achieved by selecting the full rule or just specific parts of this rule and drag and dropping it to the KDB. Further functionalities linked to the 'Rule Explorer' are described by Wagner et al. [15]. In the initial version of the KAMAS prototype, the rule storing process was limited to use either preprocessed rules or their included single calls. Thus, the analyst was not able to change patterns like e.g. the order of included single calls inside a rule. Following the implementation of the RCA, the analyst can now switch to the 'Rule Creation' screen and create own rules from scratch. In the beginning, the analyst can explore and select specific calls from the single call table and drag them into the RCA. After the desired calls were added to RCT, the interface offers suggestions for calls which occur before and after the currently dropped call sequence. These calls can also be used in the further creation process by dragging them into the RCT. Additionally, it is possible to highlight either more or less frequent call suggestions by increasing their font size. This can be achieved by clicking the 'Enlarge less/more frequent calls' button. The number next to the calls inside the RCT represents the occurrence of the currently dropped rule in the analysis file. During the whole process, the analyst can adjust the created rule by reordering calls inside the RCT via drag and drop or deleting unnecessary calls via right clicking on the desired call and using the 'Delete' pop-up. Finally, the rule can be dragged at the top of the RCT and moved to the KDB. Afterwards the RCA can be reset to its default state by clicking the 'Reset Rule Creation Area' button. The analyst can now return to the 'Rule Exploration' screen and continue the analysis with an updated KDB containing the newly created rule.

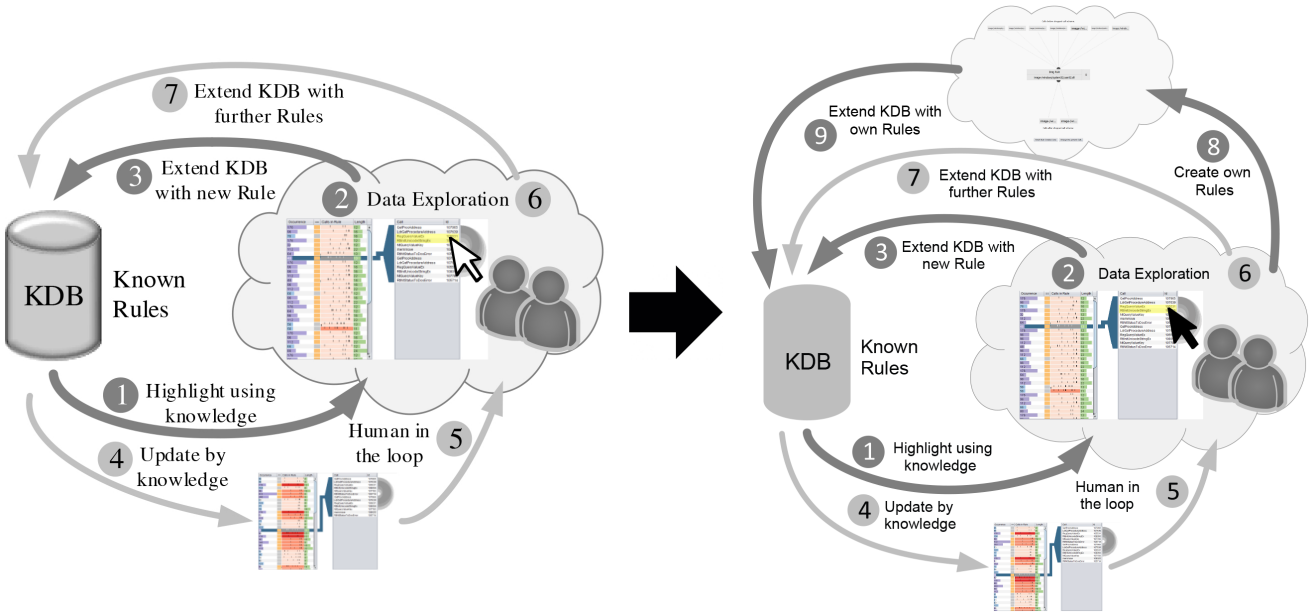

Fig. 2. Comparison of the Knowledge Generation Loop before and after the implementation of the RCA. **Left**: Knowledge Generation Loop of the initial KAMAS prototype visualized by Wagner et al. [15]. **Right**: Knowledge Generation Loop including rule creation process. The range of possibilities for the analyst has increased.

V. The Knowledge Generation Loop

Initial Knowledge Generation Loop: Wagner et al. [15] provided an overview of the knowledge generation process in the KAMAS system (see Figure 2 on the left). It highlights the KDB, which contains all known rules and the data exploration through the analysts as central elements in the knowledge generation loop. By extending the KDB with new rules, the system automatically revalidates the highlighting in the 'Rule Exploration' area depending on the new knowledge state. The rules used for the extension of the KDB are automatically generated and displayed in the 'Rule Exploration' table.

Extension of Knowledge Generation Loop: With the implementation of the RCA, the possibilities for knowledge generation have expanded. The usage scenario now includes the following process (see Figure 2 on the right): The analyst can load a new file, which is going to be checked automatically against the stored data in the KDB (see Figure 2.1). Afterwards, the system provides a visualization of the loaded rules in comparison to its current knowledge state in the system (see Figure 2.2). In contrast to the initial KAMAS prototype, the analyst can now choose between two different options to expand the KDB. The analyst can either use automatically generated rules from the 'Rule Exploration' table (see Figure 2.3 and Figure 2.7), or create own rules, which can include presently unknown sequences based on the system's current state (see Figure 2.8 for the creation and Figure 2.9 for the knowledge generation). As a result, the KAMAS prototype is not limited to rely on automatically generated rules anymore, but rather offers more flexibility by providing an opportunity to create rules from scratch based on individual needs.

VI. Evaluation

As third step in this design study [13], it was necessary to evaluate the newly implemented functionalities with real world users. Therefore, a formative usability evaluation including a usability test with malware analysis experts was conducted and finally compressed to the most important findings.

A. Method

TABLE I
Overview of the domain experts who participated in the user test. (E = expert)

Person	Age	Gender	In field	Experience	Education
E1	30-39	male	5 years	expert	MSc
E2	60-69	male	6 years	advanced	PhD

Participants: For this user test, two malware analysis experts (see Table I) were invited to test and comment all functionalities of the RCA. Both experts had more than five years of experience in this field and were also part of previous KAMAS case studies. Therefore, both of them were familiar with the general appearance and functionality of the user interface.

Design and Procedure: In the beginning, the experts received a brief overview about the main functionalities of the RCA. In order to provide a realistic testing environment, an execution trace sample was provided and loaded into the system. Then, the experts were asked to test each possible feature and to speak out their thoughts on the user interface.

Apparatus and Materials: Both case studies were performed in a seminar room. To guarantee the testing of all possible features, a questionnaire based on the functionalities of the

TABLE II
OVERVIEW OF EVALUATED FEATURE REQUESTS, SEVERITIES AND EFFORT (FR: 1: = NICE TO HAVE, 2: = GOOD FEATURE, 3: = ENHANCES USABILITY; SE: 0: = NO PROBLEM, 1: = COSMETIC PROBLEM, 2: = MINOR PROBLEM, 3: = MAJOR PROBLEM, 4: = CATASTROPHE; EFFORT: 1: = MIN, 2: = AVERAGE, 3: = MAX).

Description	Feature Request (FR)	Severity (SE)	Effort
Call Exploration: Change selection mode to single selection	2	3	1
Call Exploration: Display only the last part of single call names	2	3	1
RCT: Provide a clear visualization of the drop location	3	3	2
RCT: Add arrow symbols to visualize the possibility of reordering	1	1	1
RCT: Display only the last part of single call names	2	3	1
Enlarge-Button: Change 'Enlarge' to 'Highlight'	1	1	1
Call Suggestions: Display current highlighting status in separate label	1	1	1
Call Suggestions: Change 'before/after' to 'which appear before/after'	1	1	1
Call Suggestions: Change 'scheme' to 'sequence'	1	1	1
KDB: Implement possibility to use known rules in the RCA	3	3	3
RCA: Display name of rule if it is already stored in the KDB	2	3	2
RCA: Implement a separate save button	1	2	2
Connection lines: Use logical elements	1	0	2

system was provided. The comments stated by the experts were documented by notes on the printed questionnaire.

B. Results

Moving Single Calls to the RCT: E2 mentioned that it was difficult to find the desired calls, since the names of the displayed single calls were not fully readable due to lack of space in the 'Call Exploration' table. He suggested to display only the last part of the names because this part mostly differs from other single call names. Both of them showed uncertainty regarding whether if it is possible to move more than one call at the same time or not. Consequently, they were not sure if their desired calls were correctly dropped into the RCT. E2 suggested to change the selection mode of the 'Call Exploration' table to single selection when the RCA is activated in order to avoid misconceptions.

Moving Call Suggestions to the RCT: While they tried to move the calls to the RCT, both of them were confused by the drop location inside the RCT. They stated that they were not able to recognize a drop scheme, which led to the assumption that the calls get randomly added to the table.

Reordering of Calls Inside the RCT: E1 did not recognize the possibility of reordering in the first place and suggested to add arrow symbols near the table cells of the RCT in order to make it more clear. Furthermore, E2 mentioned that long call names could get cropped off inside the table cells. Thus, the reordering of multiple calls with the same structure could be difficult to recognize.

Change Highlighting of Call Suggestions: Both experts had problems understanding the naming convention of the button. Since the word 'Enlarge' is present in both versions of this button, E2 expected the call suggestions displayed above and below the RCT to extend to the follow-up call. Furthermore, he suggested to change the description linked to the call suggestions to 'Calls which appear before/after dropped call sequence' to clarify the meaning. E1 added to show the current highlighting state in a separate label to provide a more clear appearance for the user.

Deletion of Single Calls and Reset of RCA: Both experts were able to delete single calls from the RCT and reset of the whole RCA to its default state without any uncertainty.

Adding Created Rule to KDB: Both experts expressed their wish for a possibility to drag rules from the KDB into the RCA. Additionally, the RCA should display the name in a label when editing an already created rule. In order to provide an alternative for the drag and drop approach, E1 suggested to implement a separate save button for the storing of rules into the KDB.

General Exploration: Both experts were pleased with the general appearance of the user interface. They found the functionalities to be valuable and the interface easy to understand. Furthermore, the simplicity of the user interface was rated positively.

C. Rating

Based on the experts' comments, the exploration results were combined and rated in a list of the most important issues (see Table II). The rating procedure in this list is inspired by Nielsen's severity ratings [26]. It includes a description of the issue, feature requests (FR), severities (SE) as well as the associated effort for the solution of the issue. The conducted rating is illustrated in Table II.

Summary: The conducted evaluation showed that the implemented functionalities were well received by the domain experts, although there are still certain improvements to consider for the future. By rating found issues and suggested improvements (see Table II), it was possible to determine major areas for further development of the current prototype.

VII. LIMITATIONS

Following the evaluation by malware analysis experts, certain limitations in the scope of functionalities for this prototype can be determined:

Adjustment of Stored Rules: This prototype does not provide a possibility to drag already stored rules from the KDB into the RCA. However, the workflow of malware analysts also includes the manual adaption of already found rules [4]. With

the implementation of this feature, it would be possible to cover all essential needs of malware analysts and subsequently improve the analysis process even more.

Displaying of Rule Names: The process of rule creation can also lead to a situation, where analysts are constructing rules, which are already stored in the KDB. However, the RCA is not able to recognize already known rules and consequently does not provide the rule name in the user interface. By enabling the RCA to check the currently constructed rule against the KDB and subsequently recognize known rules, the workflow of analysts can be enhanced by e.g. preventing the storage of duplicates in the KDB.

Creating Rules with More than Eight Single Calls: The RCA offers the possibility to create rules with up to eight single calls. Nevertheless, rules can contain much more single calls in reality. Since the RCA also provides call suggestions above and below the RCT, the capacity of space in the RCA is rather limited. To overcome this, for example Focus+Context and/or aggregation techniques could be applied.

Drop Location Visualization: As mentioned in Section VI, the RCA does not provide a visual preview of the currently dragged single call in the RCT. Since both experts were struggling with this issue, the implementation of a visual preview of the dragged single call would have enormous potential for improving the quality of the user interface.

VIII. Reflection and Conclusion

In order to complete the methodology of Sedlmair et al. [13], this section focuses on the reflection of the combined results emerging from the design and implementation of the prototype and its evaluation by real world users. The requirements (R1 - R4) described in Section III were omnipresent during all steps in this design study and serve as point of reference for the following reflection.

R1 Consistency: In order to stick to the defined requirements, drag and drop operations served as the major interaction technique in this prototype. This involves the addition of single calls and call suggestions to the RCT, the reordering of calls inside the RCT as well as the storing of the created rule in the KDB. Both analysts were comfortable with the handling of the given interaction possibilities. However, the evaluation showed that additional visualization is needed to make the outcome of drag and drop operations fully transparent.

R2 Creation Support: As mentioned in the beginning, analysts have to deal with a large amount of data during the exploration process. Therefore, the implemented prototype provides call suggestions to accelerate and simplify the rule creation process. Based on the currently dropped call sequence, the previously described filter pipeline (see Section IV) validates the displayed call suggestions automatically. Additionally, the prototype offers a possibility to highlight more or less frequent call suggestions, which assists analysts in their decision making process.

R3 Editing Options: To ensure editability during the rule creation process, the prototype provides possibilities to delete and reorder single calls in the RCT as well as a button to restart the whole process from scratch. A particularly interesting outcome of the evaluation was that both experts expressed their wish for reusing/adjusting already stored rules in the RCA. This aspect was not taken into account during the development of the current prototype version. After the evaluation, it can be considered as highly recommendable to implement this feature.

R4 Knowledge Extraction and Extension: The possibility to drag the newly created rule and store it in the KDB was also well received by the experts. As mentioned in Section IV, the implementation of this feature expands the knowledge generation loop (see Figure 2). Analysts are now more flexible when it comes to the extension of the KDB. By providing the possibility to create individual rules based on the experts current state of knowledge, the prototype shifts towards the in the beginning mentioned ”human is the loop” philosophy [20]. As a result, the overall knowledge generation process is getting more individualized and the following analysis process can draw upon different expertises.

Lessons Learned: In the course of this design study, it became clear that the continuous integration of domain experts in interaction processes enhances the efficiency of the analysis procedure. As the number of malware families is growing, higher importance has to be attached to the integration of expert knowledge [24]. At the same time, VA techniques have to adapt to the need for more human integration in the analysis process [20]. With the implementation of the RCA based on the interface design prototype by Wagner et al. [16], both previously mentioned challenges were tackled. In cooperation with malware analysis experts, the implemented prototype was proven to enhance the knowledge generation process and to handle the need for increasing focus on human interactions in VA. However, the evaluation also revealed that interaction visualization is a key factor for providing a satisfying solution. Additionally, it showed that there are still possibilities to improve the knowledge generation process. Subsequently, humans could be even more integrated into the previously mentioned knowledge generation loop. In this system, the knowledge is stored based on the same rule structure as they are generated by sequitur [17]. But the storage of knowledge depends on the structure of the underlying data. Thus, also value ranges or process structures can be used.

Future Work: The usage of already stored rules for rule creation can be seen as the next logical step for further development of the presented prototype. Additionally, the enhancement of interaction visualization should round off the overall appearance and usability of the user interface. In general, further exchange with malware analysis experts should be taken into account in order to stay on track with the developments in the scene.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) via the “KAVA-Time” project (P25489-N23). We would also like to thank all focus group members and test participants who have agreed to volunteer in this project.

REFERENCES

[1] T. Micro, "Addressing big data security challenges: The right tools for smart protection," *US: Trend Micro*, 2012.

[2] E. Gandotra, D. Bansal, and S. Sofat, "Malware Analysis and Classification: A Survey," *Journal of Information Security*, vol. 05, no. 02, p. 56, 2014.

[3] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A Survey on Automated Dynamic Malware-analysis Techniques and Tools," *ACM Computing Surveys*, vol. 44, no. 2, pp. 6:1–6:42, 2008.

[4] M. Wagner, W. Aigner, A. Rind, H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, "Problem Characterization and Abstraction for Visual Analytics in Behavior-based Malware Pattern Analysis," in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*. ACM, 2014, pp. 9–16.

[5] K. Fu, *Syntactic pattern recognition and applications*, ser. Prentice-Hall advanced reference series: Computer science. Prentice-Hall, 1982.

[6] R. Gonzalez and M. Thomason, *Syntactic pattern recognition: an introduction*. Addison-Wesley Publishing Company,Reading, MA, 1978.

[7] H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, "Malicious Behavior Patterns," in *IEEE International Symposium on Service Oriented System Engineering*, 2014, pp. 384–389.

[8] E. Kandogan and U. Engelke, "Agile Visual Analytics in Data Science Systems," in *IEEE International Conference on High Performance Computing and Communications; IEEE International Conference on Smart City; IEEE International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2016, pp. 1512–1519.

[9] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, *Mastering the Information Age Solving Problems with Visual Analytics*. Eurographics Association, 2010.

[10] M. Chen, D. Ebert, H. Hagen, R. S. Laramee, R. v. Liere, K. L. Ma, W. Ribarsky, G. Scheuermann, and D. Silver, "Data, Information, and Knowledge in Visualization," *IEEE Computer Graphics and Applications*, vol. 29, no. 1, pp. 12–19, 2009.

[11] X. Wang, D. H. Jeong, W. Dou, S.-W. Lee, W. Ribarsky, and R. Chang, "Defining and applying knowledge conversion processes to a visual analytics system," *Computers & Graphics*, vol. 33, no. 5, pp. 616–623, 2009.

[12] D. Lee, I. S. Song, K. J. Kim, and J. h. Jeong, "A Study on Malicious Codes Pattern Analysis Using Visualization," in *International Conference on Information Science and Applications*, 2011, pp. 1–5.

[13] M. Sedlmair, M. Meyer, and T. Munzner, "Design Study Methodology: Reflections from the Trenches and the Stacks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2431–2440, 2012.

[14] J. Schick, M. Wagner, N. Thür, C. Niederer, G. Rottermanner, P. Tavolato, and W. Aigner, "Supporting knowledge-assisted rule creation in a behavior-based malware analysis prototype," in *Poster of the 14th Workshop on Visualization for Cyber Security (VizSec)*, Phoenix, Arizona, USA, 2017.

[15] M. Wagner, A. Rind, N. Thür, and W. Aigner, "A knowledge-assisted visual malware analysis system: Design, validation, and reflection of kamas," *Computers & Security*, vol. 67, pp. 1–15, 2017.

[16] M. Wagner, A. Rind, G. Rottermanner, C. Niederer, and W. Aigner, "Knowledge-assisted rule building for malware analysis," in *Proceedings of the 10th Forschungsforum der österreichischen Fachhochschulen*, FH des BFI Wien. Vienna, Austria: FH des BFI Wien, 2016.

[17] R. Luh, G. Schramm, M. Wagner, and S. Schrittwieser, "Sequitur-based inference and analysis framework for malicious system behavior," in *Workshop for Formal Methods in Software Engineering (ForSE), 3rd International Conference on Information Systems Security and Privacy (ICISSP)*, SCITEPRESS Digital Library. Porto, Portugal: SCITEPRESS Digital Library, 2017, pp. 632–643.

[18] J. J. Thomas and K. A. Cook, "A visual analytics agenda," *IEEE Computer Graphics and Applications*, vol. 26, no. 1, pp. 10–13, 2006.

[19] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. A. Keim, "Knowledge Generation Model for Visual Analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1604–1613, 2014.

[20] A. Endert, M. S. Hossain, N. Ramakrishnan, C. North, P. Fiaux, and C. Andrews, "The human is the loop: new directions for visual analytics," *Journal of Intelligent Information Systems*, vol. 43, no. 3, pp. 411–435, 2014.

[21] M. Alazab, S. Venkataraman, and P. Watters, "Towards Understanding Malware Behaviour by the Extraction of API Calls," in *2010 Second Cybercrime and Trustworthy Computing Workshop*, 2010, pp. 52–59.

[22] A. Mohaisen, O. Alrawi, and M. Mohaisen, "AMAL: High-fidelity, behavior-based automated malware analysis and classification," *Computers & Security*, vol. 52, pp. 251–266, 2015.

[23] 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: ACM, 2014, pp. 72–79.

[24] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, and W. Aigner, "A survey of visualization systems for malware analysis," in *Eurographics Conference on Visualization (EuroVis) - STARs*, R. Borgo, F. Ganovelli, and I. Viola, Eds. Cagliari (Sardinia / Italy): The Eurographics Association, 2015, pp. 105–125.

[25] H. Sharp, Y. Rogers, and J. Preece, *Interaction Design: Beyond Human Computer Interaction*. John Wiley & Sons, 2007.

[26] J. Nielsen, *Usability Engineering*. Morgan Kaufmann Publishers Inc., 1993.