A Bigram Supported Generic Knowledge-Assisted Malware Analysis System: BiG2-KAMAS

Niklas Thür[1], Markus Wagner[1], Johannes Schick[1], Christina Niederer[1], Jürgen Eckel[3], Robert Luh[2], Wolfgang Aigner[1]
[1]Institute of Creative\Media/Technologies, St. Pölten University of Applied Sciences, Austria
[2]Josef Ressel Center for Unified Threat Intelligence on Targeted Attacks, Austria
[3]IKARUS Security Software GmbH, Austria
Email: [1,2]first.last@fhstp.ac.at, [3]eckel.j@ikarus.at

***Abstract*—Malicious software, short "malware", refers to software programs that are designed to cause damage or to perform unwanted actions on the infected computer system. Behavior-based analysis of malware typically utilizes tools that produce lengthy traces of observed events, which have to be analyzed manually or by means of individual scripts. Due to the growing amount of data extracted from malware samples, analysts are in need of an interactive tool that supports them in their exploration efforts. In this respect, the use of visual analytics methods and stored expert knowledge helps the user to speed up the exploration process and, furthermore, to improve the quality of the outcome. In this paper, the previously developed KAMAS prototype is extended with additional features such as the integration of a bi-gram based valuation approach to cover further malware analysts' needs. The result is a new prototype which was evaluated by two domain experts in a detailed user study.**

I. Introduction

Malicious software, or short malware, is one of the biggest threats to computer systems these days [1]. 'Malware' refers to software programs, which are designed to cause damage or perform other unwanted actions on a computer or network. Therefore malware plays a big part in most computer intrusions and security incidents. Malware includes inter alia: viruses, trojan horses, worms, rootkits, scareware, and spyware [1]. By now there are millions of malicious programs and the number is increasing every day.

"Malware analysis is the art of dissecting malware to understand how it works, how to identify it, and how to defeat or eliminate it" [1]. In malware analysis, there are two basic approaches to examine a malware program: the static and the dynamic approach. Often the malware analyst only has the potentially malicious executable, which includes the machine code but is not human-readable. Therefore, static malware analysis involves the investigation of the malware executable as well as certain reverse-engineering tasks to recover the sample's source code. On the other hand, dynamic analysis requires the execution of the malicious software on e.g. a virtualized host machine to detect the malware's runtime behavior [1]. To cover all of the malware analyst's needs, Wagner et al. [2] performed a problem characterization and abstraction elaborating the analysts needs in relation to behavior-based malware analysis. In the article by Wagner et al. [3] a design study for a behavior-based knowledge-assisted malware analysis system (referred to as KAMAS) is described. The malware analyst's workflow involves the tasks of examining potentially malicious behavior patterns, selecting them, categorizing them, and storing the found rules in the knowledge database (KDB) [3]. We developed an interactive prototype to extend the KAMAS design study [3] with a new feature of **Bi**-**G**ram supported **G**eneric **K**nowledge-**A**ssisted **M**alware **A**nalysis **S**ystem (BiG2-KAMAS) [4]. A focus group meeting with members of an Austrian IT security company, the Information security department of St. Pölten UAS and the developers of the initial KAMAS prototype was conducted to identify the tasks and needs for additional features requested by the IT security company to extend the KAMAS design study [3]. Based on this feature list, the paper at hand contributes the following:

1) Integrating a generic data loading process enabling KAMAS to load any kind of data, based on a given structure;
2) Storing benign rules and their highlighting when loading new cluster files, thereby supporting the analyst;
3) Identifying malicious or benign call sequences by including a bi-gram based valuation;
4) Presenting in detail two user studies validating the new features.

This paper is structured as follows: Sect. II provides background knowledge about the work of our collaborators and related work in the field of malware analysis. In Sect. III we describe the prototype's design, visualization methods and implementation. Furthermore, Sect. IV defines the integration of additional knowledge in the prototype's knowledge database. Sect. V shows the prototype's evaluation method, while results are discussed in Sect. VI.

II. Related Work

Shiravi et al. [5] published a survey related to network security visualization, comparing the data sources and visualization techniques of thirty-eight different systems. Furthermore, Egele et al. [6] presented a general literature for malware analysis techniques and tools. In their work they surveyed different approaches for dynamic automated malware analysis and compared them based on their analysis techniques.

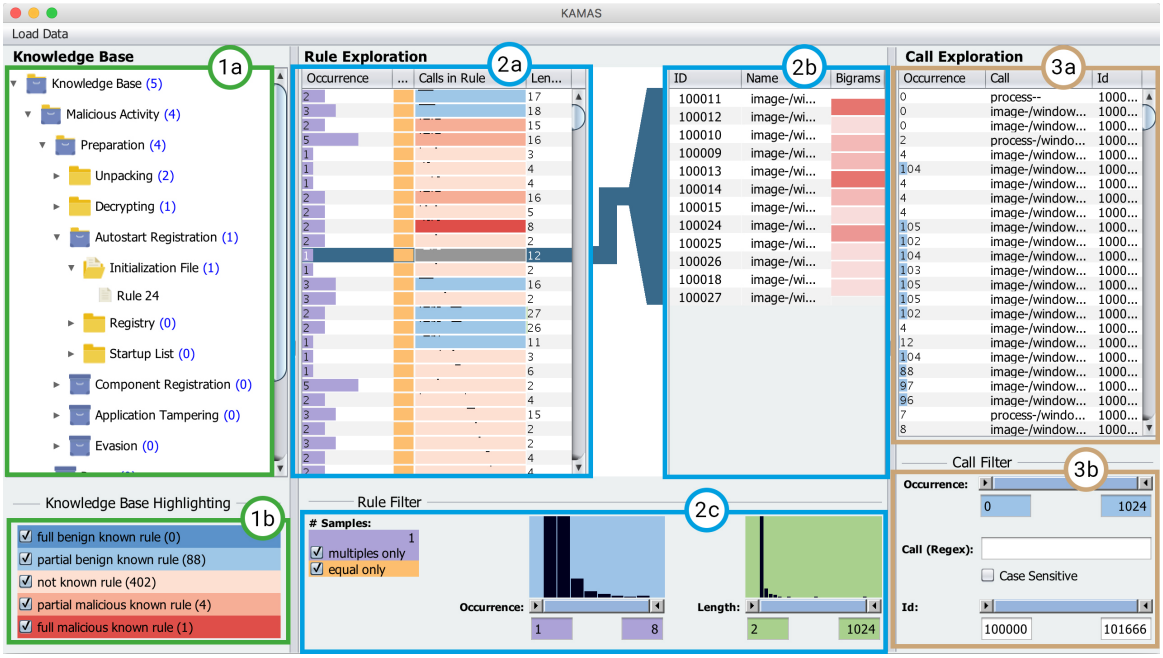

Fig. 1. The BiG2-KAMAS prototype and it's three sections: Section 1 shows the knowledge base including the KDB (1a) with it's new category for benign activity. Beneath the knowledge base highlighting filters are displayed (1b). Section 2 shows the rule exploration area including the bigram visualization (2b) and new color highlighting for benign rules (2a). Finally, section 3 shows the call exploration area.

Likewise, Bazrafshan et al. [7] surveyed various heuristic malware detection techniques as well as malware obfuscation techniques. Additionally, Wagner et al. [8] published a survey of 25 different visualization systems for malware analysis. The objective of their work was the comparison and categorization of the malware systems visualization methods and features and categorizing them along their novel 'Malware Visualization Taxonomy'. Furthermore, McNabb and Laramee [9] published a survey of surveys: Mapping The Landscape of Survey Papers in Information Visualization.

In 2017, Wagner et al. [3] published a paper on a Knowledge-Assisted Malware Analysis System, referred to as KAMAS. In their user study, they found out that the experts are not only interested in visualizing patterns. A supportive valuation approach was implemented by Luh et al. [10], [11], calculating the degree of maliciousness based on system and API call bi-grams. Somarriba et al. [12] presented another malware detector system for Android Malware Behavior. Besides, Marschalek et al. [13] published a system for threat detection using a real-time monitoring agent to gather all or only selected system events and visualize these using event propagation trees. Xiaofang et al. [14] published a paper of a malware variant detection approach using Similarity Search "by processing malware as content fingerprint" [14]. Jain et al. [15] presented a visual exploration approach of android binary files. Their approach is based on the visualization of android .dex files to analyze and compare malicious android executables. David et al. [16] presented "a novel deep learning based method for automatic malware signature generation and classification" [16]. Wrench and Irwin [17] published an approach in which they identify and classify Remote Access Trojans (RATs) and other malicious software based on the programming language PHP.

III. Prototype Concept

This section describes the new features of the '**Bi**-**G**ram supported **G**eneric **K**nowledge-**A**ssisted **M**alware **A**nalysis **S**ystem (BiG2-KAMAS), conceptually grounded on the KAMAS prototype [3].

A. Data

In its current iteration, BiG2-KAMAS bases its visualization on sequential traces of Windows kernel operations amounting to benign and malicious application behavior in the context of OS and user-initiated processes. These *events* are typically abstractions of raw system and API calls that yield information about the general behavior of an unknown application sample or resident process [8]. Raw calls may include wrapper functions (e.g. `CreateFile`) that offer a simple interface to the application programmer, or native system calls (e.g. `NtCreateFile`) that represent the underlying OS or kernel support functions. In the context of BiG-KAMAS and its data providers, events are collected directly from the Windows kernel. We employ a driver-based monitoring agent [13] designed to collect and forward a number of events to a database server. This gives us unimpeded access to events depicting operations related to process and thread control, image loads, file management, registry modification, network socket interaction, and more. For example, a shell event that

creates a new binary file on a system may be simply denoted as a triple `explorer.exe,file-create,sample.exe`. Additional information captured in the background includes various process and thread ID information required to uniquely identify an event within a system session and to link individual events to a full sequence (trace) needed for further processing stages. Based on aforementioned traces, BiG2-KAMAS uses two distinct mechanisms to further process arbitrary kernel event sequences:

Pattern inference: Our introduced framework has been developed in concert with an event extraction system called SEQUIN [11]. SEQUIN uses grammar inference extended with statistical evaluation to automatically identify and crop relevant sequences (rules) from traces of kernel-level behavioral data for further processing and visualization. Generally speaking, grammar inference is the process of computationally assembling a formal ruleset by examining the sentences of an unknown language [18]. In the information security domain, grammar inference is primarily used for pattern recognition, computational biology, natural language processing, language design programming, data mining, and machine learning. Grammar inference has also been proven to be a feasible approach to anomaly detection, since "algorithmic incompressibility is a necessary and sufficient condition for randomness" [19]. We use grammar inference as key component in the process of 'compressing' a sequential trace for extracting relevant behavioral patterns.

To achieve inference by compression in a computationally feasible way, we selected an algorithm that losslessly produces (without changes to order and immutability) a context-free grammar (CFG) in unsupervised operation. As opposed to context-sensitive grammars, languages created by a CFG can be recognized in $O(n^3)$ time, which is a relevant distinction for all future parsing efforts. The choice ultimately fell on Sequitur [20]. Sequitur is a greedy compression algorithm that creates a hierarchical structure (CFG) from a sequence of discrete symbols by recursively replacing repeated phrases with a grammatical rule. The output is a compressed representation of the original sequence. The algorithm creates this representation through the application of two base properties: *rule utility* and *bi-gram uniqueness*. Rule utility checks if a rule occurs at least twice in the grammar, while bi-gram uniqueness observes if two adjacent symbols occur only once. Assuming we have a string `abcdbcabcd`, where every character represents an event, the first bi-gram of that trace would be `ab`, followed by a second bi-gram `bc`, and so forth. See Table I for a complete example of the process.

Sequitur is linear in space and time. In terms of data compression, the algorithm can outperform other designs that achieve data reduction by factoring out repetition. It is almost as performant as designs that compress data based on probabilistic predictions [20].

Bi-gram extraction and scoring: In addition to rule inference, BiG2-KAMAS uses precomputed maliciousness scores of event bi-grams separately explored using a sentiment-like extraction system based on the log likelihood ratio (LLR) test

TABLE I
OPERATION OF SEQUITUR AFTER [20]. PROPERTY APPLICATION IS *italicized*.

Symbol	String	Grammar	Remarks
1	a	S → a	
2	ab	S → ab	
3	abc	S → abc	
4	abcd	S → abcd	
5	abcdb	S → abcdb	
6	abcdbc	S → abcdbc	bc appears 2x
		S → aAdA	*bigram uniqness*
		A → bc	
7	abcdbca	S → aAdAa	
		A → bc	
8	abcdbcab	S → aAdAab	
		A → bc	
9	abcdbcabc	S → aAdAabc	bc reappears
		A → bc	
		S → aAdAaA	*bigram uniqness*
		A → bc	aA appears 2x
		S → BdAB	*bigram uniqness*
		A → bc	
		B → aA	
10	abcdbcabcd	S → BdABd	Bd appears 2x
		A → bc	
		B → aA	
		S → CAC	*bigram uniqness*
		A → bc	B used only 1x
		B → aA	
		C → Bd	
		S → CAC	*rule utility*
		A → bc	
		C → aAd	

[10]. An LLR test is a statistical method used test model assumptions, namely the quality of fit of a reference (null) and an alternative model. When determining the occurrence of rarely observed events – which are often at the core of malicious traces – likelihood ratio tests show significantly better results than alternatives such as x^2 or z-score tests [21].

In preparation for sentiment-assisted visualization, we use the LLR method to learn likely benign and malicious event sequences in big corpora of recorded kernel operations (traces). The resulting sentiment dictionary can be used to accurately and effectively determine if an investigated event bi-gram is contextually suspicious. Specifically, we compute the LLR score for each bi-gram to highlight collocations characteristic to sequences of malicious and benign system events [10].

The resulting occurrence counts (shown in Table II) are the basis for this calculation: Following the approach by [21], we define the number of times both event tokens occur in combination (k_{11}), the number of times each token has been observed independently from the other (k_{12} and k_{21}, depending on the relative position in the bi-gram), and the number of times the token was not present at all (k_{22}).

TABLE II
EVENT OCCURRENCE MATRIX [10]

	A	!A
B	k_{11}=$k(AB)$	k_{12}=$k(!AB)$
!B	k_{21}=$k(A!B)$	k_{22}=$k(!A!B)$

The same process is later applied to the pattern's general occurrence in a labeled benign versus malicious corpus. The final result is a normalized sentiment rating ranging from $+1.0$ (benign) to -1.0 (malicious). Unknown bi-grams are ultimately scored against the resulting dictionary, the outcome of which is at the core of the bi-gram evaluation feature in the new BiG2-KAMAS prototype.

B. Visualization Design

Structure: Wagner et al. [3] describe in their article that since IT-security experts are commonly familiar with programming IDEs, they used the design concept of IDEs like Eclipse or Netbeans for their prototype. The updates to the new prototype also follow this design concept approach. In contrast to the previous prototype, the new one has an additional view. In this initial view the KDB is situated on the left side, which can be compared to the project view in Eclipse. On the right side only the file load buttons are displayed, which can be compared to the initial view of Eclipse, where no project has been opened yet.

Coloring: For the rule highlighting as well as the Bi-Gram visualization we selected a sequential color scheme from red to blue. Red indicates that the rule or bi-gram is malicious and a blue one stands for a benign rule or bi-gram. To avoid problems with red and green hues for colorblind people [22, p. 124], we used blue instead of green and select colorblind-safe qualitative colors from Colorbrewer[1].

Layout: The prototype is structured into three parts: knowledge base, rule exploration area and call exploration area (see Figure 1). On the left side the knowledge base is visualized with it's 'Knowledge Database (KDB)' (see Figure 1:1a) and the KDB's color highlighting filters (see Figure 1:1b). The KDB is displayed as a tree, in which each category of the database can have several subcategories. Each category with subcategories is shown with a box icon (see Figure 1:1a) and the ones without subcategories are displayed with folder icons. Each rule, which is stored in the database, is displayed with a paper icon. Beneath the KDB the 'Knowledge Base Highlighting' filters are displayed (see Figure 1:1b). Each filter can be activated or disabled with its checkbox and updates the result of the prototypes filter pipeline and visualization of the 'Rule Overview Table' (see Figure 1:2a).

After loading and translating the input file, the system updates the 'Graphical User Interface' (GUI) and visualizes new elements. In the middle the 'Rule Exploration' area (see Figure 1:2) is visualized, while the right side contains the 'Call Exploration' area (see Figure 1:3).

In the 'Call Exploration' area all the included system or API calls of the loaded input file are represented in the call table (see Figure 1:2b) as described by Wagner et al. [3]. The rules included in the input file are visualized in the rule overview table located in the 'Rule Exploration' area (see Figure 1:2a). If the user loads several trace files, each trace file will be displayed as one rule.

[1] http://colorbrewer2.org

The background of the third column of the 'Rule Overview Table' indicates whether a rule is fully benign, partially benign, not known, partially malicious or fully malicious. The background of the malicious rules will be painted in red and the background of the benign rules in blue. The fully known rules will be displayed in a dark red/blue while the partially known rules are highlighted in a light red/blue (see Figure 1:1b). The red color highlighting for malicious activity is adopted of the KAMAS prototype [3]. If a rule is fully known and, therefore, highlighted in dark red, the rule is included as-is in the KDB. A partially known rule is only a part of one rule in the KDB. This kind of rule has at least one additional call at the beginning or at the end of a fully known rule [3]. If an input file was loaded, the system automatically calculates the knowledge state of each rule. For this purpose, the system compares each rule of the input file with each rule of the KDB. After the calculation process the system highlights the rules in the corresponding colors in the rule overview table.

Bi-Gram Visualization: The rule detail table is located next to the rule overview table (see Figure 1:2b). The rule detail table automatically updates its content when clicking on a rule in the rule overview table and represents all system and API calls included in the selected rule. From left to right, the table displays the unique id as well as the name of the call. The last column visualizes the new bi-gram based valuation approach for the corresponding calls. As mentioned before, the prototype uses the bi-gram approach of Luh et al. [10]. A bi-gram is an n-gram where the length of $n = 2$. An n-gram, in turn, is a coherent sequence of n elements. In this approach the elements are system or API calls. Each bi-gram has a score in the range [-1, 1], which indicates whether this pair of calls is malicious or benign. For bi-gram based valuation, two different visualization approaches were implemented following a semantic zooming approach: First, if the width of the bi-gram column is bigger than 75px, the prototype visualizes the bi-gram values as bar charts (see Figure 2:a), whereby each bar starts in the middle of the bi-gram column. If the bi-gram score is between 0 and -1, the bi-gram is malicious. Therefore, the red color bar chart unfurls from the middle towards the left side. If the bi-gram score is between 0 and 1 the bi-gram is benign and the bar chart is visualized from the middle to the right side in a blue color. The colors correspond to the KDB highlighting. The visualization approach was chosen to give the user a quick but still precise overview of the bi-gram based scores.

If the width of the bi-gram column is smaller than 75px and therefore the bar charts are hardly recognizable, the system switches to the second visualization. Here, the bi-gram values are visualized as a color-filled rectangle (see Figure 2:b). As before, a red colored rectangle indicates that the bi-gram is malicious and a blue one stands for a benign bi-gram. To visualize the value of the malicious or benign bi-gram, the system changes the alpha value of the displayed color. Therefore, the darker the color, the higher the respective value. Since the difference of an alpha value between 255 and 240 is

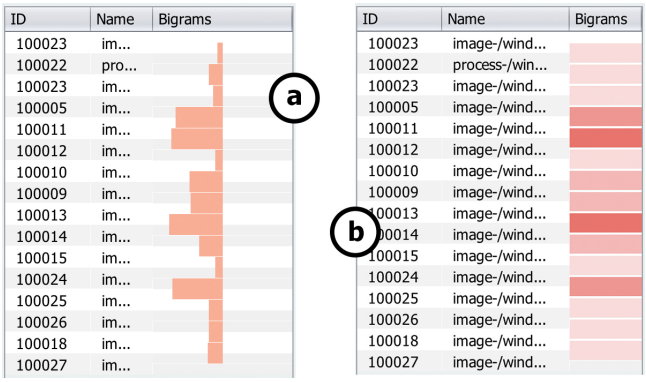

Fig. 2. The two different visualisations methods of the call bi-grams. The first method visualises the bi-grams as bar charts (a), whereas the second visualisation uses the alpha channel to show the severity of the bi-gram (b).

not easy to recognize and every value below 100 is generally difficult to see, we decided to implement only four graduation steps for the alpha value. The visualization with the alpha value is less precise than the visualization with the bar charts but, at the same time, significantly easier to interpret. Table III shows the different graduation steps and their value ranges.

TABLE III
COLOUR GRADUATION STEPS FOR THE ALPHA VALUE BI-GRAM VISUALISATION.

Colour	Alpha value	Value ranges
	200	>= 0.75
	150	>= 0.5 && <0.75
	100	>= 0.25 && <0.5
	50	>= 0 && <0.25
	50	<0 && <= -0.25
	100	<-0.25 && <= -0.5
	150	<-0.5 && <= -0.75
	200	<-0.75

C. Interaction

Like the KAMAS prototype of Wagner et al. [3], the BiG2-KAMAS's functionality will be described in accordance to the four steps of the visual information seeking mantra of Shneiderman et al. [23], namely overview, rearrange and filter, details-on-demand and, extract.

Overview: The BiG2-Kamas prototype has an additional initial view where the user can decide whether to load a Sequitur input file or several raw trace files. When the analyst loads a Sequitur file, the rule and call tables will be filled with the rule and call data included in the input file. Each entry in the rule overview table represents one rule of the loaded cluster. Furthermore, the histograms in the rule exploration area give a quick impression of the distribution in the rule occurrence and length [3]. When the user loads one or more trace files the rule and call tables will also be filled with the data of these files. Contrary to a loaded Sequitur file, each entry of the rule overview table represents an entire trace file. Thus, if the user loads three traces the rule overview table will have only three rows. Furthermore, due to the fact that the user analyses several independent trace files the histogram for the rule occurrence is insignificant. Therefore, only one histogram for the trace length will be displayed in the rule filter area.

Rearrange: If the rule overview table and the call overview table are loaded with data, the user can rearrange their content by clicking on a table's column. This will re-sort the included data and update the visualization [3]. The content of the rule detail table cannot be rearranged since the calls are shown in their sequential order and should therefore not be changeable.

Filter: In the next step the user can reduce the number of rules or trace files by using the rule/trace and call filters [3]. No matter which files were loaded, the user always has the opportunity to filter the rules or traces by the included calls (events). The user can rearrange the call filters or select a specific call in the call overview table to reduce the number of shown rules [3]. Furthermore, the analyst can filter the rules or specific traces by using the filters in the rule exploration area. If loading a Sequitur file, the analyst can filter the rules by their occurrence, length, whether they are equally distributed in the input file or if they match, partially match, or don't match the stored rules in the KDB [3]. By changing the filter settings, the included rules in the rule overview table automatically update immediately. If one or more trace files were loaded, the analyst can only filter the shown traces in the rule overview table by their length. In addition, the highlighting and filtering of the KDB is switched off.

Details-on-Demand: If the user wants to analyze a rule or trace, he/she can open the rule/trace in the rule detail table by selecting it in the rule overview table. This will display all the included calls in the rule detail table in their sequential order [3]. The bi-grams provide information whether a combination of two calls is malicious or benign. This should support the user in finding interesting call sequences more quickly.

Extract: Independent of the loaded files the analyst can add a new rule to the database using two different ways. One method is to simply select one rule or trace in the rule overview table and simply drag and drop it in one leaf category of the KDB. This will add the entire rule or trace file to the database [3]. Alternatively, the analyst can select several calls of interest in the call overview table and add these by dragging and dropping them to the KDB. When adding a new rule to the KDB, a popup window will show up where the analyst can assign the rule a specific name. If the user has loaded a Sequitur file, the system will now update the knowledge state for all rules as well as the highlighting in the rule overview table for further analysis.

D. Implementation

Since the BiG2-KAMAS prototype is based on the prototype of Wagner et al. [3], it also uses a data-oriented design concept [24]. To increase the performance of the prototype, the system only works with integer comparisons. Therefore,

the input data only includes the call ids. It is only possible to translate a call id to the actual call value with an additional translation file. This translation file is also used for the bi-grams. The original bi-gram file has several columns in which only the string values of the system or API calls are stored. To increase the performance and to reduce memory usage, the BiG2-KAMAS prototype generates its own bi-gram file. When starting the prototype the system checks with md5 hash values to determine whether the translation file or the original bi-gram file has changed. If so, the system converts the original bi-gram file to the translated bi-gram file in which also the integer values of the system calls are stored. Like the prototype of Wagner et al. [3] the new prototype is using the action pipeline for filter options. This enables dynamic query environments and real-time data operations.

To evaluate the robustness and performance of the BiG2-KAMAS prototype three different Sequitur cluster-grammar files containing between 10 and 500 rules were used. The file with 500 different rules contained a total amount of 30,000 system and API calls. To test the bi-gram functionality, a bi-gram file with nearly 117,500 bi-gram entries was loaded. On a machine with an 2.1GHZ Dual-Core processor and 12GB of memory it took the system about four minutes to translate the original bi-gram file to the translated bi-gram file. The malware and bi-gram samples were collected by collaborators in the Josef Ressel Center TARGET of St. Pölten UAS.

IV. Externalized Knowledge Integration

As Wagner et al., [3] described in their article, we integrated a knowledge database to support the user during their analysis tasks. The KDB is based on the malware behavior schema of Dornhackl et al., [25]. The KDB is located at the left side of the prototype and is implemented in a hierarchical structure (tree structure). In the BiG2-KAMAS prototype the KDB was extended by one additional category to store the benign rule data, namely benign activity. In the current version of the prototype there is only one category to store benign rule data. Each category is displayed with either a box or a folder icon, the category description and the number of included rules in the integrated subfolders. The analyst can add new rules by drag & drop. When adding a new rule, the KDB automatically unfolds closed categories. Additionally, a popup window opens in which the analyst can enter a rule name. To investigate a rule stored in the KDB, the user can open a context menu by right clicking on the chosen rule. The context menu will show two different menu items, namely 'Information' and 'Delete'. The information menu item opens a popup window in which the analyst is presented the following information:

- **Assigned Concept:** This information tells the analyst in which schema category (concept) the rule is currently categorized. The assigned concept is implemented as a selection list to give the user the opportunity to change the assigned concept. For that purpose, the analyst must select a different concept in the list and press the save button at the bottom of the pop up window.
- **Rule Name:** Here, the actual rule name is displayed. The rule name is implemented as a text field to quickly change it if necessary.
- **Included Calls:** Finally, the calls included in the stored rule are displayed in a table. Thus, the calls are visualized in their sequential order and each call will be shown with its unique call id which corresponds to the call id of the translation file and the actual call value. In the current version of the prototype it is only possible to investigate the included calls in their sequential order, but not to delete specific calls which are listed in the table.

The second menu item is the "delete' item, which allows the analyst to delete the currently selected rule. Furthermore, when selecting a concept instead of a rule, the BiG2-Kamas prototype will show a context menu with which the user can disable a category and all its integrated subcategories. Thus, the analyst can disable the entire KDB or only specific categories. If the user disables a category all the included rules will no longer be considered in the knowledge base highlighting and filtering.

When the user clicks the right mouse button to open the corresponding context menu before selecting a rule or category, the system automatically selects the rule/category at the actual mouse position.

Searching: If the user searches for interesting rules or specific calls or call groups he/she can use the call filter options to reduce the data to be analyzed. In the call exploration area, the user can search for a specific call by entering its name or use regular expressions to find an entire call group. Beneath the search text field the user can enable case sensitive search with the corresponding checkbox 'Case Sensitive'. Filtering or searching the calls affects the data shown in the call overview and rule overview table. Additionally, to find rules of interest the analyst can use the rule exploration filters or the knowledge base filters.

V. Prototype Evaluation

This section describes the procedure of the performed user studies, the specific results, as well as further feature requests. For the prototype validation, a user study with two domain experts was conducted. The domain experts validated the functionality as well as the visual design interface.

Participants: Both participants work at St. Pölten UAS and have more than five years of experience in the field of malware analysis. The first participant is between 30 and 39 years of age, male and holds a masters degree. The second participant is between 60 and 69 years of age, male, and holds a PhD. Generally, both participants are well experienced in this field and can be categorized as experts.

Design and Procedure: Each participant was interviewed individually and had already tested the previous version of the prototype at least once. First, the participants received a short introduction to the new features of BiG2-KAMAS and also a quick reminder of the basic features and workflow. The participants were asked to mention additional missing functionalities and to criticize all potential usability issues.

Both participants took part in the same two scenarios: First, the participants had to load a Sequitur file, investigate the loaded rules and filter specific call sequences. At the end they had to store a rule in the KDB and name it. In the second scenario, the participants had to load three trace files. They were asked if they perceived any differences when loading trace files instead of a Sequitur file. At the end they had to investigate a rule stored in the KDB and move it to a different category.

Equipment and Materials: The latest version of the BiG2-KAMAS prototype was used in the evaluation. For the first user scenario, the participants had to load a Sequitur file with about 500 rules and 30,000 system and API calls. In the second scenario, three trace files with a length between ten and fifteen calls were used. The bi-gram file had a total number of about 117,000 bi-grams. The translated bi-gram file had already been generated so that the participants did not have to wait until the system finished the translation process. As evaluation equipment, two different setups were used. Both participants worked on a 13 inch Macbook Pro with a Retina display (screen resolution of 2560x1600) and a mouse for navigation. Participant #1 worked with an additional 20 inch Monitor with a full HD screen resolution and an external keyboard. Each user test was conducted with the same version of the BiG2-KAMAS prototype and was documented on paper.

A. Results

The following section discusses the results of both scenarios. Both the results of 'Scenario 1' (Sequitur file) and 'Scenario 2' (trace files) will be presented. Both participants had no problem loading the different files for the user scenarios.

Scenario 1: *Loading and Analyzing a Sequitur file.*
Both participants quickly recognized the additional color scheme for the new benign category. The colors for the knowledge base highlighting were assessed as easily understandable and the additional rule counter next to the knowledge base filters were mentioned as being very useful. Participant 1 mentioned that if a rule in the rule overview table is highlighted, it would be useful to know which rule or rules of the KDB match this rule in the table. Therefore, a tooltip would be helpful which tells the user the names of the matching rules of the KDB. Furthermore, participant 2 suggested to always show the rule counter of the KDB's categories. If there are currently no rules in a category, the counter should be zero.

When participant 2 first saw the bar chart bi-gram visualization, he assumed it visualizes the occurrence of the combined call sequence. In contrast, the alpha color visualization was immediately recognized as an indicator for maliciousness or benignity. Participant 1 also mentioned that the alpha color visualization is easier and faster to recognize. Furthermore, both participants mentioned that the color visualization is not as precise as the bar chart visualization and therefore would only be useful for initial malware classification. Participant 1 suggested an additional tooltip to display the accurate bi-gram value. Participant 2 remarked that it would be more useful if the calls in the call overview table only showed the beginning and the end of the call's value. This would simplify finding a specific call in a group of similar calls. Additionally, he recommended a search button for the regular expression call filter. This could help some users, since currently it is only possible to search by pressing the enter key. Adding a new rule to the KDB was no challenge for either participant and both valued the ability to give the rule a specific name.

Scenario 2: *Loading and analyzing three trace files.*
Both participants had no difficulties with loading the three trace files. They also recognized quickly that each entry in the rule overview table now represents one trace. Neither of them realized that the knowledge base filters and highlighting were disabled. Participant 1 suggested to gray out the knowledge base filters to make it clear that these are disabled. Participant 2 proposed to change the headings for the trace file analysis view in order to avoid confusion. He remarked that it could be misleading if the headings say e.g. 'Rule Overview Table' when analyzing a trace file. Furthermore, both participants recommended to change the occurrence column in the rule overview table to the file names of the traces. As the last task, the participants had to change the corresponding category of a random rule. Even if both participants solved this task easily, both remarked that it would be useful if the user could move a rule from one category to another per drag & drop.

B. Result Analysis

This section gives an overview of the issues which were mentioned during the expert reviews. Like Wagner et al. [3] each issue was rated based on Nielsen's [26] severity ratings. Table IV shows the potential new features noted by the test persons and includes three columns: 'feature requests' (FR), 'severities' (SE) and the effort it would take to implement these changes [3]. The features mentioned in the table include small cosmetic changes as well as real usability improvements. The only feature mentioned by all participants is an additional tooltip which shows the actual bi-gram values.

TABLE IV
LIST OF REMARKED FEATURE REQUEST AND SEVERITIES AND THE EFFORT IT WOULD TAKE TO IMPLEMENT THEM IN THE PROTOTYPE. (FR: 1 = NICE TO HAVE, 2 = GOOD FEATURE, 3 = ENHANCES USABILITY; SE: 1 = MINOR, 2 = BIG, 3 = DISASTER; EFFORT: 1 = MIN, 2 = AVERAGE, 3 = MAX) [3].

Description	FR	SE	Effort
KDB: Move a rule to another category by using drag & drop.	2	1	1
KDB: Show the rule counter even if zero rules are included.	1	-	1
KDB: Gray out the knowledge base filters if they are disabled.	2	1	1
Tables: Highlighted rules in the rule overview table should show the KDB's corresponding rules.	3	2	3
Tables: Change the occurrence column to the trace file names.	2	1	2
Tables: Show only the begin and the end of the calls in the call overview table.	3	2	2
Tables: Implement a search button for the call regex search.	1	-	1
Bigram: Tooltip to show the bi-gram values.	3	-	1
Headings: Change the headings when loading trace files.	2	-	1

VI. Discussion & Reflection

The performed user studies described in Section V confirmed that the four feature requests, which are determined in Section I are fulfilled by the BiG2-KAMAS prototype:

1) *Generic data loading:* The BiG2-KAMAS prototype is structured to enable the generic loading of data sequences. To make this possible the input data as well as the prototype's database are based on unique identifiers (id) instead of the actual values. Thus, all system-internal comparisons are based on integer values instead of string values. Only with the corresponding translation table, the system can translate the ids to the actual values. Thus, it is possible to load data sequences independent of their actual values as long as there is a translation table through which the prototype can translate the data. Furthermore, the system was adopted to also offer the opportunity to load raw system or API call based traces. In this state the KDB highlighting and filtering is disabled but the user can explore the loaded trace files and add new rules to the KDB. The prototype can't only load Sequitur call sequences, but also independent data sequences as long as the the data sequence has the given structure and a translation file.

2) *Extend the KDB with benign rules:* To fulfill this requirement the KDB was extended with an additional category for benign activity. In this category, all rules which are identified as benign can be stored. Additionally, the KDB's highlighting and filter pipelines were extended to identify and filter partially and fully benign rules. Rules with a partially or fully benign knowledge state are highlighted in blue in order to avoid the combination of the colors red and green.

3) *Implementation of bi-gram based valuation:* To support the bi-gram approach of Luh et al, [10] the prototype's rule detail table was adopted. Since many domain experts mentioned [3] that the arc-diagram visualization is not very helpful, it was replaced by the bi-gram visualization. Bi-gram based valuation is implemented with two different approaches. If the width of the bi-gram column is bigger than 75px the valuation is visualized with bar charts and colored in red (malicious) or blue (benign). If the width is less than 75px the bi-gram visualization uses the alpha channel to show the severity of the bi-gram (see Table III).

4) *User studies to validate the new features:* The results of the user studies show further feature requests which could be implemented in a future project. However, both participants mentioned that the bi-gram visualization is very helpful for identifying potentially malicious or benign call sequences and, therefore, helps to decide whether a rule is malicious or not.

Future Work: For the behavior-based malware analysis process, it could be valuable to implement a rule creation process where the analyst can build their own rules based on the known system and API calls [27]. Furthermore, it could be beneficial to edit the stored rules in the KDB or to build new rules based on existing patterns. Further avenues for future work are to include possibilities to hide, shrink an expand areas to provide the user with more flexibility. Moreover, to update the occurrence column of the Call Exploration area (see 1:3a) to show the relation to the total number of occurrences included in the loaded file. Additionally, normalizing the occurrence dataset and visualization to this total could be beneficial.

Categorization of BiG2-KAMAS: Like the KAMAS prototype [3] the BiG2-KAMAS prototype can be categorized as a Malware Forensic as well as a Malware Classification tool in the Malware Visualization Taxonomy of Wagner et al. [8]. However, due to the bi-gram based valuation the BiG2-KAMAS prototype offers the malware analyst an additional assistance for the Individual Malware Analysis.

VII. Conclusion

In this work, we presented a design study for a Bi-gram Supported Generic Knowledge-Assisted Malware Analysis System (BiG2-KAMAS). The prototype is based on the KAMAS prototype [3] and extended by additional features such as generic data loading, an extension of the KDB to enable the analysis of benign rules, and the implementation of a bi-gram based valuation approach. The requirements were discussed in a focus group meeting and then implemented as part of a functional prototype. After implementing the new features, two user studies were conducted to evaluate the design and the functionality of the new BiG2-KAMAS prototype.

Acknowledgments

The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development is gratefully acknowledged.

This work was supported by the Austrian Science Fund (FWF) via the "KAVA-Time" project (P25489-N23) and by the Austrian Federal Ministry of Science, Research and Economy under the FFG Innovationscheck (no. 856429). We would also like to thank all focus group members and test participants who have agreed to volunteer in this project.

References

[1] M. Sikorski and A. Honig, *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*, 1st ed. No Starch Press, 2012.

[2] 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*, ser. VizSec '14. ACM, 2014.

[3] 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.

[4] N. Thür, M. Wagner, J. Schick, C. Niederer, J. Eckel, R. Luh, and W. Aigner, "Big2-kamas: Supporting knowledge-assisted malware analysis with bi-gram based valuation," in *Poster of the 14th Workshop on Visualization for Cyber Security (VizSec)*, Phoenix, Arizona, USA, 2017.

[5] H. Shiravi, A. Shiravi, and A. Ghorbani, "A survey of visualization systems for network security," vol. 18, no. 8, pp. 1313–1329, 2012.

[6] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A survey on automated dynamic malware-analysis techniques and tools," vol. 44, no. 2, pp. 6:1–6:42, 2008.

[7] Z. Bazrafshan, H. Hashemi, S. Fard, and A. Hamzeh, "A survey on heuristic malware detection techniques," 2013, pp. 113–120.

[8] 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*. The Eurographics Association, 2015.

[9] L. McNabb and R. S. Laramee, "Survey of surveys sos - mapping the landscape of survey papers in information visualization," *Comput. Graph. Forum*, vol. 36, no. 3, pp. 589–617, Jun. 2017. [Online]. Available: https://doi.org/10.1111/cgf.13212
[10] R. Luh, S. Schrittwieser, and S. Marschalek, "LLR-based Sentiment Analysis for Kernel Event Sequences." IEEE, 2017.
[11] R. Luh, G. Schramm, M. Wagner, and S. Schrittwieser, "Sequitur-based Inference and Analysis Framework for Malicious System Behavior," 2017.
[12] O. Somarriba, U. Zurutuza, R. Uribeetxeberria, L. Delosières, and S. Nadjm-Tehrani, "Detection and visualization of android malware behavior," vol. 2016, p. e8034967, 2016.
[13] S. Marschalek, R. Luh, M. Kaiser, and S. Schrittwieser, "Classifying malicious system behavior using event propagation trees." ACM Press, 2015, pp. 1–10.
[14] B. Xiaofang, C. Li, H. Weihua, and W. Qu, "Malware variant detection using similarity search over content fingerprint." IEEE, 2014, pp. 5334–5339.
[15] 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. ACM, 2015, pp. 9:1–9:9.
[16] O. E. David and N. S. Netanyahu, "DeepSign: Deep learning for automatic malware signature generation and classification." IEEE, 2015, pp. 1–8.
[17] P. M. Wrench and B. V. W. Irwin, "Towards a PHP webshell taxonomy using deobfuscation-assisted similarity analysis." IEEE, 2015, pp. 1–8.
[18] A. Stevenson and J. R. Cordy, "A survey of grammatical inference in software engineering," *Science of Computer Programming*, vol. 96, pp. 444–459, 2014.
[19] L. Ming and P. Vitányi, *An introduction to Kolmogorov complexity and its applications*. Springer Heidelberg, 1997.
[20] C. G. Nevill-Manning and I. H. Witten, "Identifying hierarchical structure in sequences: A linear-time algorithm," *J. Artif. Intell. Res. (JAIR)*, vol. 7, pp. 67–82, 1997.
[21] T. Dunning, "Accurate methods for the statistics of surprise and coincidence," *Computational linguistics*, pp. 61–74, 1993.
[22] C. Ware, *Information Visualization: Perception for Design*. Elsevier, 2012.
[23] B. Shneiderman, "The eyes have it: a task by data type taxonomy for information visualizations," in *Proc. of VL*, 1996, pp. 336–343.
[24] R. Fabian, "Data-Oriented Design," 2013, accessed on Nov. 11, 2015. [Online]. Available: http://www.dataorienteddesign.com/dodmain/dodmain.html
[25] H. Dornhackl, K. Kadletz, R. Luh, and P. Tavolato, "Malicious behavior patterns," in *SOSE*. IEEE, 2014, pp. 384–389.
[26] J. Nielsen, *Usability engineering*. Boston: Academic Press, 1993.
[27] 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.