MIMIR: Modelling user Intentions with Markov chains for Intention Recommendations

Romain Brisse, Simon Boche, Frédéric Majorczyk, Jean-François Lalande

Chapter 1

MIMIR: MODELLING USER INTENTIONS WITH MARKOV CHAINS FOR INTENTION RECOMMENDATIONS

Romain Brisse*†, Simon Boche†, Frederic Majorczyk*‡ and Jean-Francois Lalande*

* CentraleSupélec, Inria, Univ. Rennes, CNRS, IRISA, 35000, Rennes, France

† Malizen, 35000, Rennes, France

‡ DGA, 35000, Rennes, France

Abstract Despite detection tools and the automation of cybersecurity, analysts are more in-demand than ever. They have to perform complex security investigations in order to find and qualify threats. It is necessary to speed up and ease security tasks in order to reduce the effects of analysts shortages. Recommender systems are widely used in the task of helping users find their way in enormous amount of heterogeneous data for example in online marketplaces. That situation is similar to the one face by analysts. We thus offer to design a recommender system for incident response. By recognizing 7 relevant user intentions throughout the investigation process, we propose MIMIR, that provides relevant recommendations for the analyst's next actions based on their most probable objectives. We evaluate MIMIR in different ways, using 4 experiments and 5 datasets. The results show the validity of the model as well as the relevance of recommendations, which is a first step towards recommendations based on user intention recognition in the field of incident response.

Keywords: recommender systems, security investigations, forensics, log exploration

1. Introduction

The need for cybersecurity has been growing exponentially for the last few years. As attackers have been multiplying [13], the sheer volume of attacks registered has never been higher [12]. New APT groups are discovered on a regular basis and they each have their own methods, forcing automated tools to be retrained on newer and more representative data.

Those attacks are more likely to evade automated tools, causing an additional burden on already overloaded incident response teams. This work is dedicated to find a solution to lighten the workload of analysts during their investigations, especially junior analysts that need to be guided.

Recommender systems are a solution to gain speed and efficiency during specific tasks. They are already commonly used in fields like e-commerce to suggest relevant items to buy. Recommender systems are beginning to be used in the cybersecurity field, notably for incident handling and response [9, 6]. Their goals range from detection and mitigation recommendation, to analysing the security standards of a company in order to recommend protection plans.

Within incident response, investigations are conducted by analysts. During investigations, analysts rely on their knowledge, experience and instincts to detect suspicious behaviours, find the path followed by the attacker and understand the impact of the actions that the attacker carried out. We believe that it is possible to model the way analysts interact with data during investigations and, afterwards, to use this model to provide recommendations during future investigations. This recommender system would help reduce the total time needed for incident handling and limiting the impact of the incident.

In this paper we present MIMIR: a recommender system that recognizes analyst intentions during log exploration. MIMIR records and analyses actions taken by analysts and then offers exploration paths. The idea is to contribute a decision-helper tool that integrates well within an analyst's workflow and allows testing out exploration paths corresponding to one's intentions rapidly and efficiently. Indeed, analysts often know what to do but not always how to translate their instincts into concrete actions. MIMIR is based on the concept of intentions. Intentions are an abstraction of what analysts want to do with log data, such as *deepening his search*. We designed an experimental methodology that relies on the observation of investigations and the processing of their user actions, to extract a model of their intentions. This analyst intentions model can be reused to speed up other analysts during future investigations. From this model, we are able to build a Markov chain representing the next most probable intentions of an analyst. It allows us to recommend actions to execute in agreement with that intention.

We developed a prototype of MIMIR integrated in the visualization platform used for performing investigations. We evaluated this approach by analysing the relevance of the produced Markov chains with 80 investigations over 5 different log datasets and through 4 experiments.

This paper is organized as follows. Section 2 presents a use case where we show a relevant use for MIMIR. Related work regarding rec-

ommender systems as well as some work done in the investigative part of incident response are presented in Section 3. We present an overview of the complete recommender system in Section 4. Section 5 presents the model we have designed for the recommendation engine. Section 6 then presents how this model is designed and implemented. A description of the datasets used as well as an account of the evaluations we conducted are related in Section 7. In Section 8 we conclude and discuss future work.

2. Use case

In this section we describe a situation encountered during a threat hunting investigation where our recommender system would be relevant. The goal of our recommender system is to understand the intentions of an analyst during an investigation and offer him relevant actions that help to reach his goal.

In this scenario, an attacker has already infiltrated the network and exploited some client machines. An analyst has found these attacks and has found connection attempts from the attacker towards the machine hosting the Active Directory (AD) and some logs showing the AD being compromised. However, the analyst is interested in how the attacker worked his way into the Active Directory. He will try to find more information and context about the situation in order to formally identify the techniques used by the attacker.

The analyst searches for available data that he can correlate with the information he already has. He knows the attacker already has access to the machine, so he will check what processes are running at that point and who is running them with which privileges. In the interface, the analyst now observes the new visualizations and notices two strings from the logs: *powershell.exe* and *administrator*. Knowing this, the analyst is able to find out that the attacker gained access to the Active Directory using a Zerologon attack. The attacker used Powershell, and an administrator account without a password [8].

In this example, we demonstrate the intention of broadening one's research in order to gain information about a security situation. This will be a user intention we later call *broadening*. Many other user intentions can appear during an investigation and recognizing them well and associating them with corresponding user actions will be the core of this work. By anticipating the user intent, we can find out the next most probable intention the user might have and recommend some related actions to undertake. These next few actions we recommend represent the best way to achieve their goals for analysts. In doing so, we allow

analysts to focus on their expertise during investigations instead of the platform they are using to investigate.

Going back to the example, the analyst knows how the infected client machine was able to escalate its privileges and become an administrator on the AD. Now that he has all the context he needs, the analyst might want to find out the exact log that identifies this attack. The recommendation would offer him to filter the data on the value proving the use of the Zerologon attack in order to only see information related to it, and then filter again using the hostname of the machine he is studying instead of the user account like before, because this way he can uniquely identify a log line.

3. Related work

This work concerns multiple wide fields: recommender systems , incident response, and understanding user intentions. We will mainly focus on those topics.

3.1 Recommender systems

RSs are decision-helping tools. They are built as engines that take input data from various sources, extracting candidates for recommendation from this input, ranking them using a model of scoring and then presenting them to the user [10, 17]. The sources used as inputs define the type of recommender system being built (knowledge-based or collaborative filtering for example). They have been used in plenty of fields with success [1] and studied extensively to improve them [3] and hybridize them [4]. In this section, we will focus on their use in the cybersecurity field and specifically in incident response.

3.1.1 Recommender systems in cybersecurity. RSs applied to the cybersecurity field have been receiving some attention recently, surveys have been published allowing us to understand their role better [14, 9]. Most of them are related to attack prediction [15]. Based on various inputs, such as usage logs, topology or threats found in nature, an engine tries to predict which attacks are occurring or risk occurring in the near future and recommend those to the user. While the results are promising, the false positive rate in automatic detection can endanger the trust the user will put in the RS. Others derive from detection and try to offer protection plans [7]. In this case, the recommender system uses input data from a user's list of requirements and a list of protection services and their properties. It is particularly interesting to see that the hybrid sources of data are the one that allow for the extraction of

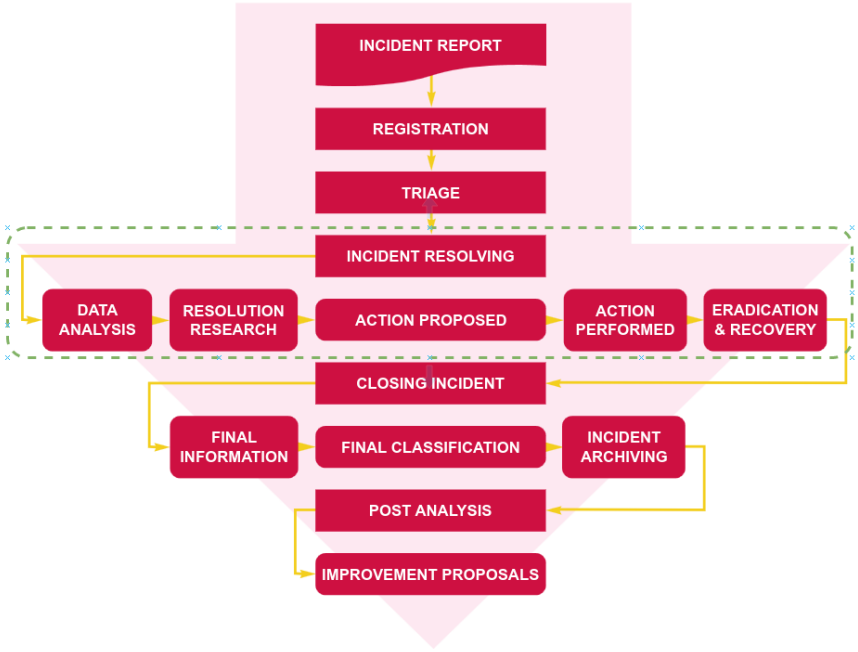

Figure 1. The workflow of incident response proposed by ENISA [12]

pertinent candidates and provide a valuable service to the user. It integrates the expertise in choosing protection services and offers it back to the user, thus facilitating his work.

3.1.2 RSs used for log analysis & incident response. RSs are also becoming widely used in incident response. Figure 1, shows the different steps of incident response. The main steps are in the center column. Most recommender systems work in this context focuses on data triage and incident resolving. For example, Del Esposte *et al.* [5] built a recommender system that uses alerts as well as network administrators ratings and preferences in order to dispatch alerts to the right analyst. In Security Operation Centers [21] the triage of alerts is time-consuming and so a tool like this one could be a good way to help analysts gain efficiency.

The most popular task to tackle in incident resolving is finding the best countermeasures and mitigations to a detected attack. APIRO [19] is a recommender system that uses API documentation and data augmentation techniques as inuts to a neural network. It then produces recommendations of the right API in order to understand incidents. Some

other research also tries to tackle this problem by directly recommending protection measures. For example, MENTOR is a tool developed by Franco *et al.* [7] that matches tools used for implementing some defence mechanisms according to user needs. Its goal is to provide an adequate level of protection.

Another step related to incident response is prevention and proactive protection. A lot of different works have addressed the problem. For example Soldo *et al.* contributed a novel method of predictive blacklisting [18]. The idea is to use blacklists of previously banned IP addresses and their behaviour and interactions with their victims in order to predict which IP addresses might be malicious in the future.

However, all these contributions are focusing on solving very specific tasks of the incident response chain. Often, automating some steps like triage is interesting because an error has little consequences: another analyst will receive the wrongly routed alert, find it to be mistriaged and redirect it to the right analyst.

In this paper, we focus on the highlighted part of Figure 1. Our work is set at the incident resolving step. It would be dangerous to try and automate incident response as a whole because an error could have a significant impact on response time. However, we work specifically with analysts that conduct investigations during incident response. Their work being done after the initial detection step by automated tools, we have the perfect opportunity to focus on helping users accomplish a task rather than replacing them.

3.2 Understanding analyst intentions

Moskal and Yang [11] developed a new method by using a machine learning model whose goal is to translate alert descriptions into a more interpretable state. They call it the *action, intent, stages* model. The idea is to combine expert knowledge databases such as MITRE ATT&CK, CVEs, well-known IDS signature bases etc. in order to refine the meaning of alerts which is often complicated to understand at first glance. Then, they recommend these improved descriptions to their users. While this type of recommender system helps in understanding attacker intentions and helps their users, they do not make use of the actions directly but rather work with established knowledge.

Zhong *et al.* [20] improves the performance of data triage and especially helps less experienced users in making the right decisions. They use the recorded actions of senior analysts during the analytic process of intrusion detection and compare the similarity of the recorded situation with the new contexts encountered by the junior analysts. By associ-

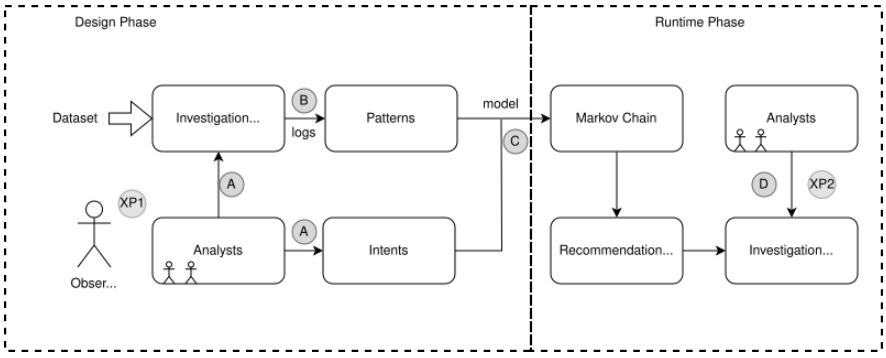

Figure 2. Overview of the design and runtime of the MIMIR RS

ating the resolved incidents with ongoing ones, they are able to make triage a lot more efficient. The work is driven by attempts to understand the user better, on how to help him gain better cybersecurity situational awareness. They also advocate for human-in-the-loop processes, particularly in cybersecurity where humans are way better than machines at interpreting data. Nevertheless, their work is only applicable to the data triage part, and we want to address all incident response steps.

In this work we have decided to focus on inferring user intention from data. However, instead of using cybersecurity knowledge databases or directly matching previously encountered situations to answer to specific tasks, we take a different approach. Our idea is to consider the incident resolving step as a whole and to focus on the analysts performing it. To the best of our knowledge, there is no existing work trying to model cybersecurity analyst intentions during investigations to make recommendations.

4. Overview

The recommender system MIMIR is composed of two phases: a design phase, and a runtime phase. The design phase aims to provide us with the necessary input for the recommender engine that operates during the runtime phase. We present in this section the details of the two phases, summarized in Figure 2. It should be noted that a log investigation platform is used in both phases. This platform allows an analyst to investigate his data using visualizations through a graphical interface. We give a more complete description of the log description platform in Section 4.3.

4.1 Design phase

The goal of the design phase is to observe security investigations and extract the parameters of two components needed to build the recommendation engine: patterns and intentions. Thus, we gathered participants and let them conduct investigations through our log investigation platform.

During investigations conducted by analysts which we observed, we were able to interact with analysts to understand their way of thinking and how they work during an investigation. From our observations we were able to infer various goals analysts can have when investigating; we call those goals intentions (arrows **A** in Figure 2). We implemented a logging mechanism inside the investigation platform (arrow **B**). This allowed us to capture all the actions performed by analysts during their investigations. From the user actions captured we were able to extract meaningful groups of actions; they are extracted in a specific way described in Section 5.2. Our goal is then to match the meaningful groups of actions with the previously identified user intentions (arrow **C**). The matching operation we do between groups of actions and user intentions allows us to create the model that will feed our RS. The idea is to be able to materialize intentions through concrete actions.

As an example, during investigations, an analyst will often find a field value somewhat suspicious but not definitely. In need of confirmation, analysts will *deepen* their search. This intention can be realized by filtering the data according to the suspicious value found, as well as filtering the timeline in order to focus on the particular event that led to the logging of the suspicious value. The precise methodology on how to link an intention with a group of actions will be presented in Section 5.

4.2 Runtime phase

In the runtime phase we focus on how we managed to use the resulting model in order to provide recommendations. In **C**, we can see that the previously obtained data is used to build a Markov chain. Markov chains are a commonly used tool in decision-making and was the perfect way to model the probability of an analyst going from one intention to another during an investigation. Integrated to a recommendation engine, the Markov chain is now able to recommend a pattern of actions associated with the next most probable intention of an analyst. It is described in Section 5.

Our recommender system is then used during the runtime phase shown in **D**. During an investigation the recommender system will trigger a recommendation when a pattern of actions is recognized. The associated

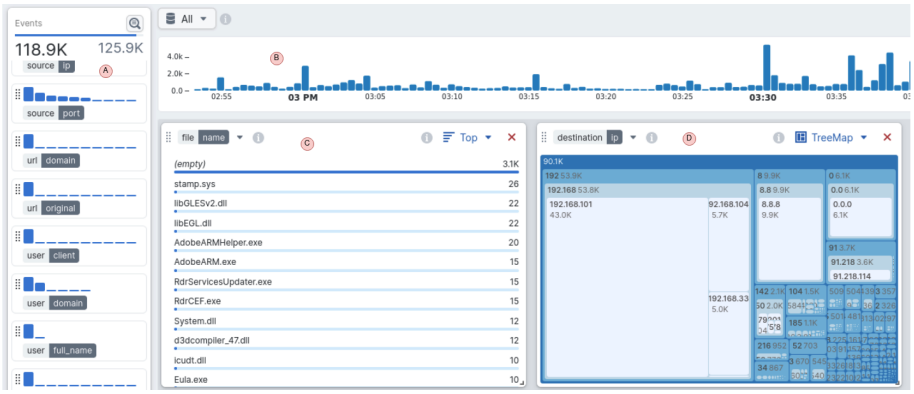

Figure 3. A screenshot of the analysis part of the log investigation platform

intention will be given to the recommender system that will then decide on the most probable intention the user can have and recommend an associated pattern of actions.

4.3 Log investigation platform

This platform helps analysts perform log analysis for incident response and threat hunting. The platform allows an analyst to push logs from any source into it. The platform will consider each line of log as a set of fields and aggregate all values among identical fields. Fields are identified using the Elastic Common Schema (ECS) [1]. That way, during investigations, analysts can visualize the values contained in the logs using the ECS fields.

The platform mainly provides a graphical interface to perform investigations. An analyst can easily investigate the logs by dragging and dropping fields from **A** to the board, creating visualizations as shown in **C** and **D** from Figure 3. These visualizations present the aggregation of values for any one data field. For example, visualization window **C** shows a visualization called Top10 for the field *file.name*. Multiple visualizations can be observed in concurrence and filters by value, range of values, and/or time can be applied and will dynamically influence the other visualization windows. Part **B** of Figure 4 shows a timeline of the logs and allows analysts to focus on a particular time interval.

In order to select only the actions relevant in the context of a cybersecurity investigation we filtered all possible actions and came up with a list of different possibilities, belonging to four different categories. The complete list of actions selected (AL), and their categories are shown in

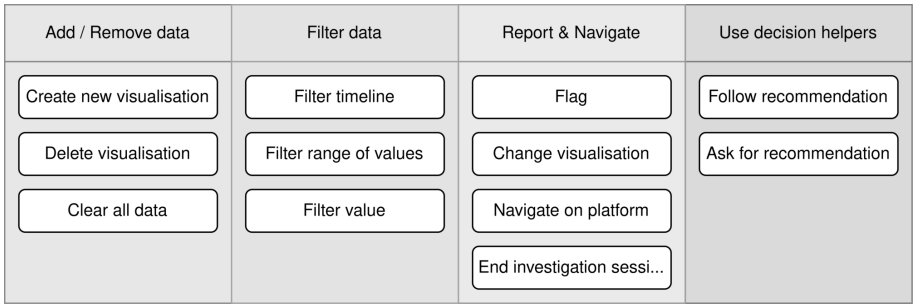

Figure 4. List of actions selected as carrying semantic intent

Figure 4. In this list there are more complex actions. Three complex actions need explanations: *flag* is the action of saving a value along with the state of the tool and some comments in order to be able to find it again and report on it easily, *change visualization* is way for the user to switch from a Top10 to a treemap for example, and *navigate on platform* is the action of changing pages in the platform. The recommender system MIMIR, described in Section 4.2 is integrated to this platform and its recommendations are triggered according to the intentions behind the actions of analysts.

5. Recommender system

In this section, we go into details regarding how we designed the recommender system.

5.1 Collecting intentions

We gathered five cybersecurity students, and presented them with two unknown datasets containing attacks to explore. The first dataset is the VAST 2012[2] dataset and the second one is the TC3[3] dataset. We briefly presented the two datasets to participants to help them start their investigations. After a presentation of how to use the investigation platform and the goal of the experiment, we gave them 30 minutes to explore the VAST 2012 dataset and 45 minutes to explore the TC3 dataset. During both explorations we waited for them to express the intentions they had, and we then observed the actions they did in the platform to carry out said intentions. We had 10 resulting investigations. From the oral discussions during the investigations we were able to infer seven different exploration intents:

- Startup / Discovery (S): corresponds to the user wanting to find an entry point into the investigation.
- Broaden search (B): adding new information to the current state of the investigation to contextualize it better.
- Deepen search (D): deepening the search which corresponds to a need for reducing the amount of data investigated; a way to deep dive into the data.
- Report findings (R): showing the intention of saving his work from the analysis: an important step in an investigation.
- Backtrack (X): returning to a previous state in the investigation by backtracking the last few actions made.
- Searching for a new lead (L): shows that even going back to a previous state of investigation would not help, and that analysts need an entirely new path.
- Guided by recommendation (M): this intention is a rare case designed to show a user guided by a recommendation.

5.2 Collecting actions and pattern creation

We have at our disposal the actions performed by the participants using the platform described in Section 4.3. This platform allows users to perform multiple actions, however they do not provide the same information. A unique action can often be linked to multiple intentions, so we decided to focus on groups of actions. We are going to define the notion of patterns. A pattern is a pair of user actions and their associated context. Each of these actions is captured as a single log line containing the action performed by the user and all the context associated to it. We define it like so because the context associated with an action can sometimes change the meaning of the action entirely.

An example of an action's meaning changed by its context would be the *filter value* action. In this case, the context of the action would the state of the filter: enabled or disabled. If the filter is being disabled, it shows that we are trying to expand the scope of our research whereas if the filter is being enabled we are restricting our scope of research: two very distinct intentions. Let us formally define what constitutes a pattern:

$$P_{ik}^{i'k'} = ((action_i, context_k), (action_{i'}, context_{k'}))$$
$$with : (i, i') \in |AL|, (k, k') \in |\{contexts\}|$$

Let us illustrate two different groups of actions using the actions *follow-recommendation* and *filter-value*. In this example the *follow recommendation* action corresponds to a recommendation made by the platform and followed by the user. The two following groups of actions with their associated context are associated with two different intentions: ((follow recommendation, ∅), (filter-value, disable)) shows the user *broadening* his scope of investigation by removing a filter after an exploration recommendation, while ((follow recommendation, ∅), (filter-value, enable)) shows the user restricting, and so *deepening*, his scope of investigation to look at a smaller part of the dataset after getting an exploration recommendation.

The pattern size was empirically set to two actions. We could have chosen a different size for patterns, but the size directly impacts the frequency at which patterns are found in investigations. We needed to find a pattern size where its frequency was high enough to reappear in later investigations. Patterns of size 3, 4 and 5 were tested but their frequencies of apparition within investigations were too low to be significant. As a result, with all possible actions and possible contexts combined, we end up with 16 unique actions, and we obtain 120 possible patterns.

5.3 Linking patterns to user intentions

With each of the ten resulting investigations, we had the necessary material to extract patterns from the investigations traces at our disposal and see if they matched specific intentions. We manually match each pattern to one intention. In very few cases, a pattern can be representative of two different intentions, in which case we arbitrarily decided on only one of them for the purpose of transparency and traceability. In order to check our matching, we also consulted two cybersecurity experts, knowledgeable about the investigation platform and asked them to list a maximum of patterns they would use to achieve each intent. The final list of patterns we use in the rest of this work are the ones located in the intersection of both sets. This gives us a list of 39 relevant patterns associated with the seven user intentions. The resulting model links concrete actions within the investigation platform with user intentions. We can now build a recommendation engine based on this model.

6. Recommendation engine

This section describes the Markov chain used to implement the model. Its goal is to detect actions during the use of the log investigation plat-

form, understand the intention behind it, and recommend the next intention.

6.1 Using a Markov chain to link intentions

In order to make recommendations we need to be able to link intentions together. Indeed, our recommender system will suggest the next intention when detecting an intention. Nevertheless, our observations and the data at our disposal showed that in a given situation analysts do not all follow the same intention, meaning there was a probability attached to going from one intention to another. Markov chains seemed to be a natural representation to model this behaviour. Thus, we implemented a Discrete Time Markov chain [16] and, in the rest of this paper we work with the Markov chains transition matrices. A state is defined by a unique intention, meaning we have seven possible states in the chain. Each state has its set of associated patterns, that allows a user to perform the intention. A transition has a probability p to go from a known intention detected from a pattern, to another intention.

From the experiment data previously obtained, we converted every investigation we had from actions to patterns, and matched them to intentions. From the sequences of intentions we found out how often one intention led to another one and were able to build a transition matrix, defining the Markov chain. Figure 5 shows this transition matrix. We can easily identify which recommendation we can make, given any intention. For example when the Backtrack (X) intention occurs, 59% of the time an analyst then performs actions that correspond to the intention of Broadening (B), which is the best recommendation to offer.

6.2 Triggering recommendations

Our goal with the pattern detection was to find out the best trigger for our recommendations. By interviewing analysts, we found out that analysts will prefer a well-timed recommendation they do not have to ask for rather than a recommendation they have to request.

To do so, we implemented a detection module that records the last user action made and, when a new one is caught, compares it with the newly detected one in order to find out if they constitute a pattern known to be associated with a specific user intention. If the pattern exists, we trigger the recommendation with the highest probability, otherwise we switch the last known action with the new one and wait for an action to be done by the user again.

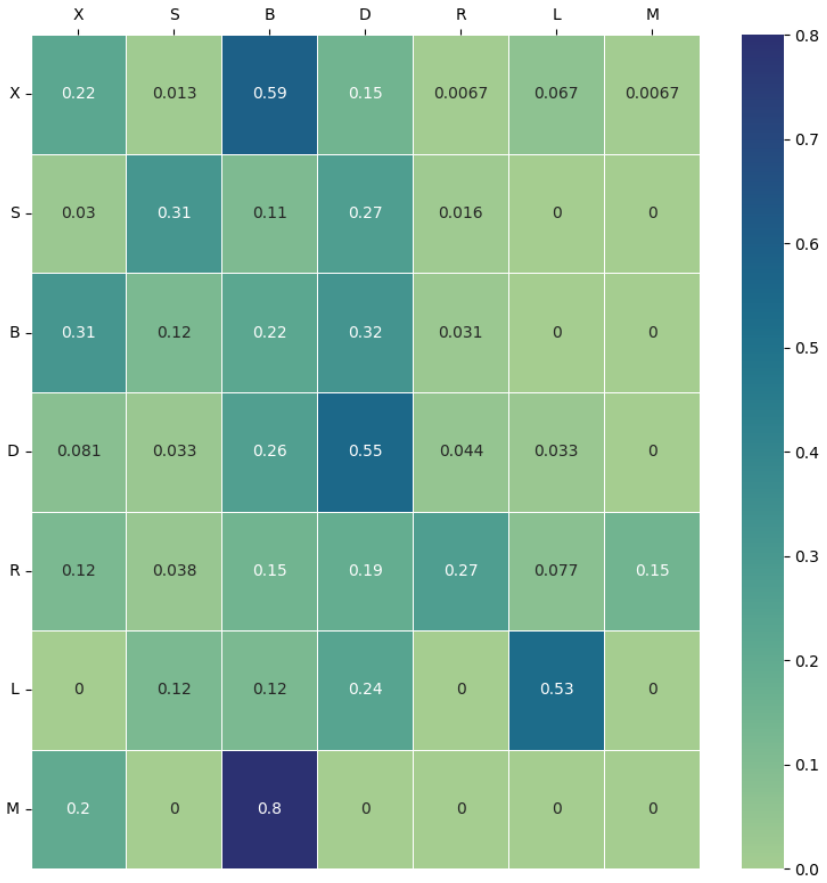

Figure 5. The transition matrix for the Markov chain with the intentions from Section 5.1

6.3 Presenting recommendations

For the instantiation of recommendations we focused on recommending actions to be performed in the investigation using the labels of Figure 4. The recommendation is the most probable next intention the user might have according to the matrix in Figure 5. Recommendations are then presented to the user by showing them the intention we predict they have as well as actions that can be performed to realize that intention.

7. Evaluation

The evaluation of this work was complex for two reasons. First, evaluating recommender systems is not standardized, and no good method

stands out. Secondly, evaluating a recommender system can be subjective. For example, an analyst could see a recommendation, find it relevant but decide to follow-up on it at a further time, marking it as not followed in the data despite it being an interesting recommendation. To make up for that we evaluated different aspects of the RS. The following sections present an evaluation of the quality of the Markov chain and an evaluation of the efficiency of recommendations using the prototype. Each section starts with a description of the data we used during the experiment. For better readability we refer to Markov chains by using their transition matrices.

7.1 Quality of the Markov chain

7.1.1 Datasets. In this section we describe sets of user investigations. An investigation is composed of a sequence of user actions. An investigation is performed on a dataset as represented in the design phase of Figure 2. All datasets used during this preliminary evaluation are described in Table 1. The VAST2012, TC3, and BotsV1 logs are available to the public. All combined, more than 85 people participated in the creation of these datasets, over 4 different occasions, with various goals and timeframes for investigations. We group investigations into sets, according to the dataset used:

1 **Reference Traces** (from datasets VAST 2012 and TC3): this set is constituted of 10 investigations, 5 on the VAST 2012 dataset and 5 on the TC3 dataset, conducted by 5 students, during an experiment that aimed to build the Markov chain in Figure 5.

2 **SUPSEC Traces**: This set is composed of 32 investigations. Each investigation was conducted by an analyst of variable experience during a blue team exercise. Each participant was given two hours to investigate the SUPSEC Dataset. They did not benefit from any exterior help besides a basic contextualization of the dataset at the beginning of the exercise.

3 **BotsV1 Traces**: This set was built at the occasion of a capture the flag event in 2023. It is composed of 48 investigations conducted by capture the flag teams. These teams were composed of cybersecurity amateurs as well as professionals. There were more than 60 teams in this exercise but only the investigations of the teams that seriously tried doing the challenge were selected, whether they succeeded or not.

Dataset	VAST 2012	TC3	SUPSEC Dataset	BotsV1
Size (# events)	23.7M	19.5M	125.9k	33.4M
Nature	Network (N)	System (S)	N & S	N & S
Investigations	5	5	32	48
Investigation time	2.5h	3.75h	64h	32h
Transition Matrices	10		32	48

Table 1. Dataset information

7.1.2 Validity of the Markov chain. For this evaluation our goal was to confirm that our approach is valid. We need to show that the Markov chain's probability of transition between user intentions make sense from a security standpoint. We perform an experiment to show that the Markov chain makes more sense than one built randomly.

For the first experiment, we used a Reference Matrix, computed from all the investigations in the **Reference Traces** set, as well as the transition matrix of each investigation from the sets **SUPSEC Traces** and **BotsV1 Traces**. These investigations were all conducted by participants that were not guided by us as we have done to build the reference matrix; they are unbiased. We also generated a set of 100 random matrices to use as a comparison point.

For each resulting transition matrix in the 2 sets and the set of random matrices, we calculated their distance to the Reference Matrix. The distance between two matrices of size *(n,n)* was calculated using the following formula:

$$d(A, B) = \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{ij} - b_{ij}|$$

Results are reported in Figure 6. We can see that the distribution of matrices built randomly has a mean around 7.2. We find the respective mean distances to the Reference Matrix of the **SUPSEC Traces** and **BotsV1 Traces** sets to be around 6 and 6.4. This shows that our Markov chain's recommendations are better than the random one.

7.1.3 Relevance of the Markov chain. For the second experiment, we wanted to confirm that we had not biased our users when we built the Reference Matrix. We want to compare transition matrices with a set of traces and see if their investigations could have been generated by it. To do so we are going to use the log-likelihood method.

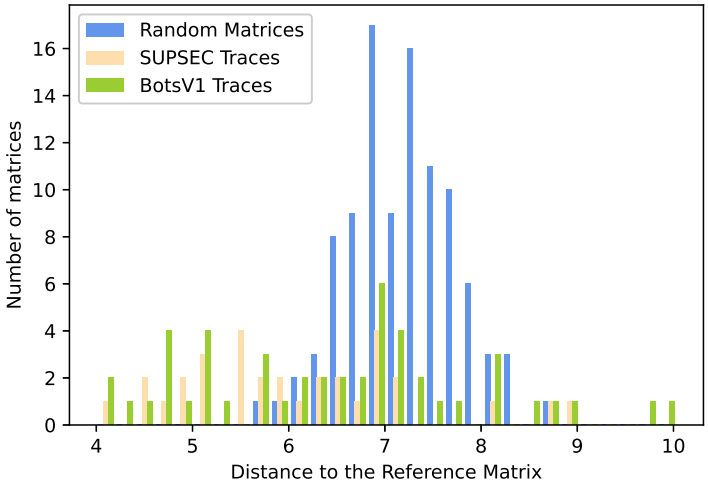

Figure 6. Representation of the distribution of the distance of multiple sets of investigation traces to a reference matrix

Results are presented in Table 2. For each investigation present in a trace of investigations (a column in Table 2), we compute the log-likelihood with the matrices that we can extract from all three sets of investigations, and an additional random matrix (duplicated 100 times). These results should be read in columns since the number of investigations is not the same for every experiment and therefore makes the likelihoods not comparable. The log-likelihood L of a sequence I to have been generated by the matrix M is for (A_i, B_i) the intention transition at the i position in the sequence I:

$$L = \sum_{i=o}^{n} ln(Transition_M(A_i, B_i))$$

The diagonal gives us a local maximum as we compare the matrix built from a set of investigations with that same set of investigations. The interesting result is that the likelihood of other matrices are close to the maximum value, except for the 100 random matrices that have a significantly lower average value. We conclude that we successfully framed the intentions of analysts during investigations.

Matrix/Sequences	**Reference Traces**	**SUPSEC Traces**	**BotsV1 Traces**
Reference Matrix	**21.556**	19.573	26.951
SUPSEC Matrix	20.317	**20.378**	26.736
BotsV1 Matrix	19.076	18.445	**27.956**
Random Matrix	10.131	9.676	14.414

Table 2. Mean log-likelihood of sequences being generated by a specific Markov chain

7.2 Prototype and recommendations evaluation

This last experiment we conducted was part of a larger security exercise organized as a red and blue team capture the flag. Due to space constraints we only discuss briefly the details of the experiment and focus on the obtained datasets.

7.2.1 Datasets. The red team event took place on variations of the same attack scenario, resulting in 13 different datasets to investigate. We selected the 5 most complete among them to be investigated. Each dataset investigated contained various attacks, network and system logs for a total number of events ranging from 1000 to 8000. The sources of data were an Auditd service on each machine in the infrastructure and a Suricata listening to the whole network. The blue team part of the exercise had 9 participants. During their investigations, the participants were using a version of the log investigation platform that integrates MIMIR. For this experiment MIMIR used a slightly less refined matrix than the Reference Matrix described in Section 7.1.2 as a recommendation engine due to time and implementation constraints.

7.2.2 Recommendations evaluation. The goal is to see whether the recommendations were followed by the user. When a recommendation is triggered we recommend the most probable transition (The highest probability of a line in Figure 5). Over 7 possible resulting transitions, only 5 were triggered during the experiment. That is not abnormal because the last 2 are rarer cases.

During the experiment, we recorded every recommendation. We considered a recommendation as followed if at least one of the actions were done in the next 2 to 3 actions following a recommendation. We consider only the case where the recommendation is followed directly or in the next 2 or 3 actions because more than that would result in too much imprecision.

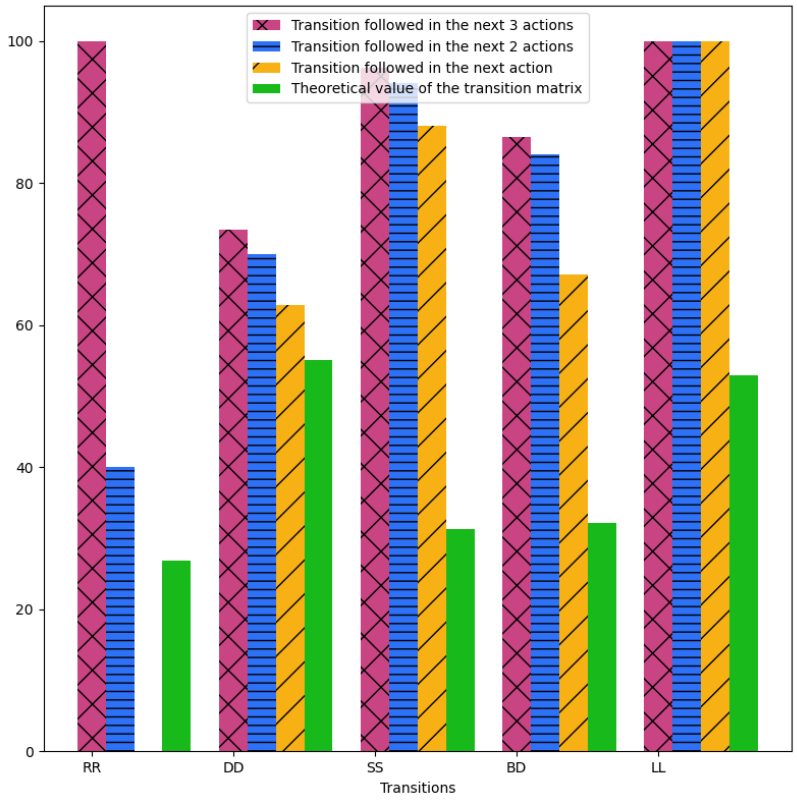

Figure 7. A representation of how much the recommendations were followed compared to the Markov chain transition probability

Figure 7 shows that for every transition considered, recommendations were followed a lot more than the theoretical value of the transition matrix anticipated. In some instances they were even always followed. There is only the case of the next action after a RR transition where we find that user never follows the recommendation directly. That can be explained by the fact that framing the reporting intention on an investigation can mean any number of things. It ranges from saving a simple reminder during the investigation to completing the investigation and leaving the platform, making it more complex to predict the next intention of the user.

We believe these results are promising. It shows that the behaviour we captured through our model and which we recommend is a behaviour that analysts tend to follow easily when recommended.

8. Conclusion

In this paper we have presented MIMIR: a recommender system that recommends exploration paths to analysts during incident response. Its engine is based on the detection of relevant user actions and their associated user intentions. With this, we built a Markov chain to help us find out the most probable intention an analyst wants to perform. We then evaluated our prototype extensively, using 5 different datasets, and through 4 different experiments. We obtained promising results, showing that our approach captures the meaning behind user intentions well and recommends actions that users tend to follow. We are currently working on improving MIMIR, especially on implementing a method to make recommendations easier to interact with. In the near future we are considering working on further substantiating the recommendations by hybridizing this recommender system with [2], and we wish to implement a learning component to the engine so that the Markov chain can self-actualise as investigations are conducted on the platform.

References

[1] Bennett, J., Lanning, S., et al.: The netflix prize. In: Proceedings of KDD cup and workshop. vol. 2007, p. 35. New York (2007)

[2] Brisse, R., Boche, S., Majorczyk, F., Lalande, J.F.: Kraken: a knowledge-based recommender system for analysts, to kick exploration up a notch. In: International Conference on Information Technology and Communications Security. pp. 1–17. Springer (2021)

[3] Burke, R.: Knowledge-based recommender systems. In: Encyclopedia of library and information systems. vol. 69, pp. 175–186 (2000)

[4] Burke, R.: Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction **12**(4), 331–370 (2002). https://doi.org/10.1023/A:1021240730564

[5] Del Esposte, A.d.M., Campiolo, R., Kon, F., Batista, D.: A collaboration model to recommend network security alerts based on the mixed hybrid approach. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)

[6] Ferreira, L., Silva, D.C., Itzazelaia, M.U.: Recommender systems in cybersecurity. Knowledge and Information Systems pp. 1–37 (2023)

[7] Franco, M.F., Rodrigues, B., Stiller, B.: Mentor: the design and evaluation of a protection services recommender system. In: 2019 fifteenth international conference on network and service management (CNSM). pp. 1–7. IEEE (2019)

[8] Grillenmeier, G.: Protecting active directory against modern threats. Network Security **2021**(11), 15–17 (2021)

[9] Husák, M., Čermák, M.: Sok: Applications and challenges of using recommender systems in cybersecurity incident handling and

response. In: Proceedings of the seventeenth International Conference on Availability, Reliability and Security. pp. 1–10 (2022)

[10] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction (2010)

[11] Moskal, S., Yang, S.J.: Translating intrusion alerts to cyberattack stages using pseudo-active transfer learning (patrl). In: 2021 IEEE Conference on Communications and Network Security (CNS). pp. 110–118. IEEE (2021)

[12] Network, E., ENISA, I.S.A.: Good practice guide for incident management (2010), https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management

[13] Pawlicka, A., Choraś, M., Pawlicki, M.: The stray sheep of cyberspace aka the actors who claim they break the law for the greater good. Personal and Ubiquitous Computing **25**(5), 843–852 (2021)

[14] Pawlicka, A., Pawlicki, M., Kozik, R., Choraś, R.S.: A systematic review of recommender systems and their applications in cybersecurity. Sensors **21**(15), 5248 (2021)

[15] Polatidis, N., Pimenidis, E., Pavlidis, M., Papastergiou, S., Mouratidis, H.: From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks. Evolving Systems **11**, 479–490 (2020)

[16] Privault, N.: Understanding Markov chains: examples and applications. Springer (2018)

[17] Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender systems handbook, pp. 1–35. Springer (2011)

[18] Soldo, F., Le, A., Markopoulou, A.: Predictive blacklisting as an implicit recommendation system. In: 2010 Proceedings IEEE INFOCOM. pp. 1–9. IEEE (2010)

[19] Sworna, Z.T., Islam, C., Babar, M.A.: Apiro: A framework for automated security tools api recommendation. ACM Transactions on Software Engineering and Methodology **32**(1), 1–42 (2023)

[20] Zhong, C., Lin, T., Liu, P., Yen, J., Chen, K.: A cyber security data triage operation retrieval system. Computers & Security **76**, 12–31 (2018). https://doi.org/10.1016/j.cose.2018.02.011

[21] Zimmerman, C.: The strategies of a world-class cybersecurity operations center. The MITRE Corporation (2014)