Exploration recommendations for the investigation of security incidents

Romain Brisse

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

CentraleSupélec

THÈSE DE DOCTORAT

CENTRALESUPÉLEC

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Informatique*

Par
Romain BRISSE

Recommandations d'exploration pour l'investigation d'incidents de sécurité

Exploration recommendations for the investigation of security incidents

Thèse présentée et soutenue à Rennes, le 15th February 2024

Rapporteurs avant soutenance :

Maciej KORCZYNSKI	Associate Professor HDR, Grenoble Alpes University
Günther PERNUL	Professor, Regensbugimprg University

Composition du Jury :

Président :	Alexandre TERMIER	Professeur des Universités, Université de Rennes
Examinateurs :	Maciej KORCZYNSKI	Associate Professor HDR, Grenoble Alpes University
	Günther PERNUL	Professor, Regensubrg University
	Laetitia LEICHTNAM	PhD, Engineer, Deveryware
	Frederic MAJORCZYK	PhD, Expert technique, DGA
	Vincent NICOMETTE	Professeur de Universités, INSA Toulouse
Invité :	Simon BOCHE	Engineer, Malizen
Dir. de thèse :	Jean-Francois LALANDE	Professeur des Universités, CentraleSupélec

Acknowledgement

Je tiens tout d'abord à exprimer ma gratitude envers toutes les personnes qui ont contribué à la réalisation de ce travail. Je débute par les membres du jury de ma thèse, présents lors de ma soutenance. Leur patience à m'écouter attentivement et leurs questions pertinentes m'ont permis d'examiner mon travail avec un regard critique. Leur bienveillance durant cet exercice parfois anxiogène pour le candidat mérite également mes remerciements. Parmi eux, je voudrais adresser des remerciements particuliers à mes encadrants et à mon directeur de thèse. À Simon, un grand merci pour son minutieux examen, sa capacité à rendre mes hypothèses exhaustives, ses discussions sur mes blocages, et son insistance à éviter les références en avant. À Frédéric, merci pour sa réceptivité à mes idées les plus folles, ses solutions ingénieuses à mes problèmes, et pour avoir toujours créé un environnement confortable, souvent à travers ses hoodies Opeth. À Jean-François, mes remerciements pour sa franchise précieuse, ses questions difficiles mais nécessaires, et ses relectures incessantes, même s'il me qualifie parfois de "pauvre flou". Un merci également à Christopher qui était toujours prêt à courir lorsque les solutions semblaient lointaines.

Je souhaite maintenant remercier ceux qui m'ont soutenu tout au long de ma thèse : CentraleSupelec, Malizen, ainsi que la DGA et l'AID. Leur soutien continu, leur confiance et leur contribution à la transformation de ce projet en recherche industrialisée sont précieux. Je remercie également mes collègues chez Malizen et chez CIDRE. Je tiens à exprimer ma gratitude envers les participants à mes évaluations, sans lesquels mon travail perdrait de sa valeur, ainsi qu'à mes collègues de l'équipe CERBERE.

Il reste cependant des remerciements à adresser à ceux qui, parfois dans l'ombre, m'ont soutenu et ont rendu la réalisation de ce travail possible. Tout d'abord, merci à Matthieu, car partager l'expérience de la thèse et les moments pour s'en détacher n'est pas donné à tout le monde. Merci également à Samantha pour son soutien indéfectible et son écoute attentive lorsque les choses semblaient difficiles. Cette dernière année fut la plus éprouvante, mais tu l'as rendue plus douce. Bien sûr, un immense merci à ma famille, toujours présente à mes côtés. Enfin, merci à Alexis, Gabriel, Anastasia et Tuong Vi, car une promesse est une promesse.

Un dernier remerciement spécial à Yotam Ottolenghi, dont les recettes m'ont nourri, pendant trois ans, et j'espère, pour de nombreuses années à venir.

Résumé substanciel en français

Au cours des dernières décennies, nous avons constaté que les technologies de l'information ont envahi tous les aspects de notre vie : la montée en puissance de la technologie a conduit à la délocalisation de toutes les formes de travail et de communication vers Internet. Les individus utilisent leurs informations et leur argent en ligne, exposant ainsi ces activités à des cyberattaques [81]. Depuis les premières cyberattaques enregistrées, les attaquants ont évolué, élargissant le panorama de la cybersécurité. Depuis le début du millénaire, les attaquants deviennent de plus en plus professionnels et organisés. Par conséquent, la nécessité de renforcer la sécurité de nos vies est plus grande que jamais.

La cybersécurité englobe de nombreux domaines, parmi lesquels les méthodes de protection les plus courantes sont les pare-feu, les systèmes de détection d'intrusion (IDS), la sensibilisation des utilisateurs, et bien d'autres. Cependant, des attaques réussissent parfois, entraînant la fuite de données, le vol d'argent et des préjudices à la réputation. Tout comme la police enquête sur les crimes, les équipes de cybersécurité disposent d'analystes spécialisés dans la réponse aux incidents. Comprendre, atténuer et remédier aux incidents de cybersécurité constitue un enjeu majeur à résoudre. Au cours de la réponse aux incidents, les analystes effectuent des investigations pour comprendre et agir. Une enquête en cybersécurité consiste à examiner les attaques et les comportements anormaux dans les données collectées à partir de n'importe quel appareil. Ces enquêtes sont précieuses, mais difficiles et chronophages, surtout parce que les analystes commencent dans l'obscurité totale, réagissant aux événements au lieu de pouvoir être proactifs.

Cette thèse se concentre sur la recherche de méthodes visant à aider les analystes à mener des investigations en cybersécurité plus rapidement et plus efficacement. Aujourd'hui, les analystes travaillent dans diverses structures telles que les Centres Opérationnels de Sécurité (SOC), les Équipes d'Intervention en Cas d'Urgence Informatique (CERT), les Équipes d'Intervention en Cas d'Incident de Sécurité Informatique (CSIRT) ou la Forensique Numérique et l'Intervention en Cas d'Incident (DFIR). Les outils de surveillance ont été les premiers à apparaître, conçus pour observer l'activité des processus et corriger d'éventuels problèmes, puis ils sont devenus un moyen de surveiller les appareils à la recherche d'une activité anormale. La détection d'intrusion a fait ses premiers pas ensuite, d'abord basée sur des signatures, puis sur le comportement, et de nos jours, l'appren-

tissage automatique accomplit un travail impressionnant pour détecter, voire atténuer et remédier automatiquement, les attaques. Cependant, les analystes sont toujours débordés, en raison de la croissance du volume, de l'hétérogénéité et de la dimensionnalité des données au cours des dernières années, et de la diversification des attaquants, de script kiddies à des menaces nationales réelles, avec des objectifs et des méthodes différents. Ces mêmes attaquants deviennent experts pour exploiter le faible niveau de cybersécurité dans le monde actuel. Même lorsque les attaques sont détectées, elles ne peuvent parfois pas être arrêtées. C'est pourquoi des équipes d'analystes, enquêtent constamment sur le système d'information de leur entreprise pour découvrir des comportements anormaux et des attaques.

Les analystes sont confrontés à de nombreux défis liés à divers aspects de leur travail : le traitement complexe des données pour les investigations, la difficulté des enquêtes et l'inadaptation des outils à leurs besoins. Conscients de cela, nous avons identifié la nécessité de trouver une autre solution. Nous avons cherché un moyen d'aider les analystes à prendre de meilleures décisions, à détecter plus rapidement les attaques et à les caractériser rapidement. En conséquence, pour cette thèse, nous avons décidé de nous concentrer sur une approche particulière des problèmes des analystes : l'utilisation de systèmes de recommandation. Un système de recommandation est un logiciel visant à suggérer des actions à un utilisateur pour l'aider à prendre de meilleures décisions concernant une tâche spécifique.

En collaboration avec Malizen, notre recherche vise à améliorer la rapidité et l'efficacité des analystes lors de leurs investigations, grâce à leur plateforme facilitant les enquêtes sur les journaux d'événements. Inspirés par le succès des systèmes de recommandation dans des domaines également submergés en données comme le commerce électronique, nous proposons d'utiliser des systèmes de recommandation dans le contexte de la cybersécurité. Ces systèmes utilisent des algorithmes pour fournir des suggestions personnalisées basées sur des connaissances, les préférences des utilisateurs, leur comportement ou leurs interactions historiques.

Questions de recherche et contributions

À la suite de nos observations, une question de recherche émerge, que nous diviserons en plusieurs hypothèses et auxquelles nous répondrons dans ce manuscrit : pouvons-nous utiliser des systèmes de recommandation pour aider les analystes lors de leurs investigations en cybersécurité ?

Afin de répondre à la question de recherche précédente, nous proposons trois contributions :

1. KRAKEN, présenté à SECITC 2021 [13]
2. MIMIR, présenté à IFIP WG 11.9 ICDF 2024 [14]
3. CERBERE, présenté à CyberHunt 2023 [11]

Le reste de ce manuscrit est organisé en deux parties distinctes, de manière à présenter le contexte et la littérature concernant nos questions de recherches, puis nos contributions. Dans le cas où une contribution nécessiterait de plonger dans de la littérature additionnelle, on pourra la trouver dans le chapitre de contribution correspondant.

Le chapitre 1 détaille le contexte de cette thèse et définit de nombreux termes que nous utiliserons tout au long du document. Ce chapitre détaille toutes les notions nécessaires à la bonne compréhension de ce manuscrit et donne une idée de l'environnement dans lequel la thèse a été réalisée. En effet, il décrit notamment le métier d'analyste et ses problématiques ainsi que les investigations, activité principale des analystes. Une section détaille également les différentes natures possibles des journaux d'événements ainsi que les challenges qui leur sont associés. On détaille finalement la plateforme d'investigation de journaux d'événements développée par Malizen à l'attention des analystes. Cette plateforme a d'ailleurs vu le jour suite à des travaux de thèse [35, 36]. Ce premier chapitre nous permet de dégager nos questions de recherche :

1. Pouvons-nous améliorer la compréhension d'un analyste pendant une investigation ? (Section 1.1)
2. Pouvons-nous tirer parti des connaissances d'experts pendant une investigation ? (Section 1.2)
3. Pouvons-nous obtenir des journaux (logs) où les attaques sont identifiées afin d'évaluer les investigations ? (Section 1.3)

Le chapitre 2 présente la littérature sur nos principaux sujets, à savoir les systèmes de recommandation et plus particulièrement l'utilisation des systèmes de recommandation en cybersécurité. Après y avoir exploré la littérature nous permettant de définir et typer les différents systèmes de recommandation, nous nous intéressons aux challenges qu'ils peuvent rencontrer dans leurs implémentations. Nous détaillons ensuite de nombreux systèmes de recommandation développés pour résoudre des tâches en cybersécurité. Tout au long du chapitre, nous essaierons de mettre en évidence les zones du domaine où des contributions sont pertinentes. Nous identifions notamment deux choses : la connaissance

experte en cybersécurité, aujourd'hui beaucoup utilisée dans des projets d'ontologie n'est que peu exploitée dans des systèmes de recommandation, et les utilisateurs et leurs besoins ne sont pas mis au centre du développement de ces systèmes. Finalement nous nous intéressons à deux sujets annexes aux systèmes de recommandation, mais néanmoins très liés à nos travaux : l'évaluation des systèmes de recommandation et la visualisation appliquée à la recommandation. Ce chapitre nous permet de préciser nos questions de recherche initiales :

R1 : Pouvons-nous tirer parti des connaissances d'experts pendant une investigation en utilisant des systèmes de recommandation basés sur la connaissance ?

R2 : Pouvons-nous améliorer la compréhension d'un analyste pendant une investigation en comprenant leurs intentions et en les utilisant pour formuler des recommandations ?

R3 : Pouvons-nous obtenir des journaux d'événements où les attaques sont identifiées afin d'évaluer l'utilisation des systèmes de recommandation pendant les investigations ?

Avec ces nouvelles perspectives de recherche, nous entamons la deuxième partie du manuscrit. Le chapitre 3 présente notre première contribution, KRAKEN, un système de recommandation qui s'appuie sur de la connaissance experte pour guider les analystes en leur proposant des chemins d'exploration pertinents pendant leurs investigations, sous la forme de champs à explorer (*e.g.* destination.ip). La base de connaissances de KRAKEN est construite à partir des projets Mitre ATT&CK et Elastic Common Schema, permettant de lier les comportements d'attaques décrits par l'un avec les descriptions de journaux d'événements de l'autre. Ainsi, lors d'une investigation réelle, KRAKEN peut recommander des journaux d'événements pertinents à explorer. Nous évaluons KRAKEN au cours d'une expérience impliquant 7 experts en cybersécurité, démontrant la qualité de ses recommandations. Cependant, cette première expérience met en lumière la complexité de reconnaître et comprendre les comportements des analystes pendant une investigation.

Notre seconde contribution, exposée dans le chapitre 4, se concentre spécifiquement sur la compréhension des intentions des utilisateurs pendant leurs investigations. MIMIR est un système de recommandation qui se base sur les traces des investigations passées des analystes pour identifier les groupes d'actions récurrents et les intentions qui y sont associées. Nous modélisons le lien entre les groupes d'actions récurrents, les intentions utilisateur et l'enchaînement de ces intentions à l'aide d'une chaîne de Markov. MIMIR offre des recommandations d'actions correspondant aux intentions les plus probables de

l'analyste au cours de son investigation. La chaîne de Markov est évaluée de deux façons différentes pour tester à quel point nous avons pu capturer le comportement d'un analyste dans le modèle. Dans un second temps, nous testons le système de recommandation au cours d'une expérience avec 9 participants, constatant à nouveau que les recommandations sont largement suivies par les utilisateurs, un résultat prometteur.

La dernière expérience faite pour évaluer MIMIR fait partie d'une expérience plus vaste, constituant notre troisième contribution : CERBERE. Cet exercice implique à la fois des attaquants et des défenseurs, offrant différents niveaux de difficulté. Le scénario et l'architecture de CERBERE sont fournis et peuvent être modifiés pour créer un nouvel environnement de jeu. Trois types de journaux d'événements peuvent être extraits automatiquement d'un exercice joué, ce qui est particulièrement utile pour évaluer la détection d'intrusions. Outre l'exercice lui-même et son utilité pour l'évaluation de nos systèmes de recommandations, nous fournissons le jeu de données résultant de la première édition de l'exercice. Ce jeu de données contient les journaux d'événements des 13 membres de la red team pour les trois types de journaux d'événements capturés. De plus, nous avons étiqueté le jeu de données à la suite l'exercice afin d'en extraire la vérité terrain, en particulier les événements représentant des attaques réussies, rendant le jeu de données aussi utilisable que possible pour les futures recherches, quelle que soit l'application.

Nous concluons cette thèse en récapitulant nos hypothèses de recherche et nos contributions, puis nous suggérons de nouvelles perspectives de travail. Après avoir examiné des améliorations potentielles pour chacune de nos trois contributions, nous proposons des pistes d'amélioration. L'une de ces propositions consiste à explorer la possibilité d'hybrider les systèmes de recommandation KRAKEN et MIMIR, combinant ainsi leurs spécificités pour créer un nouvel outil plus performant. Nous suggérons également d'explorer les aspects temporels des investigations, en envisageant des recommandations liées à l'ordre des étapes effectuées par un attaquant. Enfin, une dernière perspective envisagée consiste à élargir le champ de nos recommandations pour inclure la possibilité d'offrir des conseils de remédiation et de mitigation d'attaques.

Abstract

The landscape of cybersecurity has expanded significantly. In the new millennium, attackers have evolved into highly skilled and organized professionals. Consequently, there is an increasing need to enhance the security measures in our daily lives. Companies notably live and breathe thanks to their information systems, and a breach of those could have devastating impacts. In order to counter as many attackers as possible, some solutions have been installed. The ones we focus on are the incident response teams that can be found in Security Operation Centers (SOC), Computer Emergency Response Teams (CERT) or Computer Security Incident Response Team (CSIRT). Within these teams, the role we focus on in this work is the one of analyst. An analyst is a professional responsible for monitoring, analysing, and interpreting security-related data to identify potential threats, vulnerabilities, and security incidents within an organization's information systems.

In the cybersecurity domain, a shortage of analysts persists, primarily attributed to challenges in handling the escalating volume of threats. Despite utilizing diverse tools, technologies, and methodologies to identify and counter security breaches, assess incident impact, and fortify security postures, analysts encounter difficulties due to the tools' lack of alignment with their specific tasks. Many existing tools are not purpose-built for the intricacies of analysts' responsibilities, impeding ease of use and familiarity. This research aims to address these challenges by proposing complementary tools designed to augment analysts' capabilities during incident response, offering enhanced support in their data investigation for threat detection.

In collaboration with Malizen, a platform facilitating log investigations for analysts, our research aims to enhance the speed and efficiency of analysts during their investigative processes. Inspired by the success of recommender systems in data-intensive domains like e-commerce, we propose employing recommender systems in the cybersecurity context. These systems leverage algorithms to deliver personalized suggestions based on knowledge, user preferences, behaviour, or historical interactions.

This thesis outlines two distinct recommender systems introduced as contributions. One relies on domain-specific cybersecurity knowledge, while the other utilizes analysts' past interactions with the investigation platform to make recommendations. Despite their different approaches, both systems share a common objective: providing analysts with

pertinent exploration paths throughout their investigations. The thesis also delves into the general evaluation of recommender systems as well as cybersecurity data generation, constituting our third and final contribution.

Table of Contents

Part I

Context

Introduction

The difference between screwing around and science is writing it down.

— **Adam Savage**

During the past few decades we have seen information technology take over every aspect of our lives: the rise of technology has led to the relocation of every and any form of work and communication to the internet. People use their information and money online and cyberattacks targeting them are committed. Since the first recorded cyberattacks [81], attackers have evolved and now the panorama of cybersecurity is wider than ever. Since the beginning of the millennia, attackers are becoming more and more professional, they have become organized. As a consequence, the need to update the security level of our lives is greater than ever.

There are numerous fields in cybersecurity, among the most common methods of protection are firewalls, Intrusion Detection Systems (IDS), sensitization of users and many others. However, attacks still get through sometimes and data gets leaked, money is stolen, reputations are harmed. Just as the police investigates crimes, cybersecurity teams have analysts devoted to incident response. Understanding, mitigating and remediating cybersecurity incidents is among the important problematics to solve in cybersecurity. During incident response, in order to understand and then act, analysts perform investigations. A cybersecurity investigation is the process of researching attacks and abnormal behaviours within data collected from any device. They are a coveted resource because their work is hard and time-consuming and yet, time critical. Investigations are also even more difficult since analysts start them completely in the dark, making the analyst react to events, instead of being able to be proactive.

The work of this thesis focuses on finding methods to help analysts conduct cybersecurity investigations faster and more efficiently. Nowadays, analysts are employed in various structures such as Security Operations Centers (SOC), Computer Emergency Response Team (CERT), Computer Security Incident Response Team (CSIRT) or Digital Forensics and Incident Response (DFIR). Information about these structures show us how necessary it is to address the issue of making the job of analyst easier, and faster. Monitoring

tools were the first to appear. They were design to observe the activity of processes and correct any issues, then moved on to being a way to monitor devices in search of abnormal activity. Intrusion detection took its first steps next, first signature based, then behaviour based and nowadays, machine-learning based. They do an impressive job of automatically detecting attacks, and even mitigating and remediating to said attacks. However, analysts are still overwhelmed. First because the volume, heterogeneity and dimensionality of data has grown during the last few years. Then because attackers have diversified themselves endlessly and have access to more and more resources since cybersecurity became popular. They range from script kiddies to actual nation state threats and have different goals and methods. These same attackers are becoming experts at taking advantage of the low level of cybersecurity in the world right now. Even when the attacks are detected, sometimes they simply cannot be stopped. That is why teams of analysts, called threat hunters, are constantly investigating their company's information system in order to discover abnormal behaviours and attacks.

Analysts are faced with many challenges regarding various aspects of their work: data is complex to process for investigations, investigations are difficult to conduct, and the tools at the analysts' disposal are not adapted to their need. Knowing this, we found there was a need to come up with another solution. We went in search of a way to help analysts make better decisions and detect attacks faster and characterize them readily. As a result, for this thesis, we have decided to focus on a particular approach to the analysts' problems: the use of recommender systems. A recommender system is a piece of software aiming to suggest actions to a user to help him make better decisions regarding a specific task.

Research question & contributions

Following our observations a research question emerges, which we will split into several hypotheses, and answer in this manuscript: can we use recommender systems to help analysts during their cybersecurity investigations?

In order to provide an answer to the previous research question we propose three contributions:

1. A recommender system based on expert knowledge in cybersecurity. This tool called KRAKEN [13] aims to use cybersecurity knowledge to identify relevant exploration paths for analysts to take during their investigations. It helps to test

hypotheses easily during investigations.

2. A second recommender system, called MIMIR, this time based on the recognition of user intentions during investigations. If we can recognize what tasks users are trying to accomplish, we can advise them the best ways to execute them, and in doing so making their investigations faster and more efficient.
3. A framework for building a blue and red team cybersecurity exercise: CERBERE. It had two aims: obtaining data usable to further our research, and evaluate our recommender systems.

Publications

Each of these contributions is associated to a publication:

1. KRAKEN, presented at SECITC in November 2021 [13]
2. MIMIR, Presented at IFIP WG 11.9 ICDF in Januanry 2024 [14]
3. CERBERE, presented at CyberHunt in December 2023 [11]

The rest of this manuscript is organized in the following way. Chapter 1 details some background about this thesis and defines many terms we will use throughout the document. Then, Chapter 2 presents the literature about our main topics, namely recommender systems and more specifically the use of recommender systems in cybersecurity. In the second part of this manuscript, we present the two recommender systems we contributed in Chapters 3 and 4, followed by our third contribution in Chapter 5. We will then conclude on this thesis in an ultimate chapter.

Chapter 1

Background

A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools.

— Douglas Adams, Mostly Harmless

In order to be able to work with recommender systems, we need to explain some core concepts of the parts of cybersecurity on which we focus. Our main point of interest is to offer help to analysts. Analysts perform investigations and report on it. However, they face numerous issues in their investigations. First, the quantity of data to investigate is often humongous. Indeed, even small companies use many devices that log information about their use, and the attacks can come from any one device, causing the quantity of data to be analysed to grow a lot. Then, each different devices that records logs will do so in its own manner, according to its own constraints and so, often, the logs obtained from various sources are extremely heterogeneous, adding to the problem of quantity the difficulty of finding where to look for attacks. Finally, to perform these investigations, analysts need tools. These tools must be able to obtain and process the data in order to present it in an investigable manner to the analyst. The variety of tools available, to aggregate the data, reduce redundancy in it, uniformize its shape or even simply present it properly to analysts is huge.

All these aspects make the work of analysts difficult. In the next sections we will present in more detail these difficulties. Because we intend to make recommendations to analysts, we will also present investigations, their main activity and the nature of the data they explore: logs. Finally, we present the specificities of the Malizen platform.

1.1 Analysts

The human component is at the center of this work. We need to understand analysts as well as possible if we are to find a way to help them without replacing them. Indeed, it

is with analysts that the expertise resides and empowering them would help in tasks such as threat hunting. We also believe that offering "human-in-the-loop" solutions approaches problems like attack detection, mitigation and remediation that have hit walls in efficiency these last few years, differently [87]. Our focus is to provide decision-making help to analysts and not to detect or mitigate discovered attack in their place. That is because we believe in their expertise and in the fact that if we facilitate their tasks they can focus on making good security decisions without relying on automatic tools whose methods are not always understood. In the following sections we will first detail the types of cybersecurity jobs there are and where to find analysts there. Then we will discuss the two standpoints of trying to automate their tasks versus trying to help them perform them more efficiently.

1.1.1 Analyst types & expertise

According to the structure they are a part of, analysts have different missions. Hereafter we describe some structures where analysts are commonly employed.

First, is the most well-known: the Security Operations Center (SOC) [94]. A SOC is a centralized facility or team within an organization or externalized, staffed with cybersecurity analysts, responsible for monitoring, detecting, analysing, and responding to security incidents and threats in real-time. The primary goal of a SOC is to protect an organization's information systems, networks, and data from various cyber threats, such as malware, unauthorized access, data breaches, and other security vulnerabilities. Their daily missions include monitoring security events and analysing alerts raised. When an alert is raised either they are a trivial threat and often, the response process is automated in playbooks, but most of the time they require deeper analysis and call for an investigation. The SOC plays a critical role in maintaining the security and integrity of an organization's digital assets.

A Computer Security Incident Response Team (CSIRT) is a dedicated group or organization within an institution, company, or government agency composed of cybersecurity experts, including analysts, responsible for managing and responding to cybersecurity incidents. CSIRTs are composed of experts and professionals in the field of cybersecurity and incident response, including analysts. Their primary role is to react to detected threats, investigate, mitigate, and recover from them. Their role is more reactive compared to SOC which are more proactive and preventive, even though the missions overlap. CSIRT analysts play a key role in assessing and understanding security incidents, identifying vulnerabilities, and recommending countermeasures. They play a crucial role in mini-

mizing the impact of security breaches and ensuring the continuity of digital operations. CSIRTs typically have established procedures and workflows for incident response, maintain communication with various stakeholders, and collaborate with other security teams to address and resolve security issues effectively.

Another example of a team where analysts are employed is a Computer Emergency Response Team (CERT). A CERT is a specialized group or organization, responsible for monitoring and responding to cybersecurity incidents, vulnerabilities, and emergencies within an organization, industry sector, or even at a national level. CERTs are dedicated to maintaining the security and resilience of digital systems and networks. These teams, including cybersecurity analysts, are focused on identifying, analysing, and mitigating security threats, vulnerabilities, and incidents. CERT analysts play a pivotal role in incident investigation, and recommending remediation actions to protect against cyberattacks and vulnerabilities. They work collaboratively to ensure the timely response to cybersecurity emergencies, enhance cybersecurity awareness, and provide guidance and support to prevent future incidents. CERTs typically follow established procedures and protocols for incident response and often engage in information sharing and collaboration with other CERTs, security teams, and stakeholders to address and manage cybersecurity challenges effectively.

There are also teams tasked with one specific mission, such as threat hunting. A threat hunting team is a group of cybersecurity professionals, with specialized expertise in proactively seeking out and identifying potential security threats and vulnerabilities within an organization's digital infrastructure. This team's primary mission is to actively search for signs of malicious activities, advanced persistent threats, and cybersecurity risks that may have evaded traditional security measures. Threat hunting analysts leverage a combination of tools, techniques, and their cybersecurity knowledge to continuously monitor networks, systems, and data for suspicious activities or indicators of compromise. When potential threats are identified, these analysts explore data during in-depth investigations to understand the nature of the threat, its scope, and the potential impact. They also recommend and implement mitigation strategies to prevent security incidents and data breaches. Threat hunting teams play a crucial role in enhancing an organization's cybersecurity posture by detecting and neutralizing threats before they can cause significant damage. Their work contributes to a more proactive and resilient security strategy.

Finally, sometimes, a cybersecurity analyst can be specialized in a specific task. For example, audit and risk assessment plays a pivotal role in identifying and mitigating secu-

rity risks within an organization's digital landscape. Analysts specialized in such matters have responsibilities that include conducting security audits to ensure compliance with regulations, assessing vulnerabilities, analysing risk factors, monitoring and enforcing compliance, simulating real-world attacks, generating detailed reports with mitigation recommendations, shaping security policies, preparing for incident responses, fostering security awareness, and maintaining continuous surveillance of the threat landscape. These efforts collectively bolster an organization's security posture, facilitate regulatory compliance, and proactively manage cybersecurity risks.

Analysts can be found at different posts and have different goals but as described in this section some common ground is always found. Apart from the last few profiles described, their main goal is always to uncover attacks, analyse and report on them, through investigations.

1.1.2 Analyst experience

The important thing to discuss here is the fact that nowadays, cybersecurity is part of everyone's daily life. Even if we focus only on companies, some bigger ones or the ones better sensibilized to the topic will maybe have people dedicated to maintaining security levels and protecting the infrastructure and its users. However, in many other companies, often smaller structures and older ones, someone from IT is assigned this job and has to perform outside his area of expertise.

So far, my experience in the domain shows that tools at the disposal of analysts are over-specialized and as a result cannot be used properly. Even worse, they can be so complicated to handle that they provoke mistakes and create more attack surface.

However, we are still able to face the challenges of cybersecurity because of too few experienced cybersecurity experts. So, we have decided to focus on working with solutions that help analysts make the best out of their own expertise or to rely on the expertise of others, without trying to replace them, because in the last twenty years, cybersecurity has not found a magic detection, mitigation and remediation solution to every attack.

1.2 Incident response

Incident response refers to a systematic approach taken by organizations to manage and mitigate security incidents, such as data breaches, cyberattacks, malware infections,

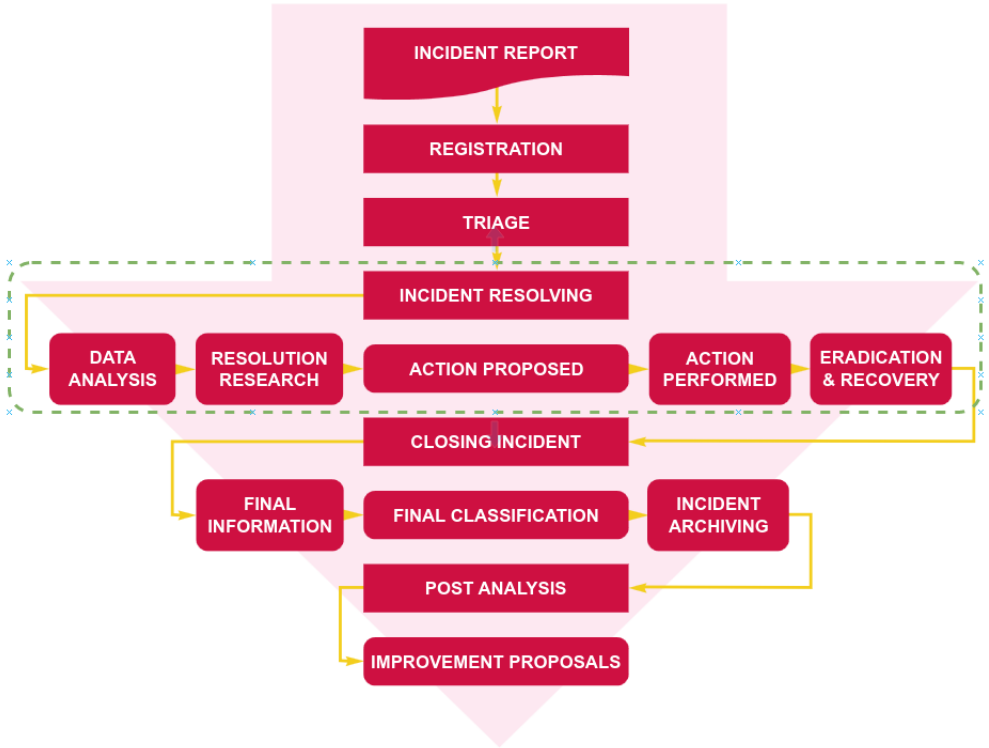

Figure 1.1 – The incident response workflow from ENISA [57]

and other cybersecurity threats. The primary objectives of incident response are to identify and assess security incidents, contain the damage, eradicate the threat, and recover normal operations. Figure 1.1 shows a workflow offered by the ENISA to map the process and different steps of incident response.

This process typically involves the coordination of various teams, the collection of evidence, communication with decision-makers at all levels, and the implementation of predefined procedures and security controls to address the incident effectively. Handling properly the management of incidents is a critical component of an organization's overall cybersecurity strategy. In the following sections we will go into more detail about incident reporting, registration and, triage. Then, we will discuss incident resolving where our focus is on the highlighted part in Figure 1.1: incident resolution, and more specifically the investigation. Finally, we will briefly develop the closing of an incident.

1.2.1 Incident reporting registration and triage

In order to reach the steps of incident reporting, registration or triage, incidents must first be detected. However, once alerts are raised by the detection system, the trivial ones can be treated automatically, but most cannot. At this point, an analyst takes over and decides what must be done regarding the alert.

Reporting and registration are fairly straightforward tasks. When reporting and registering an incident, the analyst answers some questions such as:

— What alert (or group of alerts) qualifies as an incident?
— How to register it? (What process should it go through?)
— What data is important to collect so that incidents are documented well enough?

These steps are not usually the one where decision-making is made, therefore, recommender systems are not needed at this point. Independently of the method used to report, may it be an email or a complex ticketing platform, an incident is registered and assigned to an analyst.

This leads us to the triage phase, where the fun begins. Triage involves classifying incidents into different categories, such as high, medium, or low severity, and then prioritizing the response efforts accordingly. This classification problem is well known by the community [93, 21]. Having an efficient way to triage the incidents is crucial because resolving an incident necessitates the right analyst with the right expertise.

1.2.2 Incident resolving & Investigation

Incident resolution is very context-dependant. Every team has its own set of tools, its own environments and processes to eradicate threats. That is due to various reasons, historical, analyst preferences, company policy etc. However, some common ground can be found in every team: incident resolution consists in multiple iterations of data analysis in order to find, comprehend, mitigate and then remediate an attack. The multiple iterations can be for different reasons such as the apparition of new data to analyse or the mitigation not working. Depending on the incident, handling it can be automated. Instances where an incident is often resolved automatically are DDoS attacks (Distributed Denial of Service) because detection tools nowadays have become very efficient at recognizing them early on, sometimes even earlier than analysts like in this example of a DDoS attack first mainly

blocked automatically [1] [2]. However, when it became clear the automated response was not sufficient, analysts started to investigate.

To summarize, a cybersecurity investigation is a comprehensive and methodical process undertaken to uncover, analyse, and respond to security incidents, breaches, or suspicious activities within an organization's digital infrastructure. This investigative procedure consists in the exploration of data by cybersecurity professionals, often known as (forensic) analysts or investigators, who possess specialized skills in collecting, preserving, and analysing digital evidence. Investigations differ a lot based on what data is available, how and when it was collected, the nature of the data, and of course the quantity. Nonetheless, the primary steps of a cybersecurity investigation are:

- **Identification**: This can include unauthorized access, data breaches, malware infections, insider threats, or any activity that raises suspicion.
- **Analysis**: Once evidence is identified, analysts analyse it to determine the nature and scope of the incident. This may involve examining logs, network traffic, system configurations, and artefacts left by attackers.
- **Attribution**: In some cases, analysts attempt to identify the source or the individuals behind the incident. Attribution is often challenging but can be essential for law enforcement or legal actions.
- **Mitigation**: During the analysis (and after) of the incident, analysts recommend and implement measures to contain and mitigate the security threat. This can include isolating compromised systems, removing malware, and patching vulnerabilities.
- **Documentation**: Throughout the investigation, meticulous documentation of all findings and actions taken is crucial for both legal and internal purposes.
- **Communication**: Investigators maintain clear communication with relevant stakeholders, including management, IT teams, legal counsel, and law enforcement if necessary.
- **Remediation**: Once the investigation is complete, recommendations for remediation are provided to prevent a recurrence of the incident. This can involve enhancing security controls and procedures.
- **Reporting**: A report is often generated, summarizing the investigation's findings, including the incident's cause, impact, and recommendations for future security

1. https://blog.cloudflare.com/zero-day-rapid-reset-http2-record-breaking-ddos-attack/
2. https://blog.cloudflare.com/technical-breakdown-http2-rapid-reset-ddos-attack/

improvements.

1.2.2.1 MITRE

In order to easily identify attacks discovered, classify them and find mitigations and remediation, analysts often use knowledge they can share between them. At this point we find it worth mentioning the MITRE® corporation. Specifically we wish to introduce two projects from this company: ATT&CK and D3FEND.

ATT&CK The MITRE ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) project is a comprehensive framework that aims to provide a detailed and structured understanding of the tactics, techniques, and procedures (TTPs) employed by cyber adversaries during different stages of a cyberattack. The project's primary goals are enhancing cybersecurity awareness, improving threat detection and response, and facilitating collaboration within the cybersecurity community. Other projects such as SIGMA [3], Osquery-ATT&CK [4] or RE&CT [5] perform similar joins between projects for other cybersecurity tasks than investigation, showing how widespread and used by all the ATT&CK project has become over the years. Some information about ATT&CK is given hereafter:

Tactics represent high-level objectives or goals that adversaries aim to achieve during an attack. They are categorized into several areas, each reflecting a different aspect of the attack lifecycle. The tactics in ATT&CK are as follows:

1. Reconnaissance: The adversary is trying to gather information they can use to plan future operations.
2. Resource Development: The adversary is trying to establish resources they can use to support operations.
3. Initial Access: The adversary is trying to get into your network.
4. Execution: The adversary is trying to run malicious code.
5. Persistence: The adversary is trying to maintain their foothold.
6. Privilege Escalation: The adversary is trying to gain higher-level permissions.
7. Defence Evasion: The adversary is trying to avoid being detected.
8. Credential Access: The adversary is trying to steal account names and passwords.

3. `https://github.com/SigmaHQ/sigma`
4. `https://github.com/teoseller/osquery-attck`
5. `https://github.com/atc-project/atc-react`

9. Discovery: The adversary is trying to figure out your environment.
10. Lateral Movement: The adversary is trying to move through your environment.
11. Collection: The adversary is trying to gather data of interest to their goal.
12. Command and Control: The adversary is trying to communicate with compromised systems to control them.
13. Exfiltration: The adversary is trying to steal data.
14. Impact: The adversary is trying to manipulate, interrupt, or destroy your systems and data.

Techniques are specific methods or procedures used by adversaries to achieve the objectives defined in the tactics. They provide more granular details about the steps attackers take. Techniques are organized under each tactic. There are many techniques and sub-techniques, but here are some examples:

1. Exploit public-facing application: an internet facing application is an attacker's playground. They will poke at them until they find an exploitable vulnerability. It is often the technique used in the initial access step.
2. Ingress tool transfer: an attacker is trying to transfer a tool or a piece of malicious code (*e.g.* a malware) through a channel in order to continue his attack or launch new ones. It is associated to a command and control tactic.

Understanding these tactics and techniques helps security professionals and organizations enhance their threat intelligence, detection, and response capabilities by mapping observed behaviours to the ATT&CK matrix. It provides a common language and framework for discussing and analysing cyber threats. Using this ontology it becomes much easier to classify the actual attacks, called procedures.

D3FEND D3FEND was introduced later by Kaloroumakis *et al.* [41]. In this paper, the authors discuss their efforts to create a comprehensive knowledge graph of cybersecurity countermeasures, aiming for precision and clarity in identifying and specifying the components and capabilities of such countermeasures. Their work, called D3FEND, involves encoding a knowledge base into a knowledge graph with well-defined concepts and relations, grounded in cybersecurity literature references. The graph enables inquiries that link cybersecurity countermeasures to offensive tactics, techniques, and procedures (TTPs) from the ATT&CK matrix. The authors also outline plans to expand D3FEND by incorporating research literature and utilizing machine learning for ongoing maintenance.

Through these projects, MITRE provides analysts with a common base of expert knowledge in cybersecurity that can help them in various aspect of their work. ATT&CK can be used to better identify and classify threats during investigations while D3FEND could be used to offer mitigation and remediation options after an investigation has uncovered attacks.

1.2.3 Incident closing and capitalization

The most important thing that can be done when an investigation is complete is not to lose the intelligence gained. Multiple actions can be conducted such as: writing a report, generating CTI, or building new rules for a SIEM (Security Information and Event Management), IDS [24, 70, 63], or XDR (eXtended Detection and Response) for example. A written report is almost equally important to actually conducting the investigation since someone (a technical expert or not) will want to understand the situation at one point. This corresponds to the last two steps of an investigation, as presented in Section 1.2.2.

1.3 Logs

During their investigations, analysts explore various data sources, the main type of source being logs. A log is the recording of an event that happened on a device. These records typically include detailed information about specific actions, errors, security events, or system performance metrics. They are identified by the action they record, the timestamp of the moment the action was performed, and some context associated to the action. Logs can be recorded at almost any scale, from the small module of a software to operating system-wide (*e.g.* a machine, server, IoT, probe). Logs are essential for troubleshooting, monitoring, auditing, and security analysis [25], as they provide a historical record of what has occurred within a digital environment. They are often stored in files and are regularly reviewed by system administrators, analysts, and security experts to ensure the integrity, security, and efficient operation of computer systems and networks. As a result, logs are of varying natures, present many challenges and, since they are the staple of investigations, will be discussed in more detail in the following sections.

1.3.1 Types of logs

Logs often come from one of two types or source: system or network. System logs are specific to individual devices and provide information about those devices, while network logs focus on network-wide traffic and security, helping to monitor and protect the entire network infrastructure. Both types of logs play vital roles in maintaining the integrity and security of computer systems and networks. Along with logs come metrics, closely linked to them, yet very different in nature (*e.g.* CPU load).

1.3.1.1 System

System logs, also known as server logs or host logs, are logs generated by a computer system, server, or individual device. These logs document events and activities that occur on the specific device where they are generated. System logs include information related to the operating system, hardware, and software running on the device. They track activities such as system startups and shutdowns, software installations and updates, hardware failures, and security-related events like login attempts and specific file accesses. A lot of things are "non human-readable" within those logs and often need enrichments so that the data is usable (*e.g.* identifiers, log levels, protocol numbers, specific codes in equipments). System logs are primarily used for monitoring and troubleshooting the health and performance of individual devices, as well as for identifying and responding to security incidents that may occur on the device itself.

The important specificity of system logs, is the monitoring of system calls. Having this kind of events granularity can be essential during investigations, however there are many pitfalls in using them. Indeed, tools that perform system monitoring are often difficult to configure because of side effects, such as mixing system calls logs with process logs, or the low-level nature of system calls. Most times, logs simply monitor a few syscalls, knowing which of them are the most commonly used during attacks (*e.g.* write, mprotect). Regardless, system logs give information such as the machine capabilities, memory state, the entire filesystem and, users and privileges. Such information is often crucial for a security investigation, so we need to make sure it is available.

1.3.1.2 Network

Network logs are logs generated by network devices, such as routers, firewalls, switches, and intrusion detection systems. These logs record network-related activities and commu-

nications generally called flows. They typically contain information about network traffic, such as source and destination IP addresses, port numbers, protocol types, and traffic patterns. They also log events related to network security, like intrusion attempts, firewall rule violations, and network anomalies. Network logs are complex to handle for multiple reasons. First, they generally are TCP and UDP connections. This means that logging each step of a flow (e.g. the three or four steps of a handshake) is probably counterproductive. All the while, monitoring flows to log more significant steps of the flow can lead to missing information and metadata. Second, the representation of network logs is an ongoing topic in research [48].

Network logs are essential for monitoring and securing a network. Installing a network probe is easier than configuring every machine, and allows tracking devices coming from outside the company. However, network logs have higher granularity than system logs, making both of them essential to investigations. Additionally, with the rise of remote working, network logs are more difficult to capture.

1.3.1.3 Metrics

Metrics are an entirely different breed of information. They are not logs *per se* but are often analysed alongside them [6]. In the context of information systems and cybersecurity, metrics refer to quantifiable measures and indicators that are used to assess, evaluate, and track various aspects of an organization's cybersecurity posture and the performance of its information systems. These metrics are crucial for understanding the effectiveness of security measures, identifying vulnerabilities, and making informed decisions to enhance overall cybersecurity. Common cybersecurity metrics include CPU load and network traffic volume, but also metrics used as Key Performance Indicators (KPI) such as the number of security incidents, the time to detect and respond to incidents, the percentage of patched vulnerabilities, or even user awareness training completion rates, all in order to apprehend easily the security level of a company. There are also some metrics about resource usage or flow size for example that can directly help during investigations.

1.3.2 Usability difficulties

Logs also present some significant challenges in order to be usable during cybersecurity investigations.

6. https://www.splunk.com/en_us/blog/learn/logs-vs-metrics.html

1.3.2.1 Heterogeneity

The problem of heterogeneity in logs refers to the complications that arise when dealing with logs from different sources, systems, or applications that have diverse formats, structures, and content. There is no actual standard that everybody could use. This is not even for lack of trying but because depending on the use of logs intended by the standard, the model can vary a lot. One specific project called IDMEF (The Intrusion Detection Message Exchange Format) for a standard in intrusion detection messages was even offered in a request for comments [7] but was never really adopted by the community. Depending on their intended use, logs should not be represented the same way. Here are a few common reasons for log heterogeneity that impact security investigations:

— Diverse Formats: Logs may be generated by various devices and systems, each with its own format, making manipulation challenging.

— Inconsistent Content: Log information can differ significantly between different sources, complicating analysis.

— Volume and Velocity: Vast amounts of log data are generated, with varying volumes and velocities, requiring efficient storage and processing to avoid errors and confusing analysts when sources have very different velocities.

To address the problem of heterogeneity in logs, organizations often employ log management and SIEM solutions [8] [9] that include tools for log normalization, parsing, and aggregation. These solutions can help standardize log data and facilitate centralized analysis and reporting.

Additionally, the use of common log formats, such as the Common Event Format (CEF) or Log Event Extended Format (LEEF), can aid in log consistency and integration. Overall, managing log heterogeneity is essential for effective cybersecurity and incident response.

1.3.2.2 Multidimensionality

Logs often consist of data points that capture various attributes, characteristics, or context related to an event or activity. This complexity can create challenges in log analysis and interpretation. One log line can reasonably contain up to 30 different fields, with different values, some being very redundant (*e.g.* process.name and command.name are

7. https://www.rfc-editor.org/rfc/rfc4765
8. https://docs.splunk.com/Documentation/CIM/5.2.0/User/Overview
9. https://www.elastic.co/security/siem

often one and the same, but again, sometimes not at all) other helping identify context but in an obfuscated way (e.g. ID, PID, PPID, GPID, GPPID...), and other being highly relevant only when shared between log sources. Here are some of the challenges coming from multidimensionality within logs:

— Data Volume: Multidimensional logs often result in larger volumes of data due to additional attributes, requiring resource-intensive management and analysis.
— Complex Analysis: Analysing multidimensional logs is complex, as security analysts need to consider multiple dimensions simultaneously to detect anomalies and identify patterns.
— Correlation and Visualization: Effective correlation and visualization tools and techniques are needed to handle multidimensional log data.

1.3.2.3 Completeness

Finally, sometimes log do not contain all the information. In a number of cases, enrichments must be performed so that the logs are usable directly during investigations. Sometimes that is because of storage issues; in order to reduce the volume of data recorded, some context can be overlooked. In many companies, storing logs comes with rules like saving the complete event during 30 days and when the deadline expires, only keep minimal information or nothing at all. Most of the time, logs are stored for fixed time periods. Some solutions have been put in place to save space and effort, such as aggregating and sampling. While those are smart methods to save space they are not helping the eventual analyst trying to understand the patterns of normal traffic during an investigation for example.

1.4 Investigations at Malizen

Malizen addresses cybersecurity data investigation with a collaborative log analysis and incident response tool for the analysis, understanding, and efficient sorting of all cybersecurity events. An analyst can explore logs through interactive and reactive data visualization. This platform notably aims to solve some challenges brought by logs described in the previous section. Malizen's goals are:

It is within this platform that we wish to contribute multiple recommender systems that help analyst make better decisions during their investigations. But first, let us describe the functionalities of the platform.

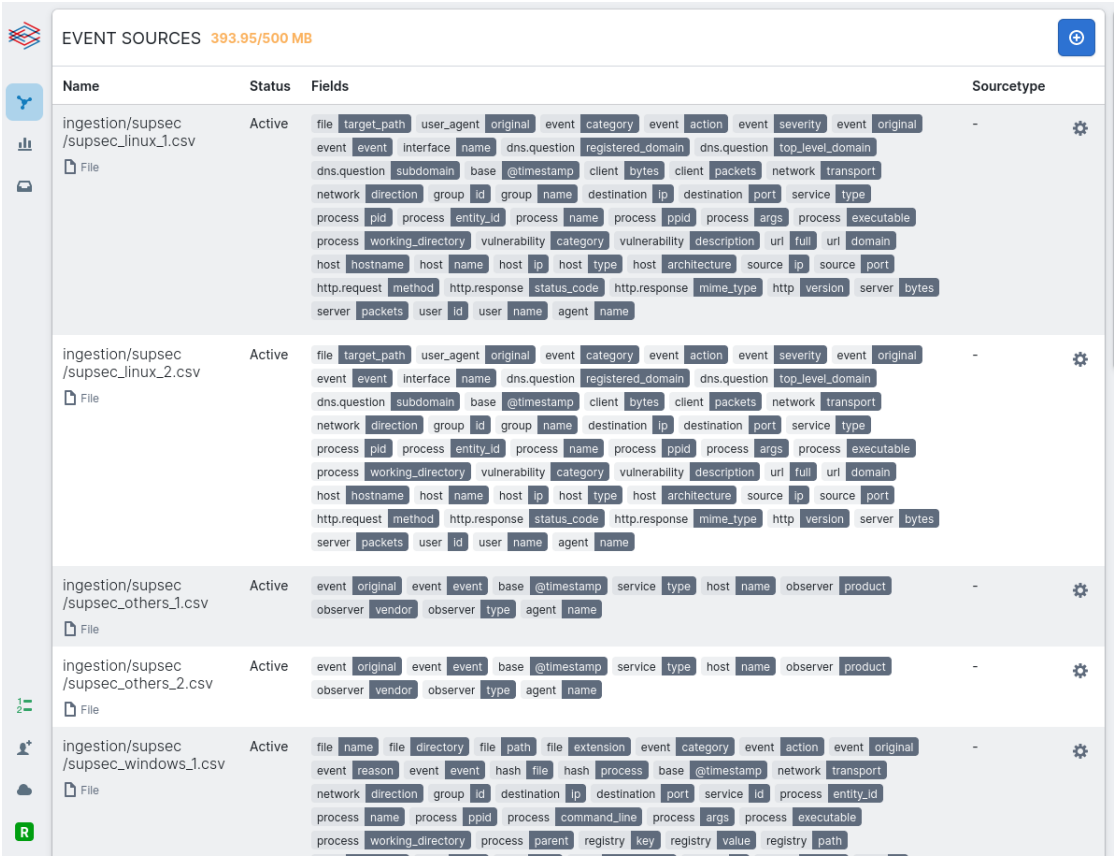

Figure 1.2 – A screenshot of the ingestion part of the log investigation platform

1.4.1 Ingestion

The Malizen platform allows an analyst to push logs from any relatively common 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) [10]. That way, during investigations, analysts can visualize the values contained in the logs using the ECS fields. ECS is an open standard for structuring and organizing data consistently within the context of log and event data in IT environments. ECS is designed to enhance the analysis, visualization, and correlation of log data from various sources by providing a common set of field names and definitions. This standardization simplifies data integration and interoperability between different tools and systems. By providing a shared vocabulary and structure, ECS streamlines data analysis, visualization, and correlation across different sources, enhancing operational insights and security investigations.

10. `https://www.elastic.co/guide/en/ecs/current/index.html`

Malizen already used ECS as a way of describing the data to investigate: keeping it as our way of describing logs was only natural. An interesting example is to take the *user* fields and see what they contain. The way ECS is built shows us that the logged information concerns a user, and after the dot is the nature of the recorded data: *user.name* is trivial, but then *user.id* informs rapidly as to how to categorize this identifier. All of ECS's fields are built like so and make it a very easy to use ontology.

Figure 1.2 shows the ingestion part in the platform. In the particular image we show an analyst that has ingested different files from different sources of data in order to investigate them later. These different files are all investigable together and the information they contain is all normalized according to the ECS format. Additionally, all log lines from all the ingested files are aggregated by fields (*e.g.*: file.target_path) in order to make the volume of data more bearable to explore later on. In this part of the platform an analyst can address some data related issues we have mentioned before, through different ways of processing his data before investigating it.

1.4.2 Analysis

The platform mainly provides a graphical interface to perform investigations as shown in Figure 1.3. 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 1.3. These visualizations present the aggregation of values for a 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 filtered by value, range of values, and time can be applied and will dynamically influence the other visualization windows. Part **B** of Figure 4.2 shows a timeline of the logs and allows analysts to focus on a particular time interval.

1.4.3 Reporting

Finally, as discussed in Section 1.2, it is essential that analysts have methods to report on their investigations. The platform provides a case management interface designed to do so, as well as a *flag* action in order to save interesting log values. Figure 1.4 shows this part of the platform. Even though it is out of the scope of our work it is important to show the effort made by the platform to reduce context switching for analysts. The platform provides analysts with a tool where they can conduct an investigation from start to finish:

from handling the data to investigate, to building a report on their findings. It backs the ideas we want to push forward for the whole cybersecurity community: understanding user needs and reducing their workload, so they can focus on their expertise as much as possible.

1.5 Conclusion

Now that we are familiar with the context in which cybersecurity analysts work in we can confidently say that while the cybersecurity field is improving day by day, some problematics still need to be addressed, regarding data and incident response mainly. Analysts are overloaded with difficult investigations. That can be explained by the complex handling of data, essential to start investigations, but also on the difficulty of actually investigating without the proper tools, whether in terms of knowledge or in understanding of their work. Our focus being the investigative part of incident response, we believe we can improve different aspects of investigations. We choose to focus on helping analysts be faster and more efficient during investigations. The new research question that emerges is: how can we understand analysts better in order to gain efficiency cybersecurity investigations and reduce the time needed to perform them. We will particularly focus on three issues identified within the field:

1. Can we improve the comprehension of an analyst during an investigation? (Section 1.1)
2. Can we benefit from expert knowledge during an investigation? (Section 1.2)
3. Can we obtain logs where attacks are identified in order to evaluate investigations? (Section 1.3)

In the following chapter we will discuss related work. In order to answer those questions, we oriented ourselves towards the field of recommendation. We will first study recommender systems, their use in cybersecurity, and how we can use them to specify and answer our research questions.

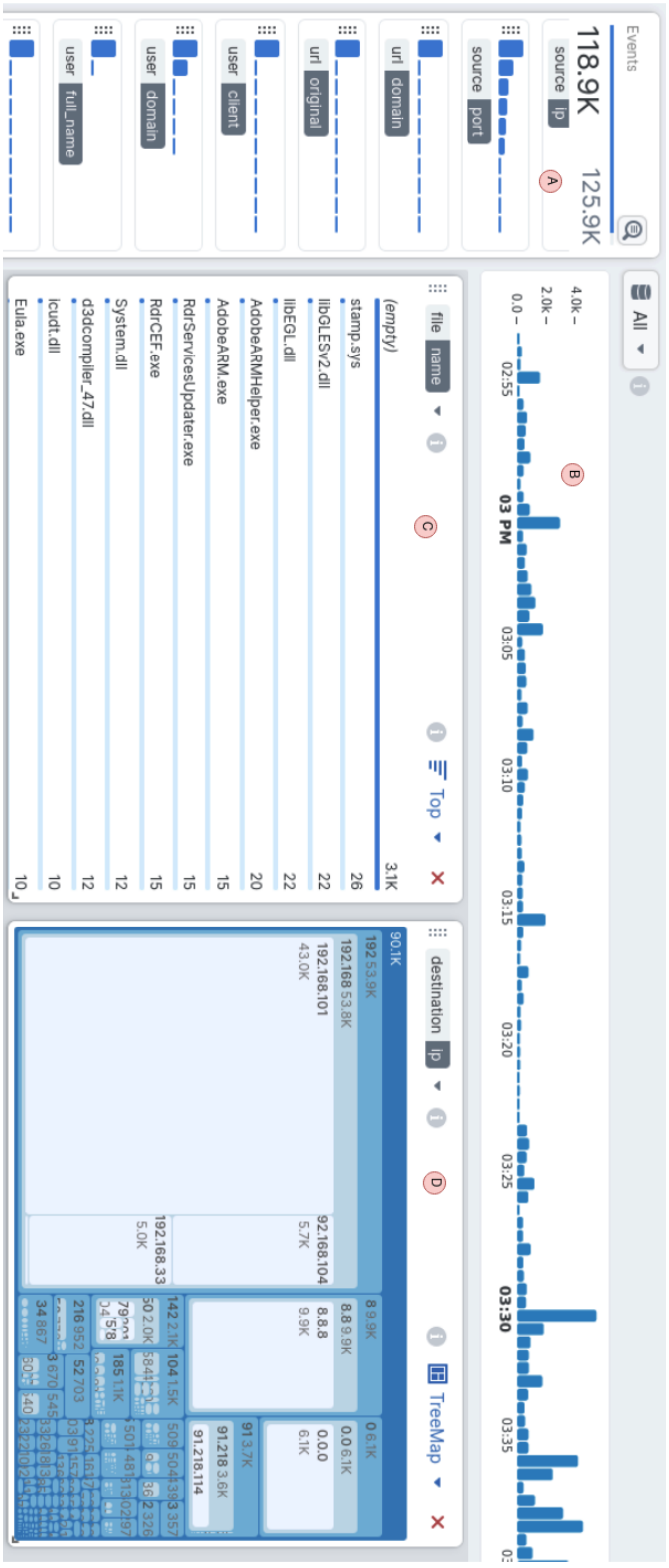

Figure 1.3 – A screenshot of the analysis part of the log investigation platform

Figure 1.4 – A screenshot of the case management part of the log investigation platform

Chapter 2

Related work

The archives are comprehensive and totally secure, my young Jedi. One thing you may be absolutely sure of: if an item does not appear in our records, it does not exist!

— Star Wars II: Attack of the clones, Jocasta Nu

I have introduced the concepts of investigations and analysts previously, to show in which context we will use recommender systems (RS). In this chapter I will try to give an overview of the nature and current use of recommender systems, specifically in the cybersecurity field.

Recommender systems have been widely used for the first time in e-commerce websites. The internet has seen the exponential growth of information available, the heterogeneity of objects and the difficulty of presenting relevant items to users happen in other areas, namely e-commerce. These platforms have begun using decision-helpers called recommender systems long ago and are very successful in doing so. We believe we can apply recommender systems to the challenges of cybersecurity.

This chapter will be structured around recommender systems and cybersecurity. After defining recommender systems I will present the different types they can take and the benefits and challenges that come with them. Afterwards I will focus on the cybersecurity uses of recommender systems in the literature. Finally, I will discuss the methods of evaluation of those recommender systems.

2.1 Recommender systems

Recommender systems are typically used when a user faces an overwhelming amount of possibilities to choose from and would benefit from suggestions of relevant items. As a consequence, recommender systems are widely used in fields such as e-commerce platforms or on-demand video content, since the number of possibilities is almost endless. Most recommender systems aim to provide relevant recommendations, but said relevance can

vary according to the situation. Recommendations need to fit the needs of users, such as discovering new items instead of similar ones, maintaining privacy, or even focusing on fast recommendations instead of accurate ones. The parameters and relevance criteria of a task to be solved by using a recommender system must be well framed before implementing the recommendation process.

A recommender system is a piece of software that aims to help a user making decisions regarding a task [69, 3]. Recommender Systems can be of different natures, but they all share some core features. First, a recommender system has some source of input data whose nature will often decide its type. Candidates for the recommendation are selected from this input data; a candidate is a selected item presented to the user. Then, all recommender systems also rank their candidates using various methods, such as similarity measures [29]. Triggering and presenting recommendations to the user are also ongoing issues in the literature. Some work presents recommenders as being explicit or implicit recommenders in the sense where the user has to ask for the recommendation himself. Presenting recommendations is however very rarely approached in the academic field, except when working in visualization, where it is very relevant.

2.1.1 Recommender system types

Historically, recommender systems mainly fit in one of three categories: collaborative-filtering, content-based and knowledge-based. These three categories are heavily reliant on the type of data used as input for the recommender system. These three types are described at length in the literature [40], but I summarize them in the following paragraphs. Recommendations techniques, benefits as well as disadvantages for each type described can be found in Tables 2.1 and 2.2, an extract of the one you can find in the work of Burke [15].

Technique	Input	Process
Collaborative-filtering	Ratings of items by users	Identify similar users and recommend well rated items
Content-based	Ratings & properties of items	Recommend similar and/or well-rated items
Knowledge-based	An expert description of user needs and interests	Recommend an inference between a user and a knowledge item

Table 2.1 – Recommender systems types

Technique	Benefits	Disadvantages
Collaborative-filtering	— Can identify cross-genre niches — Quality improves over time — Implicit feedback is sufficient — Domain knowledge is not needed	— Suffers from cold-start — Suffers from grey sheep problem — Quality dependent on dataset size — weak sensitivity to preference changes
Content-based	— Quality improves over time — Implicit feedback is sufficient — Domain knowledge is not needed	— Suffers from cold-start — Quality dependent on dataset size — weak sensitivity to preference changes
Knowledge-based	— No cold-start problem — Sensitive to preference changes — Can include non-product features — Can map user needs to products	— Does not learn — Expert knowledge required

Table 2.2 – Recommender systems benefits and disadvantages

2.1.1.1 Collaborative-filtering

The hypothesis of Collaborative-filtering recommender systems (CF) is that users often have common interests. By modelling a user profile, it is possible when trying to make a decision to find similar and/or dissimilar users in order to find out which items will be relevant to recommend.

Computing the similarity between users can be done using different properties. The main sources of user information are either given directly by the user himself, or by using the history of the user (*e.g.* E-commerce platforms use a user's purchase history to find out which items to recommend that he buys next).

Collaborative-filtering recommender systems are widely used and pretty refined nowadays; competitions on optimization of such recommender systems have been held [8]. They present interesting properties such as the capacity to identify cross-genre niches, meaning that by identifying similar users, they will be able to link some topics of interests other

recommender systems would not have been able to. Collaborative-filtering recommenders also have feedback loops, giving them the ability to improve the quality of recommendations over time. On the other hand the fact that no domain knowledge is needed for the recommender system to work implies that the recommender system is vulnerable to the cold-start problem [50, 6, 7], defined later in Section 2.1.2.

Collaborative-filtering recommender systems are often used jointly with content-based recommender system because their data inputs are similar, allowing for an easy hybridization.

2.1.1.2 Content-based

Content-based recommender systems (CB) allow users to deal with enormous amounts of information. They use item properties to filter all the available possibilities into more or less specific categories according to the user's need. The more properties are available for the items being recommended the more efficient will the content-based recommender be. However, to be able to make recommendations, content-based recommender systems also need user profiles filled with their preferences, in order to be able to match them to the most relevant items.

Content-based recommender systems also make use of similarity methods in order to match users and items. The challenges of this type of recommender comes more often from the capacity and efficiency to get the needed information as well as data scarcity. This particular problem is discussed in Section 2.1.2. The interesting property of content-based recommender systems is the fact that while they also suffer from the cold-start problem, new items are less impacted by it than new users like in collaborative-filtering recommender system. Indeed, since items come with content associated to them, we can achieve a satisfying level of accuracy without first training the recommender system, whereas the collaborative-filtering type does. However, where the collaborative-filtering recommender system did not need to know anything about the items to make recommendations, the content-based one will need content for every item in order to be able to make a single relevant recommendation.

2.1.1.3 Knowledge-based

Knowledge-based recommender systems (KB) have very different properties compared to the previous ones. The idea of a knowledge-based recommender system is to model the expert knowledge about the topic on which we wish to make recommendations, and use

it to select and rank candidates. This type of recommender system is able to map user queries and needs to the recommendable items. Additionally, since expert knowledge is used for the recommendations, the confidence in those recommendations is much higher and can be explained.

Knowledge-based recommender systems were first designed for cases where it was very unlikely that users could make relevant decisions by themselves, or in cases where other types of recommender systems could not be used. This type of recommender system is also sometimes associated with a recommender system named constraint-based. It is a subtype of knowledge-based recommender system where recommendations are based on predefined constraints or rules. In his work, Burke [17] shows how knowledge-based recommender systems can be tailored to user constraints and queries. For example, when a knowledge base is available, various methods to retrieve the information can be implemented and will yield different results. Indeed, since the knowledge is fixed, it is possible to very closely match the retrieval query with the recommendation query, instead of relying on similarity methods like in the previous recommendation methods. Burke notably shows how each method can be tweaked to maximize the accuracy and speed of recommendations.

The properties of the knowledge based recommender system make it insensitive to the cold-start problem, an important feature for industries (like cybersecurity) where errors may cost millions instead of in the e-commerce context, where recommending an item a user does not care for has little impact. However, the drawback of knowledge-based recommender systems is that the domain knowledge used to make recommendations must be maintained, a difficult task. Indeed, in cybersecurity, knowledge evolves rapidly, attacks are never reproduced twice in the exact same way and attackers find new ways of concealing their actions every day.

Fortunately, it is often possible to hybridize knowledge-based recommender systems to mitigate their drawbacks. Hybrid recommender systems are often possible and help compensate for many issues, while keeping the benefits of the more classical types of recommender.

Recommender classification problem

The scientific community is divided on the question of the classification of recommender systems. Most researchers agree recommender systems should be classified using the nature of their input data, but some other types appear in the literature, such as context-based [2], session-based [54], or the elusive demographic recommender system. All these types are either particular variations of the three categories described thereafter or not very represented in the literature. Most importantly, the different works of Robin Burke [17, 15, 16] offer, in my opinion, a clear view of the different types of recommender systems.

2.1.1.4 Hybrids

Hybrid recommender can refer to two systems: those described by Burke in his work [15, 16] and those not fitting in the previous category and mostly based on machine learning techniques (thereafter called new generation recommender systems). A hybrid recommender system combines two or more recommendation techniques in order to leverage the benefit of each technique and build an overall better recommender system.

Collaborative-filtering and content-based hybrids are the most common types of recommender systems hybridized. They somewhat compensate for each other's shortcomings, and both their inputs of data are often present; if there are users there are items and vice versa. However, both types suffer from the cold-start problem, which in some situations is not acceptable. For example situations where following a wrong recommendation could have a significant impact on the service in which the recommender system is integrated.

Table 2.3 shows commonly used hybridization methods. The weighted, switching and mixed methods are quite common even in other fields than recommender systems in order to gain some accuracy or broaden the scope of results. Some other methods like feature combination or meta-level hybridization are very specific to recommender systems, however they are often difficult to achieve [15]. Lastly, methods like cascade or feature-augmentation are used more and more because they offer more possibilities than the simple ones and are less complex to implement than the others.

It is to be noted than every classic type of recommender system cannot be hybridized

using every method; some combinations are impossible. For example, feature combination cannot be used between collaborative-filtering and knowledge-based or content-based and knowledge-based (in that order). A knowledge-based recommender system's knowledge base may take into account any kind of data, while a feature combination hybrid aims to reduce the sensitivity of the system to the number of ratings for example. If a knowledge-based recommender system considers the features of a content based recommender system has before him, the feature combination does not work. Some other combinations are also redundant, such as weighted hybridization between content-based and collaborative-filtering, since their recommendations, despite being from different recommendation types have the same benefits and disadvantages and so rely on the same assumptions most of the time. Although the redundancy can sometimes be useful when users or items are often incomplete, the recommendations can be refined using the other method.

Some hybridization methods have been favoured by researchers over the years. Among them cascade and feature hybridization are at the top [15, 16], because they have fewer constraints to be applied and because they offer better confidence in producing good results than most other methods. It is due to the fact that these methods of hybridization use can be computed separately and then hybridized without having to mix the different processes.

Hybridization method	Description
Weighted	The scores (or votes) of several recommendation techniques are combined to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

Table 2.3 – Hybridization methods from the work of Burke [15]

2.1.2 Recommender system challenges

Recommender systems help users make better and faster decisions, and they make relevant data more accessible. However, recommender systems also face significant challenges they need to overcome [27, 15]. Some of them have been mentioned in the previous sections, but I develop their associated challenges further in this section.

Table 2.2 mentions some of these issues as well. Some issues are more impacting than others and the most important among them often create the need for hybrids in order to compensate those weaknesses. In this section I will discuss the following:

— Data scarcity
— Privacy
— Scalability
— Cold-start
— Grey sheep problem

2.1.2.1 Data scarcity

In order to obtain good accuracy and stay relevant, recommender systems need data. Lack of information is fatal for recommender systems, especially for collaborative-filtering recommender systems and content-based recommender systems since they are heavily dependent on the quantity of information in order to obtain recommendations of good quality. Data scarcity refers to the challenge or limitation posed by an insufficient amount of relevant and diverse data for effectively training and improving the performance of the recommendation algorithms. However, data scarcity can also come from a lack of expertise used to maintain a knowledge-base in the case of a knowledge-based recommender system for example.

The data needed to make relevant recommendations is not any kind of data as well. Therefore, data scarcity is linked to the serendipity of recommendations. Serendipity corresponds to the apparition of good recommendations the user would not have thought of himself. These can only appear if enough data from various source is available to build recommendations. Indeed, trivial recommendations are rarely useful, especially in the case of recommender systems being applied to expert systems. Choosing the input of a recommender system is not an easy task. Knowledge-based recommender systems are particularly affected by that issue.

2.1.2.2 Privacy

Even if we manage to obtain compelling and comprehensive data to feed into a recommendation engine, user privacy can still pose a significant barrier. When it comes to collaborative-filtering recommender system, certain identifiable or nearly identifiable attributes, which could be valuable for recommendations, cannot be incorporated. In the realm of cybersecurity, most recommendation systems are purposefully developed to either avoid using sensitive information or to anonymize it before utilization. However, it's important to note that anonymized data, while privacy-preserving, can be less interesting and less effective for generating recommendations. Striking a balance between protecting user privacy and providing genuinely useful recommendations is a key challenge in the design of recommender systems.

2.1.2.3 Scalability

The developers of a recommender system should make sure it will scale up if needed. Nowadays, platforms such as Amazon or Netflix have thousands of new users and items every day, making it really hard for the recommender system to keep up with the load. Some work has started to tackle this issue by studying the difference between items and users instead of their similarity [18]. In general, many more differences can be found than similarities between users and items. By having more attributes to compare users and items, more links can be made in the recommender system. Consider a scenario where two users, User 1 and User 2, share a lack of interest in items categorized under 'A.' However, User 1 has a preference for items in category 'C,' while User 2 leans towards category 'B.' In this case, a recommendation system might suggest items from category 'C' to User 2 and items from category 'B' to User 1, using the premise that their lack of interest in 'A' makes them likely to like the same things.

2.1.2.4 Cold-start

The cold-start problem is a common challenge in recommender systems. This problem occurs when the system encounters a new user, a new item, or both, and it has insufficient historical data or information about these new entities to make accurate recommendations [50, 6, 7].

There are two types of cold-start problem:

— User cold-start: it occurs when a new user or a user that has not provided any preferences is in need of a recommendation.
— Item cold-start: this refers to the situation where a new item is introduced to the system, and there is limited, or no user interaction data associated with that item.

In both cases, the situation makes relevant and/or personalized recommendations challenging.

2.1.2.5 Grey sheep problem

Sometimes a user with particularly eclectic tastes or an item that belongs to too few or too many topics will generate a grey-sheep problem. It is when a user or an item is not similar to anything and so cannot be recommended accurately. Various works on clustering in recommender systems has well-identified the issue and tried to solve it using diverse methods [5].

It is to be noted that the grey sheep problem can be generated by a malicious user that wishes to harm the recommender's performances.

Discussion

In this chapter, we explore recommender systems as decision helper tools for analysts. Given our goal of aiding analysts in faster investigations without replacing their expertise, we believe recommender systems are a suitable method. We do not advocate for full task automation, as cybersecurity demands the nuanced judgment of analysts. However, when implementing recommender systems in cybersecurity, careful consideration is needed for the type of system, given data sensitivity. Additionally, challenges such as the cold-start problem should be addressed cautiously, as they can hinder the effectiveness of recommender systems in cybersecurity.

Now that I have properly defined recommender systems I will focus on recommender systems applied to cybersecurity for the rest of this chapter. Section 2.2 will detail how recommender system have been used to solve cybersecurity tasks.

2.2 Recommender systems in cybersecurity

The cybersecurity field faces issues similar to e-commerce, such as the amount of data and the need to browse rapidly and find relevant items. So, it has started making use of recommender systems as well. In this section, I detail various works done with recommender systems in the field [62, 38, 27]. Questions that will be addressed in this section are:

1. What data can be used to build recommendation engines in the cybersecurity field?
2. Which tasks are recommender systems accomplishing in cybersecurity?

I will also address how the different benefits and disadvantages of recommender systems and cybersecurity interact and complete each other. Table 2.4 describes the recommender systems used in cybersecurity and their characteristics. In the table, they are presented by some of their most defining attributes: the nature of their input data, the recommendation technique, the cybersecurity task addressed, and the evaluation type.

Recommender system	**Input data**	**RS type**	**Task solved**	**Evaluation**
Polatidis *et al.* [66]	System topology	CF	Attack prediction	Online
Polatidis *et al.* [65]	CVEs & CWEs	CF	Attack prediction	Offline
Soldo *et al.* [78]	Logs (malicious)	CF	Attack prediction	Offline
Franco *et al.* [31]	Services descriptions	CB	Protection service recommendation	Online
Esposte *et al.* [23]	Alerts & User preferences	CB / CF	Alert triage	Online
Sayan *et al.* [74]	Alerts	KB	Attack prediction	∅
Ayala *et al.* [6, 7]	OWASP, NIST... & User ratings	KB / CF	Attack mitigation	User study
Huff *et al.* [34]	CPE	CB	Vulnerability patching	Offline
Husak *et al.* [37]	Network monitoring data	CB	Potentially infected machines	∅

Karlsson *et al.* [42]	Vulnerabilities	KB / CB	vulnerability score	Offline
Sworna *et al.* [82]	IoC, Network monitoring data	CB	Cybersecurity tool decision	Offline
Nembhard *et al.* [56]	Insecure code, NVD, CWEs	Hybrid	Coding vulnerabilities fixes	Online
McDonnell *et al.* [54]	Malware datasets	Session - based	predict malware classification	Offline
Hussein *et al.* [1]	S&P attributes	Hybrid	Minimize cloud vulnerabilities	Online
Nunnally *et al.* [58]	Past interactions between user and visualization platform	Hybrid	Facilitate investigations	Online

Table 2.4 – Recommender systems in Cybersecurity

2.2.1 Recommender systems types in cybersecurity

Researchers have also been using cybersecurity data to build recommender systems for some time. In this section I will describe various recommender systems developed to complete cybersecurity tasks, organized according to the types described Section 2.2.

2.2.1.1 Collaborative-filtering

In cybersecurity, comparing users is often a sensitive task, so most work in the area makes use of variants of the classic collaborative-filtering method in order to avoid needing a lot of information about their users. Such methods are notably described in the work of Pawlicka *et al.* [62].

Polatidis *et al.* [66, 65] design a collaborative-filtering recommender system that uses data from CVEs and CWEs [1] to predict future attacks. Their goal is to be able to identify precise attack paths and classify them for later use in risk management. The recommender system first identifies attack paths inside an information system and then uses a parametrized version of multi-level collaborative-filtering from previous work [64] in order

1. https://cwe.mitre.org/

to match the resulting killchains with vulnerabilities. Vulnerabilities are first rated by users in terms of gravity, and the Pearson correlation coefficient between a user's rating and the average rating allows in a specific situation to predict the probability of an attack taking place. This permits for the risk-assessement of an infrastructure.

Soldo *et al.* [78] try to predict which IP addresses will be malicious in the future. It is a highly relevant issue to tackle in cybersecurity due to considerations like the attributions of public IP addresses or the fact that legitimate IP addresses can one day be used for malicious purposes. Their method is based on the use of past malicious logs that allow to model the behaviour of certain IP addresses. The prediction is made using a behaviour recognition model; some context, as well as victim information is also added to the prediction in order to compute its probability. The recommendation system compares past victims through the behaviour recognition model in order to find new potential victims in an already seen situation. This tool is used for attack prediction, but this time its recommendations could be used directly to update rules in a SIEM.

2.2.1.2 Content-based

Cybersecurity is also a field where the community shares a lot of knowledge. May it be from sources of intelligence like vulnerability databases, ontologies for cybersecurity data, or more rarely directly from information systems' topology, there is enough material to build content based recommender systems, if chosen carefully.

In their work, Franco *et al.* [31] propose MENTOR, a recommender system that recommends the right protection services to address a cybersecurity situation. The idea is to constitute a database of possible software to recommend to the user, that solves a particular request they have. The candidate selection is made using the requirements of the user and then the scoring is based on a similarity method called the squared pair-wise distance applied to each requirement. The particularity of this work is that the scoring is based not only on cybersecurity features but also considers the recommendation from an economical viewpoint. Unfortunately, no weighting of these completely different attributes is done, so, without a proper evaluation we cannot be sure that the recommender system is not biased.

A recommender with similar input data was designed by Huff *et al.* [34], but with a slightly different goal. Their content-based recommender system aims to discover which vulnerabilities a company should be concerned with. To do so, they use Common software Product Enumerators (CPEs) to obtain an item database and perform fuzzy matching

with vulnerabilities in order to recommend which vulnerabilities a software is likely affected by. Since software are not described in a rigid way, they use NLP and cosine similarity between software and vulnerability descriptions to build their recommendations.

Some other work decides to focus on using directly the topology of a network as input data for recommendation [37]. In this case the goal of the recommender is to help an administrator quickly discover which machines are likely infected after a malware attack. The recommendation system is based on similarity measures between candidate nodes and the nodes known to be infected. The higher the similarity is between two nodes with one being infected and the closer they are in the network, the more likely it is that the other one is infected.

More recently, Sworna *et al.* present the framework APIRO [82]. APIRO includes a content-based recommender system that uses network monitoring data along with Indicators of Compromission (IoCs) in order to recommend the right tool to its user during incident response. The interesting feature this tool contributes is a unified interface for security analysts to browse and select the adapted tools during security investigations.

2.2.1.3 Knowledge-based

Knowledge-based recommender systems are rarer in the cybersecurity field due to the difficulty of gathering true cybersecurity knowledge and maintain it sufficiently for it to stay relevant for extended periods of time. When knowledge-based recommender systems are used, they often come with a way to update the knowledge base.

Sayan *et al.* [74] propose a cybersecurity assistant in their work. In terms of recommendations, its goal is to detect attacks using network traffic data coming from monitoring tools in order to recommend defence solutions. The tool design considers the recommendation problem as a classification issue and uses an anomaly-based intrusion detection machine learning method. This data is then processed by applying feature engineering to it to transform it into a usable knowledge base. The resulting knowledge-based recommender system is an interesting approach, but probably very hard to maintain up-to-date since there is a lot of data pre-processing. It is however, not possible to know whether it works or not since no evaluation is provided.

2.2.1.4 Hybrids

The most famous type of recommender system in other fields [8] is a hybrid between collaborative filtering and content based types. However, user profiles and ratings make

less sense in the cybersecurity field. Esposte *et al.* [23] found a specific task where this kind of hybrid is relevant: alert triage. For their collaborative-filtering recommendation engine, they use analyst preferences and their previous ratings to match them to alerts and dispatch new incoming alerts to the right person. This engine is backed up by a content-based engine based on the properties of the alerts. The resulting recommender system is a mixed hybrid, following Table 2.3. This is promising work in solving the bottleneck of triage, however the evaluation of the recommender system is done by using machine learning methods, which does not give a clear view of the performance of the recommender system. For example, there is no feedback from users, so it is impossible to know if the recommendations were actually useful.

While the most widely used hybrids are between the collaborative filtering and content-based types, other types also exist. In cybersecurity, the cold-start problem (detailed in Section 2.1.2) is critical to overcome since there is no room for bad recommendations while the system ramps-up. Ayala *et al.* [6, 7] contribute an interesting hybrid recommender system using the feature augmentation technique between a knowledge-based and a collaborative-filtering recommender system. In doing so they benefit from the collaborative filtering method to leverage knowledge from their users, while also avoiding the cold-start problem by using a knowledge based recommender system as well. Their goal is to detect and recommend anomalies to their users, so they can prioritize the most important ones during their investigations.

The question of privacy is also an important one in cybersecurity; when the input data is domain-related knowledge, it is not always possible to use the data any way wanted. Karlsson *et al.* [42] design a recommender system that takes these problems into account. They implement a weighted hybrid recommender system between a knowledge-based and a content based recommender system, and they append to that recommender a feedback loop to maintain recommendations up to date. The recommendations are trying to score vulnerabilities using their properties and user feedback about them. They address the cold-start problem by using a knowledge-based recommender system and the privacy problem by not using a collaborative-filtering recommender system, but instead a feedback loop using only a user's own actions. This method is likely costly in terms of performance, but it ensures some data privacy.

2.2.1.5 Other types of recommenders

Recommender systems also do not always fit in a category. Some work such as the recommender system offered by Nembhard *et al.* [56] use methods such as text mining in order to find recommendation candidates within insecure code using vulnerability databases such as the National Vulnerability Database (NVD) or CWEs. A recommendation is then made by using a fixed pipeline of actions allowing the tool to find fixes for the detected vulnerabilities and propose them to the user. The fixes are ranked using a similarity scoring method among different possibilities. The fastest one to give a relevant result is selected.

Nembhard *et al.*'s recommender system and other like it are a little different because they do not fit in the categories I described. More often, instead of using classical recommendation methods, they will use machine learning. In this case the recommendation problem is approached as a classification problem. With this kind of approach, the recommendation system often benefits in terms of accuracy but loses a lot of transparency and explainability.

With the field of computer science constantly evolving, some entirely new types of recommender systems are even appearing. For example McDonnell *et al.* [54] offer a session-based recommender system. A session-based recommender system has the goal of predicting the next few user actions during short online sessions, and without any information about the past of the user. Their tool, CyberBERT has the goal of predicting the user's malware classification.

Another example is the work of Abuhussein *et al.* [1]. They develop a recommender system for cloud services named Cloud Services Security Recommender (CSSR). The idea of this recommender system is to identify the attack surface of a cloud environment and to recommend security and privacy attributes to minimize the cloud's vulnerabilities. The input data as well as the recommendations for this system come from a cyberattack taxonomy: AVOIDIT [77]. By using a taxonomy as the base for making recommendations, they effectively bring to light yet again a new type of recommender system. This recommender system has the particularity of also considering stakeholder requirements, as well as cybersecurity needs. Their qualitative evaluation with graduate students (specialized in the field) gives good feedback about their tool.

Finally, some work like NAVSEC [58] tries to solve a cybersecurity issue by using input data that is not directly cybersecurity data, namely the facilitation of cybersecurity investigations. In this case, they use sets of interactions an expert user has had with a

visualization platform that led to significant discoveries. A nearest-neighbour approach is applied to ongoing investigations and when the recommender finds a similar situation to one encountered in the past and deemed relevant, it is recommended. Moreover, Nunnally *et al.* add some important features to their tool. First, they enhance its flexibility because the system incorporates various machine learning techniques as modules, allowing developers to extend it with custom modules to meet specific requirements. Second, the tool adapts to different user communities, including those with limited resources or inexperienced personnel, making it user-oriented and accessible. Third, they stress the fact that their user interface was designed to be *unobtrusive*. This work addresses some of the challenges we wish to contribute on regarding easing investigations. However, their approach is from a visualization standpoint; we wish to provide a recommender system that helps analysts during investigations through cybersecurity expertise. Additionally, because of their use of the nearest neighbour approach, the core concept of NAVSEC is to mimic past situations and get as close to them as possible, not understand the current situation.

2.2.2 Constraints brought by using recommender systems in cyber

In Section 2.1.2 I discussed the hurdles faced by recommender systems. Some become essential to address in order for the recommender system to have value for their users.

Transparency and explainability become essential in cybersecurity. Analysts face vast amounts of false positives when sorting through alerts searching for real threats. When qualifying said threats they also need to report on them precisely and rapidly. That is why, when using a decision-helper, analysts need justification in order to work properly. Recommender systems using the more classical recommendation methods may it be in terms of ranking method or hybridization method can be explained more easily than the recent ones using machine learning for example. As a consequence, we decided not to use machine learning solutions.

Previously, I mentioned that considering the type of data recommender systems use as input, privacy could be an issue. In cybersecurity, this problem is amplified because not only the user's data is still sensitive, but item data and knowledge bases can also contain sensitive information. For example, a recommender system that would maintain a knowledge base with experiences encountered by experts would risk showing exactly what

threats the company faces by recording more details for certain attacks than for others. It would allow anyone that can use the recommender system to have a rough idea of the attack surface of this company. With this in mind we will avoid collaborative-filtering and content-based recommendation techniques, except if we can find safe input data.

2.2.3 Cyberattacks on recommender systems

Recommender systems are also vulnerable to cyberattacks. In particular, they are vulnerable to attacks that come through their input data, such as grey sheep, shilling or data poisoning. [27]

A shilling attack consists in the attacker feeding fake ratings to the recommender system in order to exploit the variation in item ranks induced. A poison attack is similar but has the goal of confusing the recommender system in order to make it give out false recommendations. [43]

Recommender systems are widely used in cybersecurity nowadays. From using the more classical types of recommender systems, research now tends to focus more and more on machine learning solutions either to feed or replace recommender systems. This is done in order to avoid certain issues inherent to recommender systems, but it also has the effect of losing transparency and explainability as well as trusting users with the final decisions more and more.

Discussion

Recommender systems are increasingly utilized in cybersecurity to address various challenges. Despite this trend, literature reviews, such as those in [38, 62], reveal ongoing gaps in the field. Projects like MITRE ATT&CK gain popularity, and we believe developing recommender systems based on expert knowledge can contribute to addressing issues like the cold-start problem and recognizing attacks during investigations. Notably, existing approaches often focus on enhancing cybersecurity tasks for analysts without placing analysts themselves at the core of recommender systems. These perspectives will be further explored in the second part of this manuscript, aligning with the research questions outlined in Section 2.5. However, to advance our understanding, proper evaluation methods for recommender systems must be established, a topic we aim to tackle.

2.3 Evaluating recommender systems

Recommender systems experts rarely agree about how to evaluate the entirety of a recommender system. Some people even argue that recommender systems have too many goals and uses to be evaluated in a standardized way. Historically, most people evaluated recommender systems using metrics such as accuracy, prediction error, etc. [40]

During this thesis I have noticed that the evaluation of recommender systems is often disregarded or incomplete. In this section I wish to discuss what efforts have been made in the community to provide recommender systems with, standard methods of evaluation. Such ways of evaluating are essential when comparing different work, may it be in terms of relevance, performance, or even, user satisfaction. Something else that is rarely present in research papers is the visual shape of recommendations. A recommendation can only be relevant if it is presented to the user at the right time and under the right shape. Most of the literature about recommender systems does not give all of this information easily.

2.3.1 Evaluation approaches for recommender systems

The evaluation of recommender systems is an ongoing topic as there are multiple facets of recommender systems that can be evaluated [68]. Studying recommender systems, I came across three main methods of evaluation. I will name those methods according to the work of Karlsson *et al.* [42].

First, the original way of evaluating recommender systems was the offline method. It consists in using a set of historical data in order to produce various metrics about the evaluated recommender system. It has the benefit of making a recommender system easily comparable to others that have been evaluated using the same method. This method is also applicable without requiring user interactions with the system. However, since the nature of recommender systems is to be decision helpers for users, not having said users participate in the evaluation makes it seem like some part of the recommender system is left untested.

User studies are a way to fill that gap. In user studies, feedback is collected from users before, during and after the testing of the evaluated recommender system. The interesting part about said studies is the fact that they highlight the differences in use and in efficiency for users induced directly by the recommender system since it is the only variable. The issue with user studies lies in the difficulty of gathering a statistically relevant number of experts of the field in which the recommender system operates.

The last method is the online study. Online studies consist in collecting data from a running recommender system at the disposal of participants. The data can then be leveraged in different ways. A common use is a qualitative feedback survey after a participant is finished using the recommender system. One constraint is to minimize the number of participants needed while maximizing the diversity of the data collected. Some online studies even do not even need users. Online studies also often try to gather better fitting data about recommender systems instead of general metrics, for example:

— Number of followed recommendations compared to number of given recommendations;
— Relevance of recommendations both in qualitative and quantitative terms;
— Estimation of the efficiency gained with the use of the recommender system.

2.3.2 Evaluation of recommender systems in cybersecurity

Table 2.4 shows what type of evaluation is performed for each recommender system used in cybersecurity and cited in Section 2.2.1.

The first interesting thing to be noted is that there are some works where evaluation is absent altogether [74, 37]. This can sometimes be due to the fact that the work only presents the concept of a recommender system, but more often, the researchers have not found a good way of evaluating. This could be improved by a standard method of evaluation for recommender systems.

Next, among all the different works presented, only one of them is evaluated with a user study [6, 7]. This is mainly due to the fact that it is extremely difficult to gather a relevant sample of cybersecurity experts in order to test out recommender systems. Most of the other evaluations that make use of participants consists in an online study and shows that often, participants are other researchers and/or students in the field.

Then, most of the evaluation methods employed are either offline or online evaluations (see Table 2.4). They are easier to organize, the results are simpler to present and speak to the scientific community in ways where even non field experts can gauge the quality of results. However, the consequence is less interest given to the human part of recommender systems in the literature.

To evaluate properly a recommender system it is necessary to know, the type, the input data and the task to solve. Once it is defined it is much easier to find out what should be measured, how to select your participants, and what feedback is needed from them. In the next part of this manuscript we will detail the evaluations that come with our contributions, where we focus on evaluating multiple aspects of our works:

— The data used (volume and quality among others.),
— Quantitative feedback (*i.e.* metrics, performance indicators etc.),
— Qualitative feedback (*i.e.* user surveys).

2.4 Recommender systems and visualization

To help analysts during security investigations, many visualization tools have been proposed by the research community to analyse various event data such as network logs [85, 20], DNS logs [71], system logs [36, 35] or file system metadata [9]. These methods allow a faster and easier investigation giving the analyst the possibility to query and visualize large amount of complex data. They however require significant expertise in both security

and visualization techniques. Recommender systems are starting to be used to tackle that issue.

2.4.1 Visualization

Data visualization is the graphical representation of information and data. It uses visual elements such as charts, graphs, and maps to help people understand the significance of data by presenting it in a visual context [89]. Building on statistics, these tools provide an accessible way to see and understand trends, outliers, and patterns in data. A single dataset can be represented in multiple different ways, focusing on subsets, transforming data and through different visual configurations. Each approach provides a different point of view and allows different insights about the information. In cybersecurity, data visualization provides an easier way to view data from different sources. Using charts, graphs, and real-time dashboards, IT teams and security professionals can reduce clutter and make it simple to spot anomalies that may indicate threat activity.

Visualization tools [30, 83, 28, 20, 85, 9] have been developed to identify attacks in the data. Among those tools, some have focused on log visualization [36, 35]. Because of the complexity of monitored systems, the quantity of events logged and their complexity, the necessary time to investigate is too long [21]. Visualization systems also require extensive field knowledge to be used efficiently [10]; as a consequence, users have to learn their usage. Recommender systems are very often integrated to a visualization interface once they go beyond the step of proof of concept. It seemed natural that we worked on the visualization part as well.

2.4.2 Visualization recommendation

Recommender systems have been proposed as a complementary approach to visualization tools to address these issues. They are mostly designed to help the user choose a better representation of the data [53, 86, 91, 90, 33]. These approaches are not specific to security investigations and require advanced knowledge and practice in terms of visualization. However, these solutions have their shortcomings such as the lack of reliable data to make recommendations, the heterogeneity and quantity of data to explore.

Outside cybersecurity, previous work combines recommendations with visualization [91, 90, 22, 86, 33]. When a recommender system is used with a visualization system, the recommendations are mainly used to offer to the analyst the more useful representations.

This can be seen as an extension of work about automatic representation [52, 53]. The recommendations can be computed using statistical and perceptual measures [91, 90] or using machine learning [33]. As a consequence multiple visualization options are offered to a user, which has to decide which one is the best suitable to her needs. We believe that the required level of expertise is too high to be usable in real use cases.

2.4.3 Visualization in cybersecurity

Visualization recommendation has also been studied for security purposes. Only few works are related to the visualization of security data enhanced by a recommender system [58, 92]. Some work such as the research of Zhong et al. [92] present their work on a visual interface. Their primary objective is to develop an efficient and effective system for data triage and retrieval in the field of cybersecurity. The authors address the growing challenges of managing and extracting meaningful insights from vast amounts of security-related data. They focus only slightly on the visualization part when tackling the question of the user interface associated with the tool. The authors emphasize the importance of user-friendliness. The system's interface is designed to be intuitive and user-friendly, facilitating ease of use for security professionals of varying technical backgrounds. However, all of these work make use of complex visualization techniques such as Sankey and Chord diagrams [49] or even spatial visualizations [58]. This forces the users, most likely cybersecurity experts, to also be visualization experts in order to be able to interpret and work with these visualizations properly.

Li et al. focus on security risk analysis and offer defensive measures recommendations [49]. They employ visualization methods to enhance the understanding and interpretation of data protection recommendations and planning support. The primary focus of this study is to develop a system that aids security analysts in identifying which data assets require priority protection measures.

The closest work to ours is NAVSEC [58], a recommender system integrated with a 3D visualization tool [59] for network data. During the investigation, NAVSEC will regularly offer to the security analyst a set of interactions with the 3D visualization tool to discover a possible intrusion. The best interactions are selected by a nearest-neighbor approach based on a database of previous investigations conducted by an expert security analyst. This input data for the recommender system could also inadvertently leak some data about the company's security if not handled properly. NAVSEC is a collaborative recommender system; it does not consider the user's need or query in the recommendations

and does not benefit from the accumulated knowledge on attacks, but only work using the visualization information.

All of these works present the same shortcomings in terms of recommendation: the cold start problem [61], unacceptable in cybersecurity. For example, the nearest-neighbor approach presented in NAVSEC relies on data from past investigations, making the system inaccurate and unreliable if no previous investigation's data is available when trying to make recommendations. After studying the possibility of designing a recommender system that would help choose visualizations we decided against for the various reasons explained. We also put pure visualization aspects out of our scope for this thesis. Nonetheless, learning about the topic has significantly helped us in framing our research.

2.5 Chapter conclusion

In this chapter we have built upon our hypotheses from the introductory and background chapters and presented in more detail recommender systems and their different facets. We have then studied the literature in order to classify recommender systems into types and understand how they work, first in general then in the context of cybersecurity. We have also introduced two topics, that are often considered when discussing recommender systems: the question of evaluating recommender systems and their closeness to the visualization field. Some issues and challenges emerging from data and recommender system have been partly addressed, but we see multiple areas still lacking. We updated our research questions in agreement with our observations to formulate three new ones:

R1: Can we benefit from expert knowledge during an investigation by using knowledge-based recommender systems?

R2: Can we improve the comprehension of an analyst during an investigation by understanding their intentions and using them to make recommendations?

R3: Can we obtain logs where attacks are identified in order to evaluate the use of recommender systems during investigations?

Our discussions about recommender systems reached a point where we are able to make our own contributions to the field. We are also able to evaluate our future recommender systems since we have studied the strong and weak points of methods of evaluation in the literature. The next part of this manuscript is dedicated to the contributions we made during this thesis, addressing our three research questions.

Part II

Contributions

Chapter 3

Helping users find the right exploration paths using expert knowledge

Overconfidence is a slow and insidious killer.

— **Darkest Dungeon, The Ancestor**

KRAKEN

This chapter is about combining expert knowledge coming from the cybersecurity community and visualized logs. Our hypothesis is that during computer security investigations, analysts are the most reliable source of knowledge and, our role is to provide them with the right tools. Our contribution aims to put relevant recommendations at the disposal of analysts during their investigations. This recommendation system links knowledge coming from advanced attack descriptions to a visual analysis tool in order to suggest exploration paths: a Knowledge-based Recommender system for Analysts to Kick Exploration up a Notch (KRAKEN). To evaluate KRAKEN we conducted a user study with seven security analysts. Using our system, they investigated a dataset from the DARPA containing different Advanced Persistent Threat attacks. The results and comments of the security analysts show the usability and usefulness of the recommender system.

3.1 Introduction

During our study of the literature, we have identified various areas where we could contribute a recommender system. In this work we first address the idea of making better use of expert cybersecurity knowledge during investigations (**R1**). The cybersecurity landscape now includes multiple projects that gather intelligence and allow experts to have relevant and complete knowledge about most attacks. However, we have not found in the literature any recommender system that leverages this knowledge as an input for recommendations. MITRE ATT&CK is one of those projects; it is often used during investigations but rarely directly integrated to them. We feel it would be a shame not to benefit from it during an investigation.

Our focus in this chapter will be to answer two questions using a recommender system:

— Can we use a cybersecurity knowledge base in order to make relevant recommendations?

— Can abstract knowledge be linked to logs in order to make it usable during investigations?

We will try to answer those questions without using visualization within the recommender system because we believe we can better help analysts by helping them leverage knowledge about which they have an expertise already. Then, our second research question implies the need to link an abstract knowledge base to real attack traces left by an attacker, which will present a significant challenge. Finally, we believe cybersecurity gains a lot by being transparent and explicable so, we decided not to use any machine learning solutions whatsoever, as they are often less easily explainable.

We intend that our recommender system helps the analyst by suggesting exploration options, either to test a hypothesis on the incident, or to analyse another part of the logs that has not been explored yet. It will exploit a knowledge base of adversary tactics and techniques extracted from real-world observations: the MITRE ATT&CK matrix [1] (see Section 1.2.2.1).

The rest of the chapter is organized as follows. First, we discuss the design of KRAKEN, a knowledge-based recommender system in Section 3.2. Section 3.3 details our evaluation of the recommender system. This evaluation was conducted on the first version of KRAKEN. Towards the end of the chapter we find Section 3.4 where we detail improvements we have made to KRAKEN after receiving various feedback during the evaluation.

1. `https://attack.mitre.org/`

Eventually, we conclude on this work in Section 3.5.

3.2 The KRAKEN recommender system

We built KRAKEN, a knowledge-base recommender system [17] that avoids the cold start problem and makes transparent and explainable recommendations, two essential properties in cybersecurity. In cybersecurity, we believe in avoiding having a ramp-up of recommendations because it could lead to errors early-on. We also need to be able to justify our recommendations because as described in Section 1.2.3, reporting on investigations is also an important part of the process. Then we want to leverage expert knowledge to make recommendations, because we believe that carefully chosen expert knowledge is safer, and leads to better recommendations. Our end goal is to see analysts discover more attacks during investigations. Our evaluation will show that depending on the experience of users, KRAKEN achieves its goals.

3.2.1 Overview of KRAKEN

Figure 3.1 shows the overview of KRAKEN. We first describe **step 1** from the figure. After converting the logs to the ECS format and ingesting them in the platform, an analyst starts investigating. The recommendation system is triggered when an analyst flags a field's value (see Section 1.4), declaring that it is relevant to his investigation.

In order to be able to make recommendations, KRAKEN will query knowledge from a knowledge base described in Section 3.2.3. Depending on the recommendation context and the decision-making process used (two possibilities), the knowledge queried will vary. This knowledge is used to select the candidates to be ranked in order to create a recommendation for the user. This is **step 2**.

Regarding decision-making in **step 3**, there are two possible situations. In both cases, the end recommendation will be possible fields to explore, but they will be selected differently. Either we consider that the state of the investigation is known by the analyst and that his goal is clear, or the analyst is unsure of the attack he has found and requires the recommendation to also help him understand the situation. The full process is described later in Section 3.2.4.

Finally, this process generate fields potentially relevant to explore as recommendations (**step 4**). More precisely, the three best ranked candidates are displayed to the analyst.

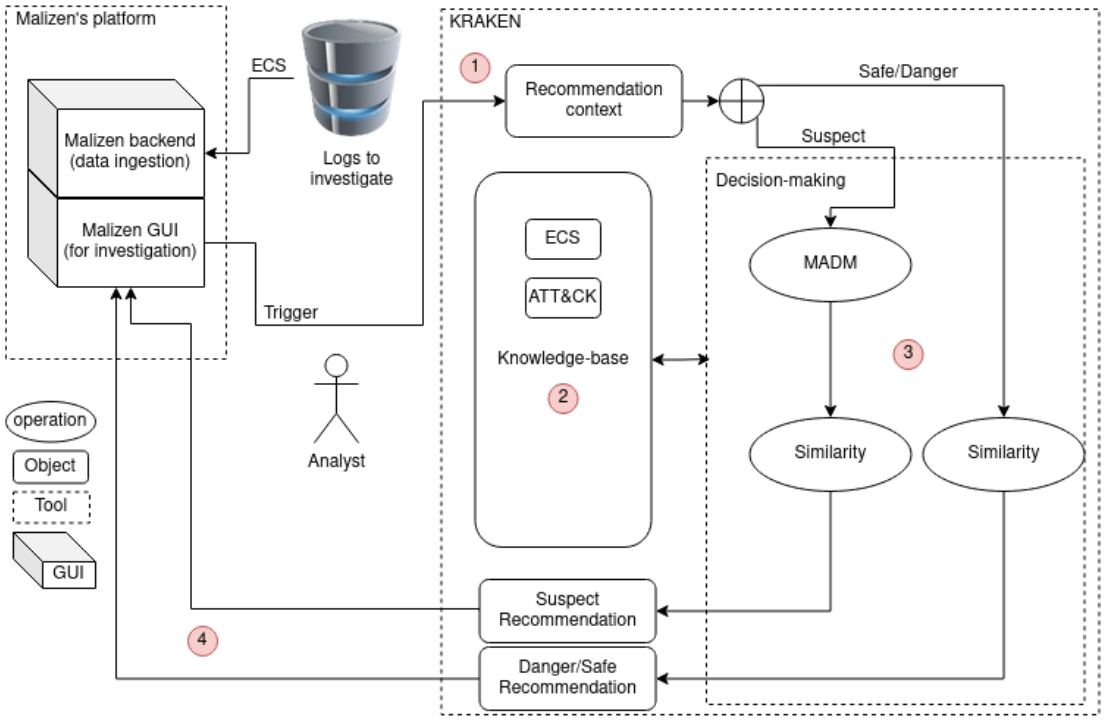

Figure 3.1 – The recommendation process of KRAKEN

Each one of them can be followed individually. Additionally, some essential features have been added to the process. The most important one is the explanation of recommendations. Should the analyst ask for it, some information about the data that motivated the recommendation is available.

The following sections go into further detail regarding the different steps of our recommendation technique. We will start by explaining what we associate with context for the recommendation. Then, we will describe the knowledge-base, followed by the two scoring methods implemented for KRAKEN.

3.2.2 Context associated with a recommendation trigger

We call the "context" some important information associated to the flag action. At the time of a flag, an analyst associates it with a field as well as the severity associated with it. This context will influence the recommendation.

This context notably contains information about log sources and severity. Log sources correspond to the data sources where a field can originate from. For example an IP address

will be found in a source such as firewall logs but probably not in system process logs. The most important piece of context is the severity associated with a flag. This severity corresponds to the relevance of the value to the investigation; if it is not associated to an attack, possibly or partly or if it identifies an attack. It is recorded and given as context to KRAKEN when a recommendation is triggered, along with other information.

$$Context = (field, severity, datasources)$$

With $severity \in \{suspect, danger, safe\}$ and data sources the sources of log where the field is observable.

All this information is given to the recommender system to generate a recommendation. The severity is studied in order to select how the recommendation should proceed:

1. The analyst has flagged with the **suspect** severity: the analyst needs more information before deciding whether the value is linked to a malicious activity or not.
2. The analyst has flagged with the **danger** severity: a threat artefact has been found and, as such, ends this part of the investigation. The analyst wants to direct his attention somewhere else.
3. The analyst has flagged with the **safe** severity: a threat has not been found yet, or that there is none. The analyst wants to take a look at the situation from another angle.

3.2.3 Structuring ECS and ATT&CK into a knowledge base

As previously described, in Section 1.2.2.1, the ATT&CK project offers us the necessary cybersecurity knowledge for our recommendation goals. The project regroups as exhaustively as possible all possible attacker behaviours, their properties and which log sources to use to observe them. On the other side, ECS (see Section 1.4.1) allows us to take logs from any source and represent them using their ontology. In order to be able to link tangible fields from an investigation to cybersecurity knowledge, we linked these two projects.

3.2.3.1 Building the knowledge base

We felt ATT&CK was the right choice to use as a knowledge base for this contribution for multiple reasons. First, the project is continuously maintained and, the knowledge is often updated. Second, the way it is built makes it resistant to tool changes from attackers. And finally, the knowledge described covers every area of cybersecurity and every step of a possible attack, making it one of the most exhaustive projects.

We mapped the data sources associated with every ATT&CK technique to a set of fields defined by ECS, thus building a knowledge base. For us this meant carefully reviewing every possible source of logs, and finding all the potential data types that could be observed through them using logs at our disposal (*e.g.*: IP addresses can be observed in network sources of logs but also in some types of system processes.). The knowledge base we obtain enables us to quickly compile a list of fields that are valuable for observing attacks in logs. Conversely, it also allows us to identify the attacks that can be observed through a specific field.

Figure 3.2 shows an extract of the knowledge base. We can see how tactics and techniques from ATT&CK were linked through their data sources to ECS fields (see dashed arrows on the figure). We bring to your attention that every edge of the graph can be browsed in any direction, allowing ATT&CK knowledge to be queried from ECS and inversely.

This knowledge base is accessed using GraphQL [2] as it allows us to have as much liberty when we need to describe and query our very heterogeneous objects, but also because it renders querying knowledge is possible in any way, from any node in the graph. The recommender system is being federated through GraphQL to others services of the Malizen platform. The result is a deeply integrated RS with the flexibility to change and iterate rapidly during the experiment phase

3.2.3.2 Recognizing an attack using the knowledge base: an example

To illustrate the use of the knowledge base as a way of understanding how to observe an attack, we show in Figure 3.2 an extract of the knowledge base that corresponds to a use case. During an investigation, an analyst is exploring the logs from a router. He observes many connections to the TCP port 6667. That port is related to IRC, which is not a commonly used protocol in an enterprise environment. Those connections are thus

2. https://graphql.org/

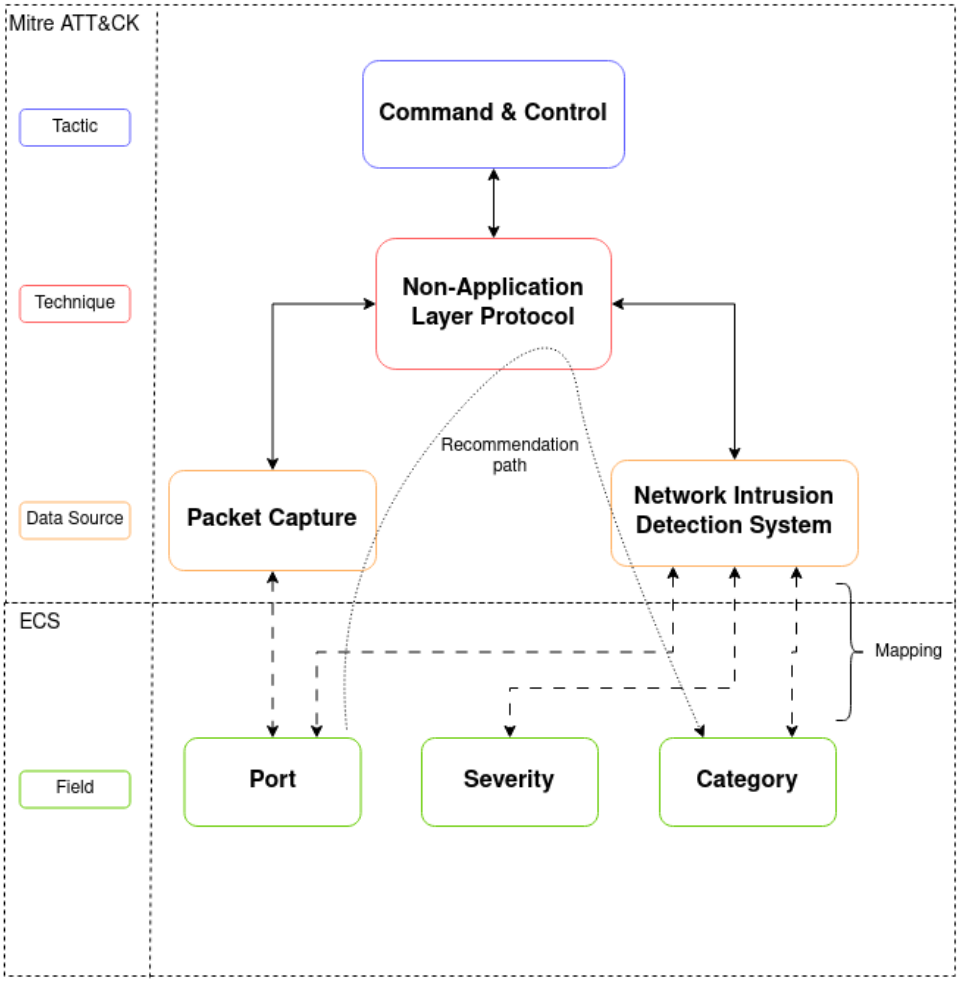

Figure 3.2 – Extract from the knowledge base

suspicious and the analyst flags this value as **suspect**. He needs now to confirm that the activity related to this destination port is linked to an attack or not. At that point of the investigation, there are still thousands of events related to that destination port and the whole process can take hours.

Here is how our knowledge base would select knowledge to observe this attack easily: Figure 3.2 shows that the port field that was studied is associated with the "Packet Capture" source, in turn associated with the technique "non-application layer protocol". From there, we can select all the data sources linked to this technique, *i.e.* "Network Intrusion Detection System" in the example, and all fields linked to them: "Severity" and "Category". In this instance, pivoting on the technique is what we call a *recommendation path*: the analyst was analysing suspect traffic coming from IRC, and upon flagging the

Algorithm 1 Our recommendation algorithm

procedure RECOMMENDATION(dt_0, severity) ▷ With dt_0: flagged field
  ▷ dt_x = field
  ▷ Dt_x = vector of fields
  ▷ Ds_x = vector of log sources
  ▷ T_x = vector of techniques
  $Ds_0 \leftarrow AllDataSourcesLinkedTo(dt_0)$
  $T_{all} \leftarrow AllTechniquesLinkedTo(Ds_0)$
  if severity is *suspect* **then**
    for all techniques in T_{all} **do**
      $T_{scored} \leftarrow MADM(T_{all})$ ▷ We score each technique using MADM
    $T_{best} \leftarrow T_{scored}[0:2]$ ▷ The three best scored techniques
    $Ds_0 \leftarrow Ds_0 + AllDataSourcesLinkedTo(T_{best})$
  $Dt_{all} \leftarrow AllDataTypesLinkedTo(Ds_0)$
  $Dt_{filtered} \leftarrow FilterIrrelevantDataTypes(Dt_{all})$
  for all fields in $Dt_{filtered}$ **do**
    $Dt_{scored} \leftarrow Similarity(Dt_{filtered})$ ▷ We score each field using Similarity
  $Dt_{best} \leftarrow Dt_{scored}[0:2]$ ▷ The three best scored fields
  return Dt_{best}

port as suspect, he is given the recommendation containing the "Severity" and "Category" fields. By following this path within his investigation, the analyst finds alerts raised by the NIDS about attempted information leaks and corporate policy violations.

However, Figure 3.2 only shows an extract of the knowledge base, meaning that if we were to try this in an actual investigation, the number of techniques resulting from our request of all techniques observable in the log source "Packet Capture" will be much higher, as well as the number of fields present in "Network Intrusion Detection System" logs. This means we have to find a way to select the most relevant techniques and fields at these steps.

3.2.4 Decision-making in KRAKEN

To compute the recommendations, we implemented a decision-making algorithm that uses the knowledge base presented in the previous section and the severity of the flag. In the case of a safe or danger flag, we only used a Similarity scoring method, whereas for a suspect flag we implemented Multi-Attribute Decision-Making (MADM), on top of the Similarity scoring.

Algorithm 1 shows how recommendations are generated. First, using the information given by the field flagged by the analyst (dt_0), lines 6 and 7 show that we select all sources of logs where that field can be recorded and then all ATT&CK techniques that can be observed by an analyst exploring the selected log sources. Then, as explained in Section 3.2.2, if the severity associated with the flagged field is suspect (line 8) then we will first use MADM to select the most likely observed technique (line 10). Once the most relevant techniques are chosen, we select all possible data sources where this technique can be observed in line 12 and select all fields present in the investigation linked to these data sources (line 13). This part of the algorithm is what we called *recommendation path* in the previous example. We rank them using our similarity method (line 16), and provide a recommendation. If the severity associated with the recommendation is not *suspect* (line 8), we will skip the MADM scoring and go directly to line 15 for Similarity scoring. For techniques as well as fields, once scored, we select the three best scored and use them as input for the next step of recommendation. Both the MADM and Similarity methods are developped in the sections hereafter. We can see how the knowledge base is used in this algorithm by following the dotted line titled *recommendation path* from Figure 3.2. The MADM scoring is used when we reach the techniques and the Similarity scoring when we reach the fields.

All functions that appear in this algorithm represent queries to the knowledge base, and they request all possible objects linked to its argument. For example: $AllDataTypesLinkedTo()$ will return all ECS fields linked to a specific log source or technique if the given argument is a log source or a technique respectively. Fields categorized as irrelevant in function $FilterIrrelevantDataTypes()$ are those that are not present in the investigation, those who only have one value through the dataset and the flagged field itself that we deem would make irrelevant recommendations and so, are filtered out.

Techniques can't be ordered totally, the number and the type of their attributes making it impossible. MADM uses partial orders to compute a weight for each attribute giving the ability to score techniques. ECS fields are scored using our Similarity scoring method. Simple similarity scoring methods focus on assessing similarity or distance between alternatives based on a limited set of attributes and are more appropriate for less complex decision contexts, and such is the case for fields.

3.2.4.1 Scoring techniques using multi-attribute decision-making

Multi-attribute decision-making (MADM) is a systematic approach used to evaluate and compare multiple alternatives based on multiple *criteria* or attributes. It involves assessing and ranking alternatives in order to make informed decisions when faced with complex and multifaceted choices. MADM methods aim to provide a structured framework for considering the diverse factors that contribute to a decision and determining the most suitable course of action. In this case, we use MADM in order to be able to rank ATT&CK techniques (line 10 of Algorithm 1). Due to their complex attributes, technique objects from ATT&CK are difficult to rank. We choose to implement a specific method of MADM: an additive Analytical Hierarchy Process (AHP) [39]. The Analytic Hierarchy process is a sophisticated decision-making method that takes into account the hierarchical structure of criteria, pairwise comparisons of preferences, and the synthesis of these preferences to arrive at an informed decision. The exact method we use is called Simple Additive Weighting (SAW) [4].

We use the attributes associated to a technique as a list of *criteria*. Some criteria are non-numerical, so we converted them manually into numerical values in order to be able to apply MADM (*e.g.* Permissions are each given a numerical value, the higher the privilege level the higher the value.). The following list shows the attributes of a technique, used as criteria in the MADM:

— Platforms (vector of names): on which the technique can be executed
— Permissions (vector of roles): needed to perform the technique
— Network requirements (numerical): needed to be able to do the exploit
— Frequency (numerical): at which this technique has been seen in real life scenarios
— Mitigation (numerical): a number of ways this technique can be mitigated.

The process is divided into two phases: the creation of a consistent Pairwise Comparison Matrix (PCM) [45] and the computation of candidate scores, which is executed each time a recommendation is needed. The goal of a PCM is to systematically evaluate and compare the relative importance or preference of different criteria within a set. To ensure the reliability of the comparisons, the matrix can be checked for consistency. Inconsistent judgments may lead to unreliable results. Various methods, such as the eigenvalue method used here, can be applied to assess consistency. After verifying that it is indeed consistent, this matrix is used to compute an overall weight for each *criterion*. From there we can score candidate techniques. Each of these steps are described in more detail hereafter.

Technique attributes	Platforms	Permissions	Network Requirements	Frequency
Platforms	1	1	0.25	0.5
Permissions	1	1	0.33	0.5
Network Requirements	**4**	3	1	3
Frequency	2	2	0.33	1

Table 3.1 – The best PCM for KRAKEN with a 0.015 consistency. *e.g.* In this PCM the bold value 4: on a scale of 1 to 5, the Network requirements for a technique are a lot more important (4 out of 5) than the platforms on which a technique can be executed.

Designing the Pairwise Comparison Matrix While PCM are an effective and widely used solution, they have to respect a rule of consistency, depending on the scale used: ratio scales, geometric scales and logarithmic scale. The simplicity of the ratio scales method presented by Saaty *et al.* [73] makes the ratio scale a good fit for our decision-making process. Two guidelines are proposed by Saaty *et al.* to build a PCM with a high level of consistency:

— Using an adapted scale for the evaluation of the relative importance (in our case, 1-5) depending on the number of *criteria* to clearly differentiate answers. These values are defined and calculated by Saaty [73].

— Keeping pairwise consistency, meaning the comparison c of criteria A and B should respect: $c_{AB} = 1/c_{BA}$ is a necessary but not sufficient condition. Although Saaty *et al.* specify that "improving consistency does not mean getting an answer closer the real life solution", a balance is to be found between perfect mathematical consistency and reality for the scoring to be relevant.

Following this method, we designed 4 PCMs by asking two different security experts to evaluate the relative importance of a technique's attributes pairwise. For each PCM they designed, they had to evaluate the importance of a criterion compared to another according to a specific security goal: first detection difficulty and then accessibility.

Checking a PCM's Consistency Before using it, the consistency of a PCM must be checked. The process is the following, as explained in [4]:

1. Find all eigenvectors and eigenvalues for the matrix.
2. Find the maximum inconsistency λ_m by taking the maximum possible eigenvalue.

3. Calculate the consistency index:

$$CI = (\lambda_m - n)/(n-1)$$

where n is the matrix size.

4. Finally, compute the Consistency Rate (CR):

$$CR = CI/RI$$

with RI the Random Index for consistency, or, in other words, the average consistency obtained when filling the PCM at random, also defined by Saaty [73].

If CR is inferior or equal to 0.1, then the matrix is considered consistent. This threshold is somewhat arbitrary but is commonly used as a practical guideline in the literature. This operation is only necessary once. From the moment a PCM is determined to be consistent, it can be used in the decision-making process.

Among all the PCM we built, we will now use only one: Table 3.1 shows the best resulting consistent matrix we built. The consistencies found for all matrices ranged between 0.025 and 0.015, we selected the lowest consistency rate to work with. In AHP, if the CR exceeds 0.1, it suggests that the judgments may lack sufficient consistency, and the decision-maker may need to review and revise their judgments to improve the reliability of the decision-making process. On the contrary, a low CR indicates that the decision-maker's pairwise comparisons are relatively consistent and adhere to the principle of transitivity, making the results of the decision-making process more reliable and meaningful.

Computing the weight of each criterion Using the PCM, we compute normalized weights to obtain values bounded between 0 and 1. Let A be the pairwise comparison matrix with elements a_{ij}, where $i, j = 1, 2, \ldots, n$ with i, j is the number of rows and columns respectively, n is the number of *criteria*, and K is the total sum of the PCM, or normalization factor. The weighted sum for each criterion is:

$$w_i = \frac{1}{K} \sum_{j=1}^{n} a_{ij}$$

Following the PCM given in Table 3.1, the resulting weights for the criteria are shown

Criteria	Platforms	Permissions	Network Requirements	Frequency
Weight	0.125	0.129	0.502	0.244

Table 3.2 – The weights of a technique's attributes computed using MADM

in Table 3.2.

Compute scores Reaching this step, a score S is computed for every technique each time a recommendation is requested, using the weights of the criteria. The score of a given attribute corresponds to the value of said attribute. For a given technique we compute the score like so:

Let n be the number of criteria, s_i be the score for criterion i, and w_i be the weight for criterion i. The final score (S) for an object is calculated as follows:

$$S = \sum_{i=1}^{n} s_i \cdot w_i$$

Using this method, when we have a group of candidate techniques, we are now able to score them and select the best ones among the candidates (line 10 from Algorithm 1). However, even with only the best techniques the number of fields we could recommend is high, and so we need to score fields associated with the best technique as well.

3.2.4.2 Comparing fields using similarity

This section describes the Similarity scoring we use regardless of the recommendation path we choose, it corresponds to line 16 in Algorithm 1. We use this similarity method to rank ECS fields between them. The different attributes of a field can have different natures. We will use a dedicated method for each field nature to compare them before computing a final score using a weighted sum. To score a field we score separately each of his attributes.

In Table 3.3, different scoring methods are proposed according to an attribute's nature. The boolean scoring method is straightforward, we use the truth value 0 or 1 as a score. For attributes that are sets we have chosen to use the Jaccard similarity to compare them. That is so because the attributes were compatible with the requirements of the Jaccard similarity but also because this method is straightforward: it does not favour any component of the comparison. Since the Jaccard similarity is applied to attributes of the

Attribute	**Attribute nature**	**Description**	**Scoring method**	**Weight**
Presence	Boolean	Availability in investigation	Boolean	20%
Prefixes	Set	Possible qualifiers for a field (*e.g.* *destination* for *ip*)	Jaccard Similarity	10%
Sources	Set	Log sources where field can be found	Jaccard Similarity	25%
Pivot field	Boolean	Present in multiple sources investigated	Boolean	25%
Interest-ingness	Ratio	Average presence in logs	Normal-ized ratio	20%

Table 3.3 – Attributes of a field and how to score them

objects we compare and not the objects themselves, and the resulting similarity scores of the attributes used in a weighted sum, we wished for the similarity computation to be as straightforward as possible. The Jaccard similarity between sets A and B is given by:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

For "interestingness", we use a normalized ratio. It measures how often this particular attribute appears in comparison to all the possible occurrences of this attribute across various data sources. Now, when we talk about "interestingness", we are looking at a broad perspective. It is a ratio that considers the availability of this field in different data sources compared to its potential existence in all conceivable data sources. In simpler terms, it's a way of looking at how rare or common this attribute's value is across all possible data sources, providing an insight into its uniqueness. Finally, we highlight that the attributes prefixes and sources are compared with those of the reference field during the scoring (*i.e.* The flagged field, line 1 in Algorithm 1).

Every attribute was also given a weight during the implementation of the recommender system. The relative importance between the attribute scores was determined by us empirically. Having manipulated these fields a lot during investigations we know which of their components are the most used to make discoveries in investigations. For example, a field that is a pivot is very often present in important discoveries and as such receives a higher weight. The score of a candidate field is computed by doing a weighted sum of the scores of its attributes. Let the score S_{dt} of a field be the sum of the scores of each attribute o_i multiplied by the weight of the attribute p_i.

$$S_{dt} = \sum_{i=1}^{5} o_i \cdot p_i$$

This particular scoring method is used directly in the case of a *safe* or *danger* recommendation (see Figure 3.1).

3.2.4.3 Conclusion on KRAKEN's recommendation technique

In the end, KRAKEN is able to recommend exploration paths in two different ways. On the one hand, in the case where an analyst knows what to do and only needs the right field recommended to him, we only use the similarity process comparing the different fields present in the investigation. On the other hand, if the analyst does not know where his investigation is leading him and need a recommendation for a relevant path to explore within the available data, we will use the MADM as well. That way, the recommendation also uses more abstract knowledge that helps frame the situation being observed by the analyst.

3.3 Evaluation of KRAKEN

As discussed in Chapter 2, evaluating recommender systems is a complex task. Metrics on the actual use of a tool using the recommender system by real analysts, investigating real-life incidents are difficult to obtain. In our case we had access to 7 cybersecurity experts, with varying degrees of experience in incident response. The data to analyse came from cybersecurity exercises.

Taking into account all our hypotheses described in Section 3.2, we designed an evaluation both quantitative and qualitative. That way we were able to assess both the recommendation system in itself and how it is perceived by its users.

3.3.1 Datasets used

For the evaluation we use a subset of the TC3 (Transparent Computing exercise 3) dataset [3]. TC3 has been released by the DARPA as part of their "Transparent Computing" program. The subset of TC3 that we used was captured in identical conditions, but at a much smaller scale in order to limit the number of threats to find during the evaluation.

3. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md

APT	Attack step	# flags to discover	Discovery threshold	Investigation ratio
APT 1	Firefox ad	2	1	10%
	Firefox extension	2	1	10%
	SSH	2	2	20%
	Wget	2	1	20%
APT 2	Pine	3	2	20%
	Tcexec Malware	1	1	20%

Table 3.4 – APT present in the TC3 dataset

This subset contains 19.5 million system call events, from one machine, targeted by the APT. The fields that can be found in this subset are grouped in different object types: file, memory, network, unnamed pipes, and sinks. The subset contains two APT summarized in Table 3.4. The APTs are composed of different steps linked to different ATT&CK techniques or tactics. For example, the two Firefox exploits aim to gain access to the machine and are linked to the ATT&CK tactic "Initial Access" and technique "Exploit public-facing application". SSH is used for network discovery by searching all machines through a discovery of open ports 22, and Wget to exfiltrate data. Pine is an old text-based email client here used to provide a backdoor into the machine and spread a malware: tcexec.

3.3.2 Experimental setup

After a short presentation of our work on recommender systems, we asked the participants about their experience in cybersecurity. We also asked if they had some previous experience in SOC or with a SIEM, in order to classify them in three categories: low, medium and high experienced analysts. Then, we did a rapid presentation of the subset of TC3 used for the investigation. Next, we demonstrated the key features of the platform. After their investigations, we collected their feedback through a qualitative interview.

3.3.2.1 Qualitative interview

The discussion was informal, yet we guided the participants to obtain answers to specific questions, each trying to assess a different aspect of KRAKEN. They are enumerated thereafter:

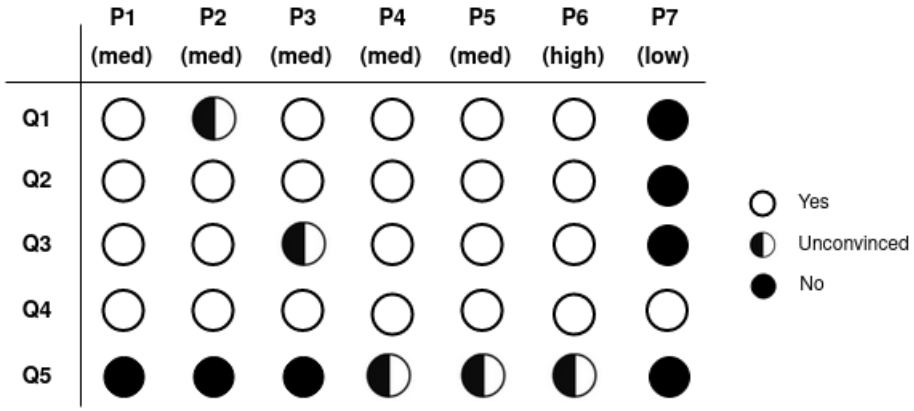

Figure 3.3 – Qualitative evaluation of participants

Q1. **Usefulness:** were the recommendations useful to your investigation?

Q2. **Efficiency:** did KRAKEN help you gain efficiency in your search?

Q3. **Relevance:** did KRAKEN offer you relevant recommendations?

Q4. **Tool future:** in the future would you use KRAKEN during investigations?

Q5. **Clarity:** did you find the recommendations clear and easy to grasp?

3.3.2.2 Quantitative measures

During all the investigations, we collected traces of user actions. The main variables we recorded and used to analyse the investigations are: number of used flags, recommendations generated, followed recommendations and the proportion of threats discovered during the 25-minutes investigation. Each APT contains multiple steps that require flags to be found by the analyst. If we find one of its flags, it counts as discovered, and the threat coverage is computed from the number of attack steps found by each participant. These measures help us quantify the usefulness of the recommendations. Table 3.4 shows the statistical importance (see the "investigation ratio" column) we gave to each attack step in regard to the overall threat coverage.

3.3.3 Qualitative results

Figure 3.3 shows the answers of the participants to the questionnaire from Section 3.3.2.1. The white dot shows a positive answer, the black a negative one and the black and white a mixed answer.

The majority of participants affirms that KRAKEN was useful to them and accelerated them during this investigation. They found the recommendations useful as they helped them progress in their investigations. In terms of efficiency, their feedback is consistent with the fact that they were all able to find parts or all of the APT in only 25 minutes. The answers given by P7 can be explained by his low experience as an analyst. During his investigation, he had little idea of how to proceed, so he could not make use of the tool properly.

The results show that all the participants were enthusiast about the future of the tool. They all saw the benefits in terms of efficiency during an investigation that this research suggests. They agreed to say that this tool helped them get better coverage of the dataset and guided them in the right direction.

The sore spot of the evaluation was the clarity of recommendations. All users felt that they were not highlighted enough in the interface. However, once familiar with the investigation interface, they were all able to use KRAKEN properly and confirmed it did not cause unwanted distractions during their work.

3.3.4 Quantitative results

Table 3.5 shows measures about recommendations aggregated by the severity of the flag: the total number of recommendations made during all investigations, the number of distinct fields concerned by those recommendations, the ratios of relevant recommendations and the ratios of recommendations followed by the analyst over the total number. A relevant recommendation is manually computed by going back on the state of the investigation at the time of the recommendation, with extensive knowledge about the dataset and verifying if following the recommendation would effectively have led to an interesting exploration path.

All recommendations triggered by a *suspect* flag were relevant. The results for the danger and safe severity are less categorical. The recommendations triggered by a safe flag seems to have mostly provided the participants with relevant recommendation according to our hypotheses. On the other hand, the recommendations triggered by a danger flag were less relevant.

Most suspect flag recommendations were followed, showing that not only we were able to provide relevant recommendations, we were also able to convey them to the analyst properly. Safe flag recommendations were also followed 60% of the time, meaning that we have mostly well interpreted the analyst's intent for it. However, the danger flag

Flag Severity	Recommendations	Distinct fields	Relevant	Followed
Safe	5	2	80%	60%
Suspect	13	5	100%	84.6%
Danger	8	3	50%	12.5%

Table 3.5 – Recommendation relevance according to its associated severity

recommendations are only followed 12.5% of the time. We noticed that analysts would often flag as danger and then start back from that point to find other threats, possibly linked to the one already found.

As a conclusion, we find that using only severity to determine the goal of an analyst is not enough. This means that creating two possible recommendation paths may not have been the right solution. In further work with KRAKEN we have used the recommendation path of a suspect flag for any type of severity.

3.3.5 Providing assistance to investigations

Figure 3.4 is a scatter plot of the overall threat coverage in function of the proportion of recommendations followed for each analyst. The analyst's experience is also represented by a colour.

Figure 3.4 shows that P7, who had little to no experience found few attacks in the dataset and did not use the recommendation, as discussed previously. However, the right-most point shows that by selecting a majority of recommendations, the experienced analyst achieved very satisfying results.

Figure 3.4 also shows that, in the case of mid-level experts, the recommendations do not help the analysts discover more than 50% of the threats. While 50% of threat coverage is a good result in 25 minutes, even if our prototype offers relevant recommendations, interpreting them still requires expert skill.

3.4 Improving the recommender system

After the first implementation of the recommender system and its evaluation, we had gathered extensive feedback about KRAKEN. We then decided to improve the recommender system. The evaluation in Section 3.3 was done on the first prototype of KRAKEN which has since been improved by features we mentioned such as a learning feature using

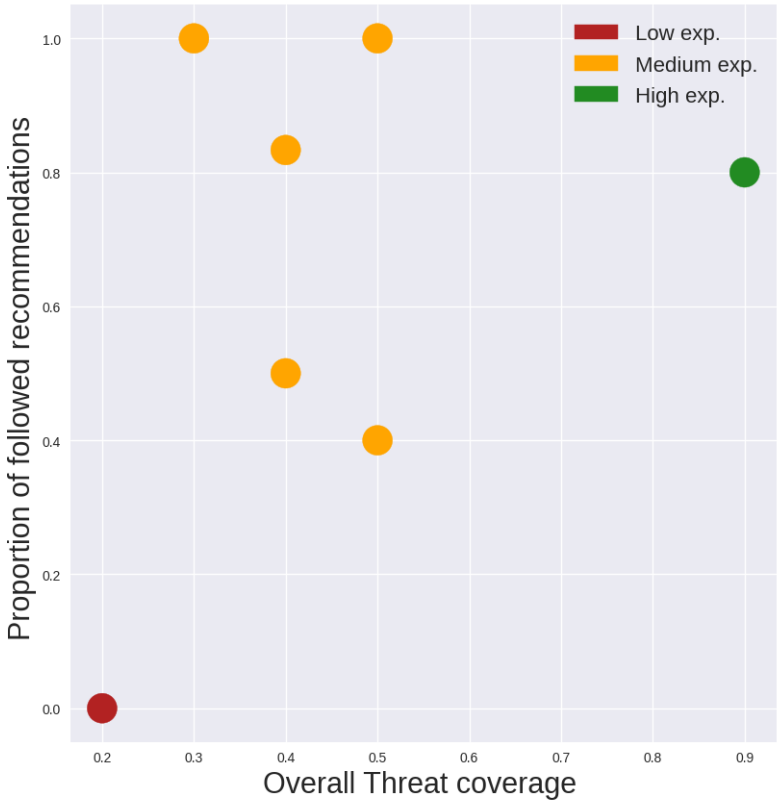

Figure 3.4 – Overall threat coverage discovered by each analyst correlated with the proportion of followed recommendations

feedback on the recommendations, more explainability and transparency, and a redesign of the way recommendations are given to the user. We present briefly these aspects below.

The first and most important feedback we had was the need for feedback on recommendations in order to refine them. Every time a recommendation is followed by a user, the data that motivated the recommendation sees the frequency attribute of the associated ATT&CK technique incremented in order to record its finding in an investigation and, to reflect that it has been useful to an analyst and should be considered in future recommendations.

Second, some smaller improvements were made, to enhance user experience. A feature allowing to click on a recommendation in order to follow it instead of having to search for the field manually was made available. We also added a history of recommendations and

explanations for a recommendation presenting as the potential ATT&CK techniques being observed and motivating the recommendation. These two smaller improvements were greatly appreciated as they bring a lot of transparency and explainability to KRAKEN.

A third improvement is the recommendation process. At the time of the first implementation in Malizen's platform, three possible severities could be associated with a flagged value: safe, suspect, and danger. These levels of security are later replaced by the more commonly known CVSS v3.0 ratings [4]. Following the use of KRAKEN by analysts, we later on made the decision of not using different recommendation paths according to severity. It changed slightly the way we trigger recommendations as we now trigger every time the more complex recommendation path using both MADM and the Similarity scoring, independently of the associated severity. We plan on reintegrating severity into the recommendation process, but as an attribute of the scoring. We have not done so yet because of hybridization considerations we shall discuss in this manuscript's conclusion.

3.5 Conclusion

During the last few years, new tools have been designed to help security analysts in their investigations using visualizations recommendation. However, analysing security incidents is still a challenging task. Indeed, visualization allows analysts to go explore the data faster, but our experience shows that it is not enough. Another issue is the quantity of expert knowledge needed for analyst to conduct investigations properly. During our study of the literature we had seen that expert knowledge is available and waiting to be exploited (**R1**).

In response to this problematic, we presented a recommender system aiming to help the security analyst in her investigation. KRAKEN suggests new paths to explore within log data. It is composed of a knowledge base linking techniques, tactics, data sources and fields, two scoring processes and several recommendation triggers. The knowledge is coming from the MITRE ATT&CK project and is linked to real log data using the ECS project. For us, it was also a way to show that we could use standards from the domain in order to solve our problematics.

We also evaluated KRAKEN with seven cybersecurity experts, whose experience as analysts were various. Our evaluation shows that recommendations are relevant most of the time, and when followed help security analysts during incident response. Participants

4. https://nvd.nist.gov/vuln-metrics/cvss

to the evaluation also noted that the recommender system did not distract them during their investigations while providing insight.

Following the feedback from the evaluation, we implemented some features enhancing recommendations and user experience. We also did a second iteration of implementation in order to add some new features asked for during the evaluation. This laid the groundwork for another evaluation on a larger scale that we leave as future work. Yet, the results help us reach a better understanding of all possible use cases for KRAKEN as well as evaluate our new features like the feedback loop.

These results show promises for our next topic: user intentions. Since we saw that the user intent associated with a severity is more complex than we thought, a larger pool of participants could better frame user intents. However, understanding users as well as we can and making a recommender system that helps them be efficient is one of the bigger part of our research, so we decided to devote a full contribution to this topic. In the next chapter we will focus on **R2**, our second research question.

Chapter 4

Gaining a better understanding of analyst intentions to make recommendations

Not all those who wander are lost.

— The riddle of Strider, J.R.R. Tolkien

MIMIR

This chapter is about gaining a better understanding of analysts during cybersecurity investigations. We believe that by modelling intentions that analysts might have while exploring data we can help them find attacks faster. We 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.

4.1 Introduction

In Chapter 3 we confirmed that recommender systems can be an interesting solution to help analysts during their investigations, but we also showed how difficult it can be to make recommendations at the right time. Indeed, the investigation goals of an analyst can

change rapidly while they investigate. What led to this second contribution began with research on how to trigger recommendations at the right time, but we rapidly realized the topic could be exploited for a new contribution. In Chapter 2, the literature shows that recommender systems for cybersecurity almost always rely on technical data to make recommendations. We wish to see if we can obtain interesting results by using empirical behaviour data produced by analysts, during investigations, rather than expert knowledge. Throughout this chapter we show how understanding and modelling user intentions has proven to be an interesting problematic.

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. We believe this recommender system would help reduce the total time needed for incident handling and limit the impact of the incident. However, it requires us to find a way to successfully model user intentions from their actions, and inversely, to be able to recommend actions that carry out specific intentions within an investigation (**R2**).

Doing so will be challenging because, in order to keep the recommender system from being extremely intrusive during investigations and directly ask analysts their intentions, we are going to need to infer them from their behaviour and the actions they do during said investigations. There we face some data challenges we have described in Chapter 1, namely data scarcity and completeness. Indeed, we will be doing this work in Malizen's platform, a controlled environment, which is good since we know exactly what actions an analyst can execute and can attach to them as much context as we need. On the other hand, we are limited by the platform as we can only use the investigation actions available to us to understand analysts and make recommendations.

In this chapter 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. 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*: the intention of focusing on a specific value of part of the

data during an investigation.

To design this new recommender system, we create 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 that match 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. Some data for this evaluation notably comes from the CERBERE exercise, our third contribution presented in Chapter 5.

This chapter is organized as follows. Section 4.2 presents a use case where we show a relevant use for MIMIR. Some additional literature about user intentions and UEBA (User and Entity Behaviour Analytics) are presented in Section 4.3. We present an overview of the complete recommender system in Section 4.4. Section 4.5 presents the model we have designed for the recommendation engine. Section 4.6 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 4.7. In Section 4.8 we conclude and discuss future work.

4.2 Use case: understanding a user's intentions

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 [1]. The attacker used Powershell, and an administrator account without a password [32].

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 detecting the user intention, we can find out the next most probable intention the user might have and recommend some related actions to undertake. In doing so, we allow analysts to focus on their expertise during investigations instead of the platform they are using to investigate. Our goal is for the recommended actions to be the most relevant for analysts.

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, found through a *broadening* of his search, the analyst might have the intention of finding out the exact log that identifies this attack by *deepening* his search. The recommendation, understanding this transition in intentions 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.

4.3 Related Works about user intentions

Recommender systems often aim to refine recommendations by considering the context in which they are provided. However, not many studies have focused on using user intentions for improving recommendations. This is mainly because inferring precise intentions solely from user actions and platform usage, without directly asking users, is challenging. Additionally, it often requires access to potentially sensitive information.

Despite these challenges, some research shows promise in understanding and utilizing

1. https://nvd.nist.gov/vuln/detail/CVE-2020-1472

user intentions for recommendations. A common approach in this area is known as User and Entity Behaviour Analytics (UEBA). UEBA was originally developed to enhance behaviour categorization in cybersecurity, specifically to detect advanced attacks. Various studies have proposed different methods for understanding user and entity behaviours, as detailed in the work by Khalik *et al.* [44]

The interesting point to remember from Khalik *et al.* [44]'s work is that UEBA is presented as a new approach to detect user activities and insider attacks. UEBA employs predefined rules, anomaly detection, and machine learning techniques to analyse data from various sources such as system logs, application logs, and network traffic. The paper argues that UEBA can effectively identify anomalous user activities, assign risk scores, and detect compromised users. However, UEBA models are known to have difficulties in interpreting behavioural changes, notably because of the volume of data to interpret, its variety and the contextual information of various natures. Since we work in a controlled environment, and have at our disposal an investigation platform where possible user actions are known, their metadata as precise as we need, and that we work on one investigation at a time we do not face these problems. We believe that by focusing on our much smaller scope, we can obtain good results.

One of the works present in the literature is some research conducted by Philip A Legg *et al.*, on Corporate Insider Threat Detection [47]. Their study introduced a method to detect insider attacks in organizations by creating user activity profiles based on roles. The system generated profiles using user activity and their associated roles. The research used ten datasets from CMU-CERT, each representing different types of activities such as file access, web usage, email... Data was parsed to extract Activity Name, User ID, Device ID, and Time Stamps. Modules processed this data, appending it to daily user profiles, while a content parser extracted data from various sources and computed Linguistic Inquiry Word Count [2]. The profiles were assessed to generate three levels of alerts: policy violations and pre-recognized attacks, threshold anomalies, and deviation-based anomalies. Synthetic data was used for testing, with 365 days of activity, 15 days for training and the rest for testing. The best results across 10 test scenarios showed a precision of 42% and a recall of 100%. However, while the system aims to alleviate the efforts required of analysts, the practical implications and user experiences of analysts utilizing the system are not explicitly elaborated. Another concern we have is the acknowledgment of organizational-dependent characteristics that can impact the effectiveness of the system.

2. https://www.liwc.app/

Focusing on understanding user behaviour, Moskal and Yang [55] 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, it does not make use of the actions directly but rather work with established knowledge. This work proposes a new approach to deal with the increasing complexity of intrusion alerts. It helps security analysts understand these alerts better by translating them into an easily comprehensible format. This approach, called Pseudo-Active Transfer Learning (PATRL), combines language models, active learning, and pseudo labels. PATRL achieves high prediction accuracy (85% for top-1 label and 99% for top-3 labels) on new, previously unseen data. It also provides security analysts with extra metrics like Monte-Carlo Dropout Uncertainty and Pseudo-Label Convergence Score for each alert prediction, boosting their confidence in the results. This method is unique as it effectively addresses the challenge of having limited labelled data by leveraging transfer learning, active learning, and pseudo labels. It significantly enhances security analysts' ability to interpret complex intrusion alerts by capturing attacker intentions through the Action-Intent-Stages approach. Despite the fact that making alerts more accessible to users, this work does not consider the exploration part of investigations.

Zhong *et al.* [92] improves the performance of data triage and especially helps less experienced users in making the right decisions. They use the recorded triage 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 associating 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 the investigative part of incident response.

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 specific tasks, we take a different approach. Our idea is to

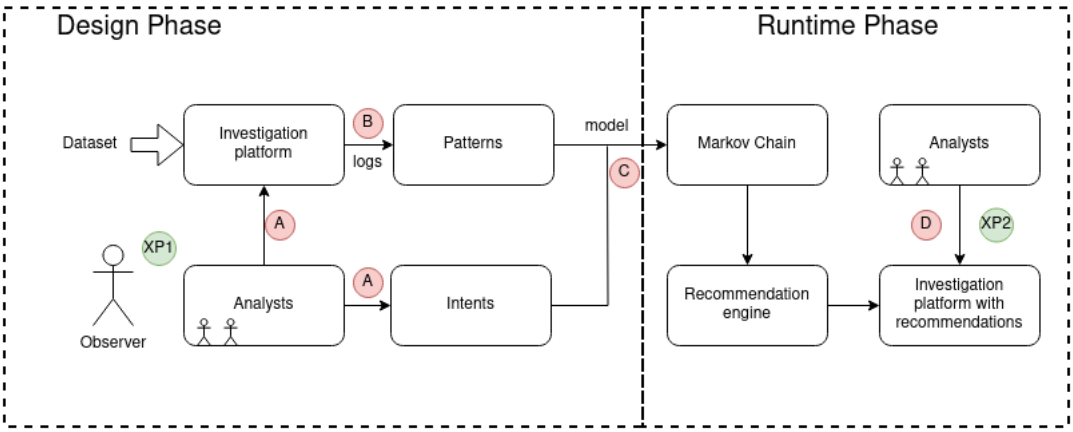

Figure 4.1 – Overview of the design and runtime of the MIMIR recommender system

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, however some of the work we presented can guide us in the right direction. While all these papers try to understand the behaviour of attackers, some things might be analogue to understanding analysts. Notably we will draw inspiration from Moskal and Yang's work [55] to try and translate actions into interpretable intentions by modelling them in groups associated with an intention. We will also take a page from Zhong *et al.*'s [92] book by using past actions to try and predict the actions to take in new situations.

4.4 Overview of MIMIR

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 4.1. It should be noted that Malizen's investigation platform (see Section 1.4) is used to host MIMIR.

4.4.1 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 4.1). 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 4.5.2. 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 recommender system. 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 4.5.

4.4.2 Runtime phase

In the runtime phase we focus on how we managed to use the resulting model in order to provide recommendations. In **C** (see Figure 4.1), 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, because they model very well the statistical transition of states we wish to represent with analyst intentions. 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 4.5.

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 intention will be given to the recommender system that will then decide on the most probable intention the user can have and recommend

associated patterns of actions.

Malizen's platform presents the recommendations in the shape of a notification showing the actions recommended along with a button to execute them automatically. The idea is to integrate said recommendations as well as possible within the workflow of the analyst.

4.5 Intentions, patterns and actions inside MIMIR

Within MIMIR, the actions analysts do during an investigation, the patterns of actions reoccurring during said investigations and analyst intentions are intricately linked. In this section we will detail how we link them in MIMIR.

4.5.1 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 [3] dataset and the second one is the TC3 [4] dataset. This corresponds to **XP1** in Figure 4.1. 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.

3. http://www.vacommunity.org/VAST+Challenge+2012
4. https://github.com/darpa-i2o/Transparent-Computing/blob/master/README-E3.md

— **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.

4.5.2 Collecting actions and pattern creation

We have at our disposal the actions performed by the participants using the platform described in Section 1.4.2. This platform allows users to perform multiple actions, and we wish to discover groups of actions that carry user intentions.

4.5.2.1 Research & experiments on user actions

To extract meaningful patterns from user actions in the context of a cybersecurity investigation, we focused on two key variables: the actions themselves as elements of the pattern and the size of the pattern. These variables directly influence the frequency of pattern occurrences in investigations and their ability to capture user intentions.

To narrow down the relevant actions for cybersecurity investigations, we filtered all possible actions into four categories, resulting in a curated list (AL) displayed in Figure 4.2. The list includes more complex actions:

— **flag** for saving values with platform states and comments
— **change visualization** for switching visualizations
— **navigate on platform** for changing pages within the platform

Choosing the pattern size was a critical decision, balancing the need for patterns frequent enough to reveal recurring patterns in investigations and rare enough to attribute them to specific user intentions. In our manual exploration, we found that patterns composed of only 2 actions were the most insightful. Patterns of sizes 4 and 5 occurred too infrequently, and considering the limited number of actions selected, lots of patterns of this size were very similar because they contained permutations of actions that could be executed in no particular order. As a consequence, working with groups of actions of this size would have made us consider user preference and habits in our patterns. Surprisingly, patterns of size 3 exhibited less clear intentions than those of size 2. That could also be explained by the fact that considering the limited number of actions available, most

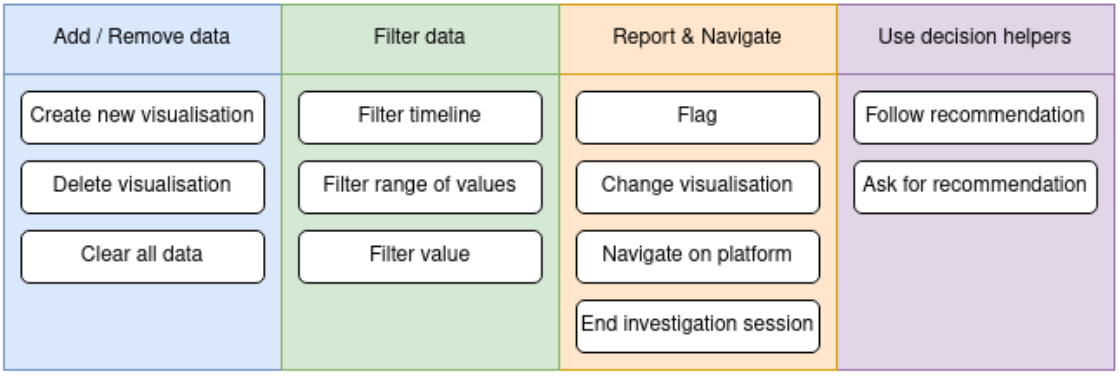

Figure 4.2 – List of actions selected as carrying semantic intent

intentions seemed executed over less than 3 actions. Additionally, further attempts to extract patterns from all available actions in Malizen's platform, both with and without our category-based abstraction (see Figure 4.2), were inaccurate.

We also tried to implement some automation in order to extract and select relevant groups of actions. Various theories were tested using different investigation traces, extracting patterns of sizes 2 to 5. Our goal was to see if we could identify interesting patterns without manual review. Our initial attempts were to eliminate outliers by setting thresholds at extreme frequencies, but after examining the outlier manually, this method showed to be ineffective. We also tried exploring alternative methods, such as using the mean frequency and standard deviation in order to select patterns according to their frequency, but they also fell short. The issue with trying to select patterns in this way is that the notion of intention is psychological and difficult to formalize so depending on the analyst the patterns used, and their frequency may vary. We believe that we have avoided as much sample bias as possible by finding our patterns by using ten different investigations coming from five different analysts, but unfortunately we do not have a sufficient amount of data to obtain some kind of statistical confidence. Ultimately, due to the complexity and nuance of identifying interesting patterns automatically, we opted to manually select patterns based on their perceived significance.

4.5.2.2 Formal definition of patterns

As patterns will be the atom of our model we need a formal definition of them. 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 the analyst is trying to expand the scope of his research whereas if the filter is being enabled he is restricting his scope of research: two very distinct intentions. Let us formally define what constitutes a pattern. Let AL be the Action List. A pattern for two actions i and i' in contexts k and k' is defined by:

$$P_{ik}^{i'k'} = ((action_i, context_k), (action_{i'}, context_{k'}))$$

$$with : (i, i') \in |AL|, (k, k') \in |\{contexts\}|$$

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, $\varnothing$), (filter-value, disable)) shows the user *broadening* his scope of investigation by removing a filter after an exploration recommendation, while ((follow recommendation, $\varnothing$), (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.

As a result, with all possible actions and possible contexts combined, we end up with 16 unique actions and context pairs, and we obtain 120 possible patterns.

4.5.3 Linking patterns to user intentions

With each of the ten resulting investigations from **XP1**, 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 exact list of patterns can be found in Listing 1. The resulting model links concrete actions within the investigation platform with user intentions. We can now build a recommendation engine based on this model.

4.6 MIMIR's 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 platform, understand the intention behind it, and use the Markov chain to infer the most probable next intention of the analyst and recommend the next actions to do, in agreement with the inferred intention.

4.6.1 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. Because of this, Markov chains seemed to be a natural representation to model this behaviour. Additionally, we did not need the model to have any memory other than the present state, so Markov chains imposed themselves. Thus, we implemented a Discrete Time Markov chain [67] and, in the rest of this chapter 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.

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 4.3 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.

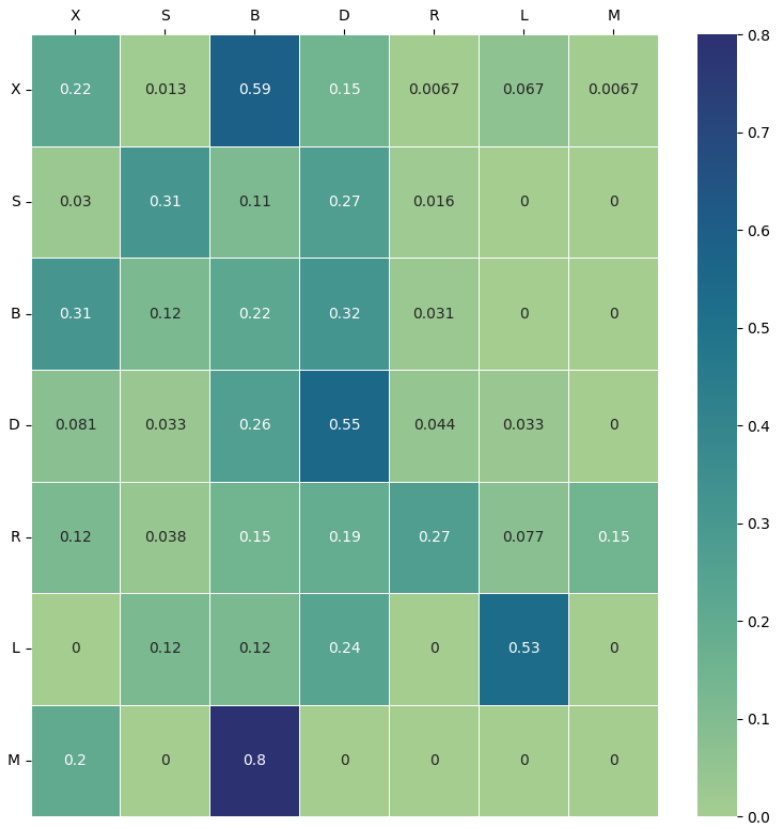

Figure 4.3 – The transition matrix for the Markov chain with the intentions from Section 4.5.1, built from the 10 investigations of **XP1**.

4.6.2 Triggering recommendations

Our goal with the pattern detection was to find out the best trigger for our recommendations. Our previous work described in Chapter 3 had informed us that triggering recommendations at a specific point was interesting because it allowed to trigger a recommendation whenever wanted. However, we wanted recommendations to integrate more with the investigation flow of an analyst. 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.

We realized that by capturing analyst intentions we had created the perfect environment to trigger a recommendation at any moment in the investigation, should a pattern be detected. To do so, we implemented a detection module that records the last user action made and, when a new one is caught, finds out if they constitute a pattern known to be associated with a specific user intention. If the pattern exists, we trigger the rec-

ommendation of the next possible intentions 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.

4.6.3 Presenting recommendations

For the instantiation of recommendations we focused on recommending actions to be performed in the investigation using the actions of Figure 4.2. The recommendation is the most probable next intention the user might have according to the matrix in Figure 4.3. Recommendations are then presented to the user by showing them the intention we predict as well as the actions that can be performed to realize that intention.

4.7 Evaluation of MIMIR

The evaluation of this work was complex for two reasons. First, evaluating recommender systems is not standardized, and no good method stands out, as discussed in Section 2.3. Secondly, evaluating a recommender system can be complex because since we evaluate them based on their users, the evaluation itself becomes 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 recommender system.

The following sections present two different evaluations of the MIMIR recommender system. First, we evaluate the quality of the Markov chain model by comparing Markov chains built with different datasets as well as randomly built ones. The second one is an evaluation of the efficiency of recommendations using the prototype. For this evaluation we use results from the exercise that constitutes our third contribution, described in Chapter 5, and the experiment **XP2** from Figure 4.1. Each evaluation starts with a description of the data we used during the experiment. For better readability, throughout the whole evaluation section, we refer to Markov chains by using their transition matrices.

4.7.1 Quality of the Markov chain

This first evaluation has the goal of testing the model we built. User intentions are notoriously difficult to capture, and we wish to verify that our model has indeed captured

some analyst intentions, and if so, how much.

4.7.1.1 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 4.1. All datasets used during this preliminary evaluation are described in Table 4.1. 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 4.3.
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.

4.7.1.2 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

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 4.1 – Dataset information

from the sets **SUPSEC Traces** and **BotsV1 Traces**. These investigations were all conducted by participants that were not guided by us; it is the same method used to build the reference matrix. We also generated a set of 100 random matrices to use as a comparison point.

For each resulting transition matrix in the SUPSEC dataset, the BotsV1 dataset 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 4.4. 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.

4.7.1.3 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.

Results are presented in Table 4.2. For each investigation present in a trace of investigations (a column in Table 4.2), 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 column since the number of in-

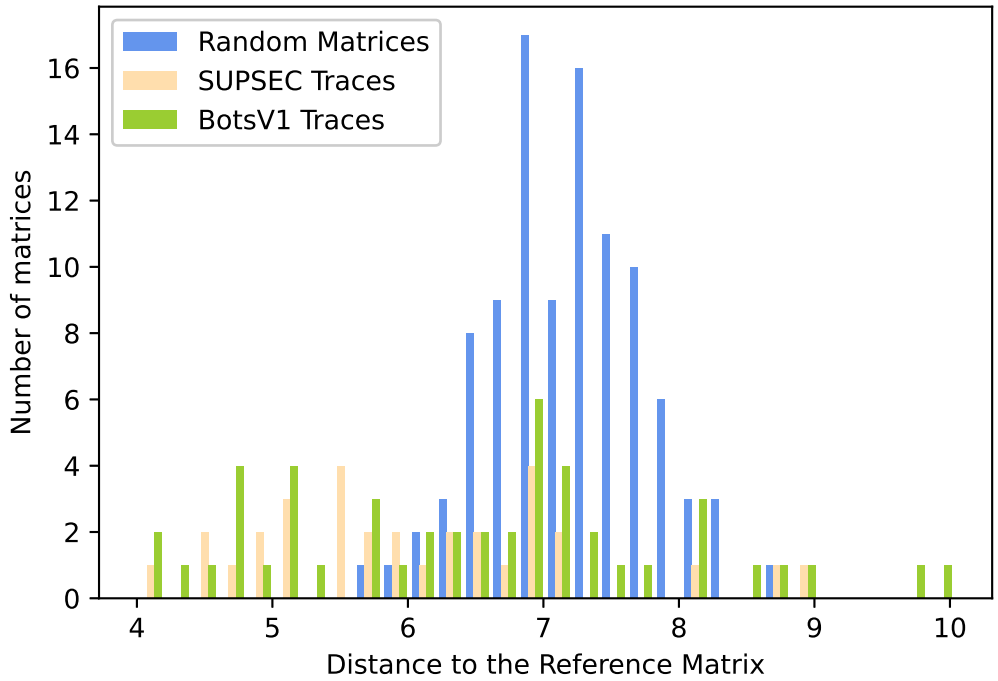

Figure 4.4 – Representation of the distribution of the distance of multiple sets of investigation traces to a reference matrix

vestigations 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:

$$L = |\sum_{i=1}^{n} ln(Transition_M(A_i, B_i))|$$

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.

4.7.2 Prototype and recommendations evaluation

The last experiment we conducted was part of a larger security exercise organized as a red and blue team capture the flag. We only discuss briefly the details of the experiment

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**
mean(Random Matrices)	10.131	9.676	14.414

Table 4.2 – Mean log-likelihood of sequences being generated by a specific Markov chain

as it will be fully described later in Chapter 5, and focus on the obtained datasets. This corresponds to **XP2** in Figure 4.1.

4.7.2.1 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 4.7.1.2 as an input to the recommendation system due to time and implementation constraints.

4.7.2.2 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 4.3). 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 2-3 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 4.5 shows that for every transition considered, recommendations were followed a

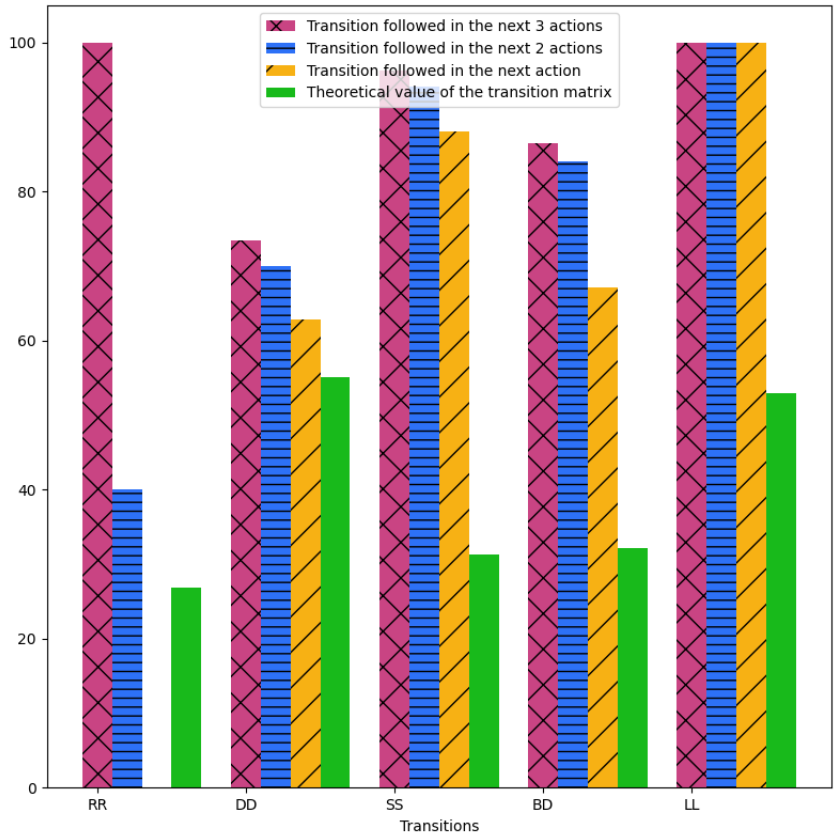

Figure 4.5 – A representation of how much the recommendations were followed compared to the Markov chain transition probability

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 (see Section 4.5.1) 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.

Discussion

For the entirety of this work, we focused on how the users reacted to recommendations, how they were perceived, whether they were followed and considered useful and, how we could improve them. During experiments, we gathered feedback and turned it into multiple improvements to implement into MIMIR. Users mostly welcomed the idea of recommendations. The idea of capturing user intentions in order to predict the next most probable one and to recommend actions to perform as a follow-up was appreciated. These discussions allowed us to design three working axes for future recommender systems. First, the need to refine the trigger for a recommendation. Every recommendation does not have the same value to a user. For example a recommendation to do something when nothing is going on is less interesting than some actions to undertake when in the middle of researching a promising lead. This means adding rules to adapt and refine the frequency of recommendations as well as the way they are offered to the user. Second, the need to be able to test out the recommendations more easily. This means we need to offer the possibility of substantiating recommendations more easily during their investigations. Lastly, the transparency and explainability of recommendations came up a lot. In the cybersecurity field especially, knowing the reasons behind decisions is crucial and should be shown through recommendations. We have already designed the first and third improvements of this list and are in the process of integrating them to the prototype.

4.8 Conclusion

In this chapter 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.

With this second contribution, we tried to address various problematics identified either in the literature and during the development of our first contribution. The first and most important question was to find out if understanding users during their investigations gave us a possibility to enhance security analysts (**R2**). We found out that modelling abstract user intentions and linking them with actions during investigations was complex, even in a controlled environment such as ours. Indeed, without directly asking their intentions to users, a lot a metadata is needed to describe refined intentions. Inversely, even considering intentions, recommending concrete ways of realizing them is equally challenging.

We are currently working on improving MIMIR. In the near future we are considering working on further substantiating the recommendations by hybridizing this recommender system with KRAKEN, 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. The idea behind this hybridization is that while KRAKEN provides very rigid recommendations, they are precise and concrete, so they can be executed easily by an analyst. MIMIR on the other hand is better at understanding when to make a recommendation and towards which investigative goal. We would like to test different hybridization methods to see if we can combine both their strong points.

Finally, during this chapter we have discussed an experiment conducted that gave us the necessary data to evaluate our recommender system. This experiment was also designed and conducted by us during this thesis along with co-workers from our research team. In the next chapter we will delve into more detail about the data problematic that followed us during this PhD.

Listing 1 The 39 resulting patterns linked to user intentions

```
[
    # Startup / Discovery (S), Broaden search (B), Deepen search (D)
    # Report findings (R), Backtrack (X), Searching for a new lead (L)
    # Guided by recommendation (M)
    (("filter-value", "include"), ("filter-value", "disable"), "D"),
    (("filter-value", "disable"), ("drag-drop-card", ""), "B"),
    (("classification-recommendation", ""), ("flag", ""), "R"),
    (("change-chart", ""), ("drag-drop-card", ""), "B"),
    (("filter-value", "disable"), ("filter-value", "include"), "B"),
    (("recommendation-chosen", ""), ("filter-value", "include"), "D"),
    (("flag", ""), ("navigation", "/leads"), "R"),
    (("drag-drop-card", ""), ("remove-card", ""), "X"),
    (("drag-drop-card", ""), ("change-chart", ""), "B"),
    (("change-chart", ""), ("filter-timeline", ""), "B"),
    (("filter-value", "include"), ("drag-drop-card", ""), "B"),
    (("filter-timeline", ""), ("filter-value", "include"), "D"),
    (("drag-drop-card", ""), ("filter-value", "include"), "D"),
    (("voluntary-recommendation", ""), ("recommendation-chosen", ""), "B"),
    (("filter-value", "include"), ("flag", ""), "R"),
    (("navigation", "/leads"), ("navigation", "/"), "L"),
    (("filter-value", "include"), ("change-chart", ""), "D"),
    (("change-chart", ""), ("remove-card", ""), "X"),
    (("drag-drop-card", ""), ("filter-timeline", ""), "D"),
    (("filter-value", "include"), ("filter-value", "include"), "D"),
    (("drag-drop-card", ""), ("voluntary-recommendation", ""), "B"),
    (("recommendation-chosen", ""), ("remove-card", ""), "X"),
    (("change-chart", ""), ("filter-value", "include"), "B"),
    (("flag", ""), ("recommendation-chosen", ""), "M"),
    (("drag-drop-card", ""), ("drag-drop-card", ""), "S"),
    (("change-chart", ""), ("change-chart", ""), "B"),
    (("filter-value", "disable"), ("remove-card", ""), "X"),
    (("filter-value", "include"), ("navigation", "/leads"), "L"),
    (("filter-value", "include"), ("filter-timeline", ""), "D"),
    (("filter-value", "disable"), ("filter-value", "disable"), "B"),
    (("remove-card", ""), ("remove-card", ""), "X"),
    (("flag", ""), ("flag", ""), "R"),
    (("filter-timeline", ""), ("change-chart", ""), "D"),
    (("recommendation-chosen", ""), ("recommendation-chosen", ""), "B"),
    (("filter-value", "include"), ("remove-card", ""), "X"),
    (("remove-card", ""), ("drag-drop-card", ""), "B"),
    (("filter-timeline", ""), ("drag-drop-card", ""), "B"),
    (("navigation", "/ingest"), ("navigation", "/"), "S"),
    (("filter-timeline", ""), ("filter-timeline", ""), "S")
]
```

Chapter 5

Data: the issue of obtaining it and using it for evaluation

It is a capital mistake to theorize before one has data.
— **Arthur Conan Doyle, Sherlock Holmes in "A study in Scarlet"**

CERBERE

Experimenting in cybersecurity requires manipulating reliable and realistic data. In particular, data from the observation of a complete campaign is rarely available, due to its high sensitivity and the difficulty of accurately labelling datasets. This situation harms the reproducibility of research results and therefore their impact. In this chapter, we present the CERBERE project that proposes a reproducible attack-defense exercise that generated a labelled dataset usable for research purposes. The attack-defense exercise is first composed of an exercise for red teamers automatically deployed with variable attack scenarios. Second, an investigation exercise for blue teamers operated through a log investigation platform using the system and network logs generated during the attack phase. Obtaining such data helps us evaluate our prototypes from Chapters 3 and 4, while addressing an important issue in the field of cybersecurity.

5.1 Introduction

This last contribution chapter tackles an entirely different topic than recommender systems, yet it seemed essential for us to address it during this thesis as it has greatly

helped us for the two other contribution chapters. One of the main academic issues is the lack of data [76]. Cybersecurity suffers from data scarcity for good reasons such as the sensitivity of security data, the high cost and low reliability of data anonymization, the difficulty of capturing, storing and releasing data to the public [46]. Additionally, even in the case where a dataset is released, complete ground truths, *i.e.* documentation about the attacks and traces of the attackers in a dataset, are rarely provided. We also need such datasets to evaluate our work.

For this thesis, in Chapter 3 we were able to evaluate KRAKEN using only one dataset because at the time we only had two available: one to implement and test the prototype and one to evaluate it. Fortunately, we were able to create two more to evaluate MIMIR in Chapter 5, but with a lot of efforts. Our situation is common and can be explained in a number of ways, but mostly, the reason is that in a three year time-frame, creating a new dataset is enough work to become another whole thesis. Fortunately, during this thesis, we had the opportunity to join hands with other PhD candidates and generate our own data.

CERBERE comprises two main parts: an initial Capture-the-Flag (CTF) exercise where players attack a uniquely generated infrastructure and a subsequent investigation phase. In the exercise, player actions undergo monitoring from three perspectives: pentesting activities in their web browser, network activity, and host operating system activity. This collected data is investigated in the second phase, where players reconstruct an attack scenario by exploring the logs. Our aim with this contribution is to initiate a novel approach to data generation, catering primarily to our project's requirements and those of our collaborators. This way, by controlling the scenario, architecture, and monitoring, we believe we can conduct evaluations with less bias and focus more on the aspects we intend to evaluate. For instance, in the context of a recommender system, understanding the scenario and attack procedures allows us to precisely understand how they appear in logs. This understanding enables an efficient evaluation of the usefulness of a recommendation.

Additionally, this contribution is completed by the CERBERE dataset containing all the logs produced during the CERBERE exercise, thus addressing our third research question: **R3**. The dataset has been generated with the participation of 22 players: 13 red players and 9 blue players with three levels of difficulty. We explain how this dataset was labelled, enabling us to reconstruct the evolution of the players. Several outputs are distributed with this publication [1]:

1. https://gitlab.inria.fr/cidre-public/cerbere-dataset/

— The software for generating the infrastructure and executing these instances on a regular Linux operating system.
— The raw dataset of the logs of each red team player.
— The enriched dataset containing labels for the logs to highlight the evidences of intrusion.

Unfortunately, the blue team logs cannot be shared as they can contain sensitive information; in any case they are very specific to the Malizen platform, making it difficult to reuse them. Nonetheless, they are used in the evaluation of MIMIR (see Chapter 4).

In the following sections, we go into further details regarding CERBERE. Section 5.2 discusses three different works we have found that tackle the same issue as we do. Section 5.3 presents the methodology employed to build our red/blue team exercise. In particular, we give details about the pentesting scenario and the information collected during a pentest. Section 5.4 presents how we conducted red team and blue team experiments with the players. We will also discuss how CERBERE has helped us to evaluate our recommendation algorithms. Finally, Section 5.6 details the labelling of the logs to form the CERBERE dataset. Section 5.7 concludes the chapter.

5.2 Cybersecurity data generation in the literature

Different initiatives have been taken to produce log data containing attacks. Whether it is needed to evaluate some work or simply to simulate a real situation, this problem has been approached before, and we can wonder what makes it so challenging; researchers from the Canadian Institute for Cybersecurity [19] produced a dataset generated through a testbed and containing multiple known network attacks: CICIDS2017 [76]. This dataset was initially designed to evaluate the efficiency of IDSs. However, researchers have studied in detail this dataset and have found multiple issues:

— Panigrahi *et al.* first present how this dataset contains many labelling issues [60].
— Joosen *et al.* presents a new version of the dataset after they found that some attacks within the dataset did not work as intended [26].
— The same team presented a second paper where they detail all their findings after a careful inspection of the packet captures of the dataset [51].
— The most recent paper gives new corrections for some attacks, labels some unlabelled attacks and deletes some redundant packets within the dataset [46].

We can make multiple hypotheses to explain why this dataset contains so many errors,

but mainly, it seems these errors are caused by a dataset too big in volume, over too many source files, trying to represent too many attacks. Additionally, it seems that the biggest issue with the dataset is the labelling. Using a dataset that is not verified carefully for errors can introduce a lot of bias, and that is why some researchers have started implementing their own methods to generate data.

SOCBED [84], addresses this issue by proposing a testbed, designed to generate reproducible, adaptable, and realistic log datasets for cybersecurity experiments. The paper includes a survey of log data generation in cybersecurity experiments, and design goals for sound experiments. The introduction of SOCBED as an open-source cyberattack experimentation testbed, and a practical attack detection experiment demonstrates the reproducibility and adaptability of log data generation. SOCBED is freely available and allows researchers to build on existing scenarios, improving the comparability of results in cybersecurity research.

However, despite the merits of all these approaches from the literature, they present important flaws [26] due to the complexity of the task. First, the presented architectures are rigid and difficult to change. Then, the management of the attack surface is complex, and must be handled with care because of potential side effects. Indeed, implementing a vulnerability often implies installing older package versions or modifying rights, creating a significant probability to open more than one way for attackers to get inside the system. Finally, the limitations of large, complex datasets, such as labelling errors and biases, highlight the necessity for a more controlled and adaptable approach.

In this context, we aim to develop our own solution by leveraging the insights gained from previous endeavours like SOCBED. While SOCBED addresses the reproducibility and adaptability of log data generation, it is essential to acknowledge its limitations, as noted in the literature. Existing architectures are often rigid and challenging to modify, and the management of the attack surface requires careful consideration due to potential side effects. Moreover, the data produced by these architectures tends to favour the attackers' perspective, neglecting the defenders.

We then turned to projects closer to the industry, in hope of finding usable tools. SecGen [75] is a project available on GitHub and its main goal is to provide cybersecurity students with a way to learn penetration testing with more diverse methods than the too well-known ones, such as Metasploit. SecGen generates randomized vulnerable systems. It creates virtual machines based on scenario specifications, detailing constraints and properties, such as specific vulnerabilities, services, networks, users, or content. The

scenarios can be broad, requiring attackers to exploit randomly selected vulnerabilities, or more specific, targeting certain services or exact vulnerabilities (by CVE). SecGen leverages an XML configuration language, reads its configuration, applies logic for randomizing scenarios, and utilizes Puppet and Vagrant to provision the required VMs. This dynamic approach aims to provide ongoing challenges for students, avoiding the static nature of traditional hacking challenges. However, while this approach is very interesting, it does not include itself within realistic situations. It offers the possibility to do penetration testing, not to actively execute a full killchain on an information system. However, SecGen provides a solution for a very specific use case, does not scale up when a bigger architecture is needed and makes it difficult to provide a real and complete scenario.

Our objective is to overcome these shortcomings and develop a solution that not only ensures reproducibility and adaptability but also addresses the limitations identified in previous approaches. We envision a solution that provides more flexibility in architecture, simplifies the management of the attack surface, and offers a balanced perspective by promoting both the attackers' and defenders' standpoint in the generated data. Through this, we aim to contribute to the improvement of log data generation for cybersecurity experiments.

5.3 The CERBERE Project

CERBERE is designed as a cybersecurity exercise in two parts: pentesting a group of vulnerable infrastructures using a similar attack scenario but with variations in the implementation of the scenario, and investigating the logs of said pentest.

5.3.1 Overview

Figure 5.1 summarizes the global architecture of the project. The first phase of the CERBERE exercise is the attack phase. Each player (represented in red) obtains his own target instance: a virtual infrastructure hosting the 3 machines to be attacked. All the instances are automatically deployed starting from a generic description of the attack scenario and closely monitored in order to gather data. Each instance deploys a variation of the generic scenario: instances differ from each other by the involved operating systems, installed software and their vulnerabilities. Each instance in the CERBERE exercise was unique to each participant allowing him to discover its own way of performing the attack

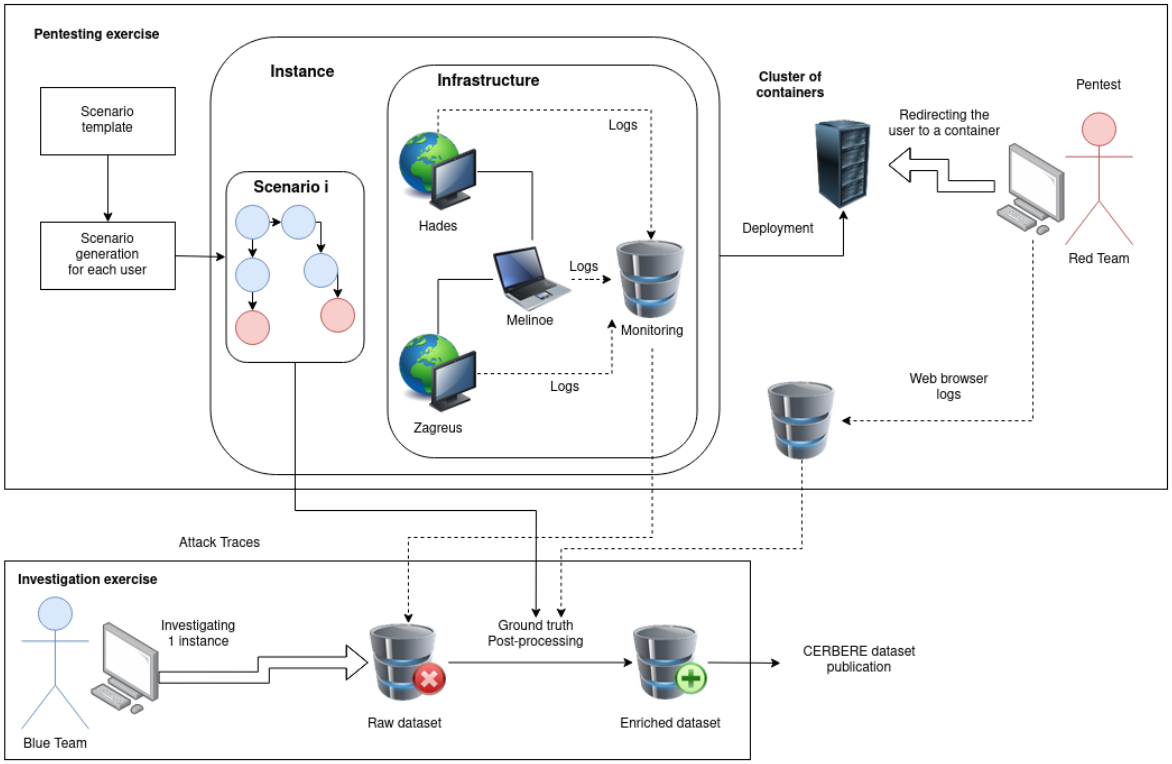

Figure 5.1 – Overview of CERBERE architecture

campaign. These aspects are presented in more details in Section 5.3.2. To these instances, two more machines implementing the GHOSTS framework [2] were deployed in order to generate fake life on the infrastructure. However, we do not represent them for more clarity as they are not relevant to the attack scenario.

In the second phase of the CERBERE exercise, new players (the blue team) receives the logs associated to a pentest session of the first phase, except browser logs that would not be available during a real investigation. The logs have been acquired from several monitors, as described later in Section 5.3.6. Some post-processing actions described in Section 5.6 in order to detect and label the log lines which identify the attacks are also performed. Blue teamers investigate the logs with Malizen and are expected to flag assets corresponding to attacker actions.

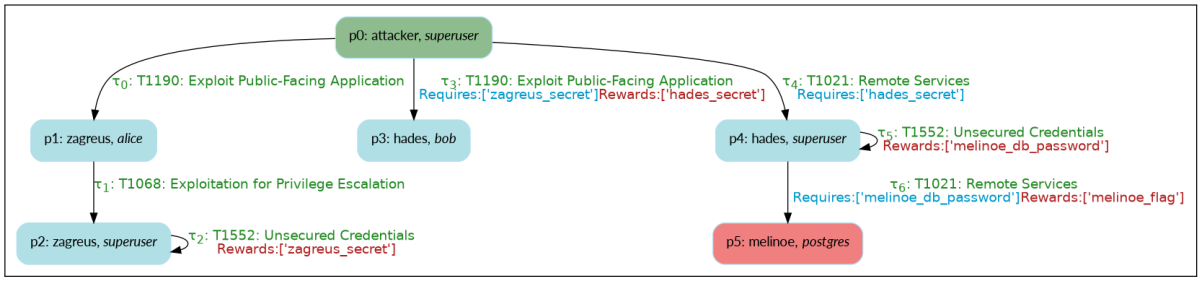

Figure 5.2 – CERBERE scenario

5.3.2 The attack scenario

This experiment was designed to be played by a group of participants with heterogeneous attacker skill levels and in a short period of time (two hours). In order to make it possible for every participant to make some progress while maintaining some depth for the more skilled participants we chose to make CERBERE a network of a few hosts with varied challenges, as opposed to a single but more resilient host.

5.3.3 The architecture

Each instance is generated from a template scenario. For this experiment, the CERBERE template scenario consists of three hosts named *zagreus*, *hades* and *melinoe* inside a local network. This scenario is meant to be played linearly: information stored on *zagreus* is required to progress to *hades* for instance. *Zagreus* and *hades* are both vulnerable to procedures linked to misconfigured websites, while *melinoe* hosts a PostgreSQL database. The goal of the attacker is to reach this database, which hosts a flag.

5.3.4 Variations in the attack scenario

A graphical representation of the overall scenario is available in the Figure 5.2. Attacker positions are represented as tuples of *(host, user)*, and transitions are labelled with the required MITRE ATT&CK technique. As previously mentioned, instances of this scenario will differ on a procedural level. Furthermore, in order to increase the range of difficulty offered by CERBERE, instances of the scenario will differ from each others. Indeed, each procedure may have several variations of different difficulties. For instance, in order to perform privilege escalation on a host, one instance might present a vulnerability

2. https://github.com/cmu-sei/GHOSTS

Table 5.1 – List of procedures available for each technique in CERBERE.

Technique	Procedures	Detection rules
$\tau_{0,3}$ = T1190	π_1: Website with command injection (easy) π_2: Website with command injection (medium) π_3: Django directory traversal rewarding ssh key	Access to Alice's home repository Chmod in images directory, commands executing .jpg files Not detectable through system logs
τ_1= T1068	π_4: Vulnerable sudo version (CVE-2019-14287) π_5: Vulnerable pkexec process	Command containing "-u#-1" Command executing the pkexec process directly
$\tau_{2,6}$ = T1552	π_6: Passwords in .bash_history π_7: Password in .txt file	Command accessing the .bash_history file Command accessing the important.txt file
$\tau_{4,6}$ = T1021	π_9: SSH Access from key π_{10}: SQL server rewarding a flag	SSHD user_login Command related to the psql executable

in its sudo package [79], while the other is vulnerable to the pwnkit exploit [80]. It is to be noted however that the overall path of attackers, *i.e.* which sessions they log in and how they progress through the network, remains consistent throughout all instances. An additional benefit is that participants cannot easily copycat actions of others. To implement these variations, we used the URSID automatic vulnerable architecture generation tool [12], which aims to convert high-level scenario descriptions into low level virtual host configuration files. Table 5.1 details available procedures for each technique. All credentials were also randomized for each instance. The second part of the table, detection rules, will be detailed later in Section 5.5.

Finally, we required a combination of network and system activities in order to get valuable logs for the blue team. This remark led to the design of an attack scenario where attackers gain shell accesses by exploiting website vulnerabilities but also use system vulnerabilities to get new credentials or roles.

5.3.5 Example of solution for red teamers

We describe globally the expected solution of this scenario: The red player should:

— Exploit the website hosted on *zagreus* by performing a command injection in a poorly configured search bar. Two levels of difficulty are available: some instances

limit how many characters could be inputted in the search bar, leading to attackers having to use the upload function of the website to execute their script instead of typing it directly. This exploit leads them to obtain a shell as user *Alice*.
— Perform a privilege escalation by exploiting one of the two vulnerabilities [79, 80] (randomly selected).
— Harvest credentials present in the *superuser* directory (which requires root access). Those credentials hint that they are linked to the host *hades*.
— Use those credentials to log as an admin on a Django website hosted on *hades*. Once this is achieved, acquire an SSH key stored in the /notes directory of the website, which hints that this key is linked to a user named *superuser*.
— Use this key to log in through SSH as *superuser*. The IP could be found by installing network scanning tools on the host *zagreus* using sudo privileges.
— Harvest credentials again, which hints that they are linked to the host *melinoe*.
— Use those credentials to access the PostgreSQL instance on *melinoe* and recover the flag, thus ending the scenario.

5.3.6 Exercise monitoring at multi-level

During the CERBERE exercise, the activities of the red teamers are monitored at different levels. First, all their actions at the browser level are recorded such as filling and sending forms, clicking buttons, checking cookies values. Second, we monitor their actions at operating system level, especially actions occurring in a shell session. Finally, network traffic is recorded in the virtual environment. Similarly, the actions of blue teamers are also recorded.

For this thesis, our focus is on two of those data sources. First, the network and system logs where we can find the attacks performed by the red teamers, constituting the CERBERE dataset. The network data was captured using Suricata while the system data was captured using Linux's auditd. With this, we can trust the data we give to blue teamers for investigation. Second, the investigation traces are our main focus since they will help us evaluate our recommendations as well as provide us with more data to evaluate MIMIR's Markov chain (see Chapter 4).

5.4 Experiments with CERBERE

This experiment was conducted at the EUR cyberschool [3] during their spring school, in April 2023. The experiment took place in two phases, each during roughly 2 hours. Our participants were of diverse profiles: graduate and post-graduate students as well as some researchers.

For the first phase of the exercise, there were 13 participants. They were given roughly one hour to succeed in executing as many exploits as possible due to logistics constraints. Each participant was given one instance to work on and instances were hosted on a single server of 36 cores and 80 GB of RAM. Each instance had five hosts in total: *zagreus*, *hades*, *melinoe*, the gateway host and the Suricata host.

It is interesting to note that red team exercise being more common than blue team exercises, we can observe more appeal for the first phase. For the blue team phase, 9 participants investigated the logs for one hour each as well.

5.4.1 Red team experiment

Since participants required remote access to the instances as they were all working on their personal computers, all hosts were made available through the Internet. Their entry point was however only accessible through specific URLs with custom port, in order to avoid the instances being compromised by attackers outside the experiment connecting to the port 80. This was achieved by carefully configuring network rules, a custom DNS server and a gateway host used by participants to be connected to the Internet.

The first phase of the exercise was the red team exercise. We requested that the participants come with Docker installed on their computer so that we can record actions coming from the Chrome web browser. Then each participant was given a URL pointing to a specific instance to attack. This way, the attacks performed by one participant could not disturb others. Participants started the exercise with very limited knowledge of the architecture they were attacking.

Participants were told that three hosts are involved in total. We gave them two URLs: one pointing to the website hosted by their specific *zagreus* host, and one for *hades*. They were instructed to first start with *zagreus*, and only move on to *hades* when they find direct clues about how to attack it. This was done due to our short time constraints for the experiment (2 hours), in order to avoid participants spending too much time trying

3. https://cyberschool.univ-rennes.fr/en/

to brute-force *hades*. Hints were occasionally provided to struggling participants in order to improve their learning experience.

5.4.2 Blue team experiment

Following the first phase of the exercise, we organized a second phase consisting of a blue team exercise. The participants gathered for this part of CERBERE did not participate in the red team experiment and have no knowledge of it. Each participant had the Malizen tool to investigate the logs generated by a participant of the red team exercise. Once the platform as well as the context of the exercise were presented to the participants, they were each given one hour to investigate and find as many steps of the scenario as possible.

The result of these investigations are stored under the shape of user action and associated context. Because this data is very specific to the graphical tool, it would be useless to publish the raw data associated to user's actions. Nine investigations were conducted, for a total of 2706 user actions recorded, allowing us to map exactly the progress of every analyst during the blue team part of the exercise. This exercise helps to evaluate the quality of the data recorded from a security standpoint.

While this second part of the experiment allowed us to introduce investigations to a cybersecurity exercise a component rarely included in this kind of work, it also served another purpose: evaluating the recommender systems from chapters 3 and 4. Contrary to the evaluations presented in the last chapters, this exercise aimed to another problem we have mentioned but not addressed: how can we integrate recommender systems into the investigative process of an analyst? The idea is to provide meaningful recommendations to analysts during their investigations without hindering them. To do so we have studied recommendation triggers as described in previous chapters, but recommendation frequency, presentation and wording as well. This falls more in the topic of user interfaces and experience, but is nonetheless important because however good a recommendation is, if it is presented at the wrong time and in the wrong form it becomes irrelevant.

5.4.3 Data quality

We manually investigated the data collected during the two exercises for all participants. The size of the red team data is approximatively 900Mo, or 69Mo per participant divided between the Suricata and auditd monitoring sources. Although, auditd logs rep-

Table 5.2 – Successful attacks (red team) and discoveries (blue team)

		Red team exploitation	**Blue team discovery**
Total nb players		13	7
Scenario step	**Mitre ATT&CK Technique**		
T_0	T1190	7	5
T_1	T1068	7	5
T_2	T1155	7	3
T_3	T1190	4	2
T_4	T1021	3	2
T_5	T1552	3	2
T_6	T1021	3	0

resent a majority of the data. Our goal is to be able to post-process the logs to quantify, how many steps of attacks have been achieved by a red teamer and how many steps of attacks have been correctly investigated by blue teamers. It is interesting to note that the investigations led by the blue teamers weigh only 1.1Mo in comparison.

When building our post-processing program, we observed the following points about the data. The nature of the log is heterogeneous and contains format variations. Despite monitoring systems or networks with only one probe each, we gather data that is very different and difficult to mix. The system data coming from auditd was notably complicated to explore and reuse as is due to a lack of standardization in the data. Some log lines were recorded over multiple lines, some not. Sometimes certain fields seemed to contain one nature of data but depending on the captured log line, this nature could change. However, all the important information to detect and understand the attack is present, it is even redundant and can be followed through the flow of the attack. In the future, we plan on using a different monitoring architecture, allowing us to produce our logs directly in a JSON format, offering the possibility of parsing them automatically more easily.

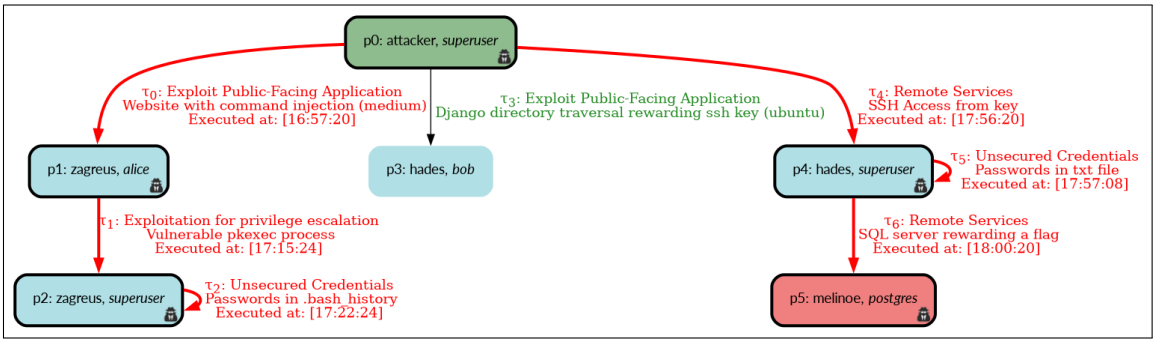

Figure 5.3 – Attack path through instance 6. Red transitions were detected using Zircolite rules. Transition 3 for attacker position (hades, *bob*) is not detectable through auditd logs, but the linear structure of the scenario ensures that it was compromised as well.

5.5 Results of the CERBERE experiment

5.5.1 Red team feedback

The first two columns of Table 5.2 shows which step and technique is related to the scenario for further technical references. The third column shows how many red teamers have managed to successfully conduct each step of the scenario. The fourth column shows how many blue teamers uncovered this step of the attack during the investigation.

Regarding the red teamers, it appears that the first exploit (finding a command injection within a website in order to create a reverse shell) was a roadblock for almost half of the participants. This does track with the expected difficulty of the scenario, as this website had a decent amount of functionalities and pages not relevant for the intended exploit, which could act as diversions for the participants. Once the attackers found the location of the command injection, they still had to properly initiate a reverse shell. Depending on the scenario variation, the difficulty of this step may vary because some variations required to upload and update the execution permission for their payload. The combination of (relative) difficulty and multistep nature of the exploit may thus explain this result, and it appears that attackers who managed to pass this step had no issues exploring the rest of the *zagreus* host.

The next roadblock appears to be accessing the second host, *hades*. Logging as *superuser* is not trivial because it required several steps. First attackers had to properly reuse the credentials acquired in *zagreus* on the website hosted on *hades*, successfully find an SSH key hidden in one of the pages, format it properly, then use it from their reverse shell acquired in the host *zagreus*. Time constraints also started ticking at this step for

participants who spent a long time on the first host. In particular, we observed in the logs that the volatile and primitive nature of the shell acquired on *zagreus* (through the combination of a reverse shell + exploit) meant attackers sometime accidentally closed it and had to redo the previous steps. Finally, attackers who managed this step appeared to have no trouble with the last step, as it simply required to gather credentials similarly to previous steps and to connect to a PostgreSQL service, whose IP is found by scanning the network.

We manually analysed the logs to understand the reasons behind each undetected step of an attack. 2 transitions (both for the initial website exploitation τ_0) were undetected as the attackers used unorthodox methods. For instance for procedure π_2, we expected attackers to upload a reverse-shell script using the website image upload function and execute it. However, one of the participants instead recognized that the host naturally had python installed and used that to directly download and execute a script instead. In these two cases the red teamer found an alternative method to the one we prepared for in order to exploit the vulnerability. If we refer to Table 5.1, it means the detection rules corresponding to these exploits were not triggered and the attackers evaded our monitoring. A most interesting result concerning the management of attack surface during the exercise. Nonetheless, in these cases, we assumed that the attacker achieved initial access anyway when the next steps were found in the logs: for instance acquiring credentials as *(zagreus, superuser)* requires the attacker to have accessed machine *zagreus* in the first place. The Figure 5.3 showcases one of those results for the instance number 6. Complete results for all participants are available in the dataset, as presented later in Section 5.6.

5.5.2 Blue team feedback

Regarding the blue teamers, the first thing we noted during the analysis of the data is that not all blue teamers reported on their results as was advised during the presentation of the tool. Unfortunately, two out of the nine participants did not register any findings and thus could not be included in these results. However, we get very interesting feedbacks from the seven remaining participants. First, we note that the web part of the scenario is the most commonly discovered step of the scenario. This can be explained because such attacks produce a lot more noise in the logs and because analysts are used to search for attackers coming from the Internet and are likely to search that part of the dataset first. Another interesting observation we made is that the participants discover groups of steps rather than a single one. Almost all the participants that have made a first discovery

Investigations	9
Recorded user actions	2706
Recommendations	729
Followed recommendations	607

Table 5.3 – Metrics about the blue team experiment

were able to discover directly related steps. Directly related steps are the main parts of the scenario: τ_0, τ_1, τ_2, τ_3, and τ_4, τ_5, τ_6. The difficulty lied in jumping from phase to phase since hopping from one host to another was not directly linked in the scenario. A final remark is that no participant was able to find the last exploit: the database access. This can be explained by the fact that it was done *legitimately* since the attackers had legitimate credentials at that point. Indeed, the last access to the database on the third machine (τ_6) is done using legitimate credentials found during the previous step. Added to the time constraint we put on the exercise, no participant reached this last discovery. This illustrates well the difficulty of the investigation job. When starting an investigation, even if there is knowledge of an attack, it is difficult to put the pieces together.

In terms of recommendations, the primary aim of this exercise, detailed in Section 4.7.2, was to gather data for evaluating MIMIR. Concurrently, we seized the opportunity to gather feedback from participants regarding the recommendations. Notably, participants generally welcomed recommendations, with KRAKEN receiving less discussion compared to MIMIR. This discrepancy might be attributed to KRAKEN's maturity as a recommender system and its seamless integration with Malizen.

5.5.3 Quantitative feedback

This section gives a few metrics regarding the blue team experiment in order to give an idea of the volumetry of the exercise, even though the recommendations' evaluation was not its main goal. The metrics are presented in Table 5.3. The most interesting number to note was the important number of recommendations compared to the number of user actions. Additionally, the percentage of followed recommendations is around 83%, showing a big interest to recommendations from the users: a promising result. We also gathered some qualitative feedback, hoping to explain these results further.

5.5.4 Qualitative feedback

The predominant feedback centered on the frequency of recommendations and, consequently, the triggers for recommendations. Our second hypothesis for recommendation triggers, implemented in MIMIR, involves making a recommendation only when a need is detected. Participants appreciated this approach; however, the higher frequency of recommendations in MIMIR compared to KRAKEN raised concerns. Unlike KRAKEN, where recommendations are solicited or triggered only when flagging occurs, MIMIR's approach resulted in more frequent recommendations. In future iterations, we need to devise a mechanism to control recommendation frequency without impeding the system's ability to provide timely recommendations when needed.

We also gathered feedback from the participants to the exercise and here are some information we have gathered. Timed notifications to present recommendations were greatly appreciated as analysts can choose to read and follow them at their will, while a change in screen or an unavoidable pop-up would have disrupted them. The availability of a history of recommendations as well as some explanations giving reasons why they were made was found to be an important feature as well as it allowed analysts to come back to recommendations and reconstruct their findings easily. On the other hand the frequency at which recommendations should appear is still unclear in the feedback. Depending on the contents of the recommendation and the moment at which it is given, and even on the analyst himself, recommendation frequency was judged differently. Finally, the wording of recommendations appeared to significantly change the confidence analysts have in them. Some recommendations were less easy to comprehend than others, namely the recommendations coming from MIMIR, and as a result we had to answer more questions about them than the KRAKEN recommendations. The wording is also a direct consequence of the level of abstraction of recommendations. It comforts us in our idea to hybridize the two recommender systems in the future.

5.6 CERBERE Dataset

With this contribution, we also provide the CERBERE dataset [4]. The dataset contains the raw logs from red teamers and the ground truth about these logs: we automatically labelled the logs to spot the lines that are related to successful attacks of the red teamers.

4. https://gitlab.inria.fr/cidre-public/cerbere-dataset/

Our goal is to make available a dataset that can be reused in new works related to attack detection. Additionally, the CERBERE dataset also contains the scenario for ensuring the reproducibility of the exercise using the URSID generation tool [12]. The monitoring configuration is also available, enabling future work to extract logs similarly to what has been done during our experiments. As presented in Table 5.1, we designed detection rules to decide if a procedure was successfully employed by an attacker. Depending on the procedure, a detection rule may be implemented at different log levels: system, network, browser. This section discusses the design of these rules to obtain ground truth labels.

5.6.1 System logs labelling

System logs created by auditd were extracted from every instance at the end of the exercise. Detection rules have been designed using Zircolite [88], a python SIGMA-based detection tool. These rules were manually crafted based on our personal experience and log results from playing the scenario and overall rely on the detection of specific commands or access to files corresponding to the evolution of the attacker in the scenario. For instance, exploiting CVE-2019-14287 [79] requires the attacker to use the command sudo with either the flags *-u#-1* or *-u#4294967295*. Since our auditd was configured to register this kind of command executed by users, we can detect the exploitation of this specific CVE with the following rule :

```
SELECT * FROM logs WHERE ((type = 'EXECVE' )) AND (a1 LIKE '-u#-1%' OR a1 LIKE '-u#4294967295')
```

This rule is similar to the ones present in professional security projects such as SIGMA [72]. Every possible procedure -except for the ones associated to τ_4 as it was not detectable through system logs.

These rules were then used on all extracted system logs and correlated with the description of each scenario in order to evaluate the participant's path through each of them. In total, out of the 15 instances planned for this exercise, 7 show at least a sign of compromise and 3 were entirely played out, meaning that the attacker reached the last host. Among these instances, a total of 28 transitions were detected using our rules, out of the 30 we expected to recognize based on attacker performance at the end of the exercise. The 2 missing were due to unexpected attacker actions as discussed before in Section 5.5.

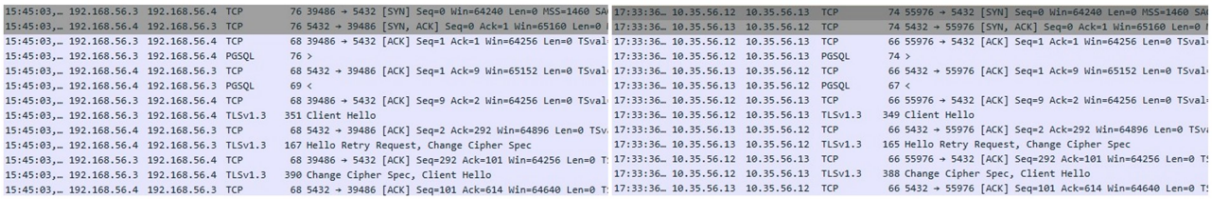

Figure 5.4 – Wireshark screenshot of melinoe pcap (left: baseline, right: instance 7) In instance 7: IP 10.35.56.13 is melinoe, IP 10.35.56.12 is Hades. In baseline: IP 192.168.56.4 is melinoe, IP 192.168.56.3 is Hades.

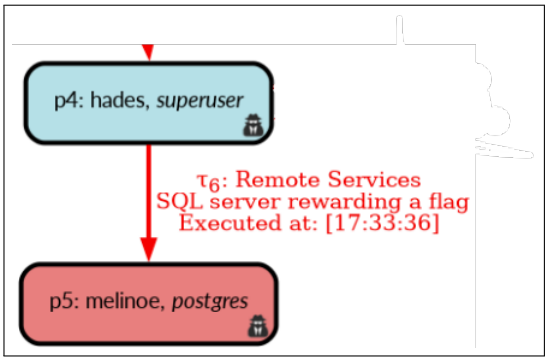

Figure 5.5 – Attack path extract from instance 7.

5.6.2 Network logs labelling

Network traffic recorded by Suricata was extracted in pcap and netflow format to give two detailed traffic levels. The pcap format provides a complete version of the traffic within packets. In contrast, the netflow structure offers a condensed version of traffic gathered in communications. These communications contain only meta-data information (without the payload). This format is studied in the intrusion detection field as it can reduce the quantity of data and eliminate the payload, which could be nonsensical (if encrypted, for example).

We provide ground-truth labels for each packet/communication. Each packet/communication that belongs to a successful attack is labelled as an attack, with the corresponding MITRE ATT&CK techniques (Table 5.1). The other packets/communications that do not participate in a successful attack in a scenario are considered as attempts. For each instance, the packets/communications of a successful attack are identified with the attack path's timestamps and a baseline. The timestamp comes from the previous labelling process of system logs (Section 5.6.1) and indicates where to look in the network logs. The baseline

is a pcap/netflow file that recorded a perfect player playing the instance, thus generating the minimal number of network lines for completing the scenario. The baseline helps us to review manually and decide if the found network logs should be labelled or not.

For example, let us focus on instance 7 and machine melinoe. We want to identify the access to the SQL server of the procedure T1021. The technique for procedure T1021 occurs at τ_6=15:45:03 for the baseline. For instance 7, as seen in Figure 5.5, we deduce from the previous analysis of system logs that it happened at 17:33:36. With these timestamps, we compare the two lines for pcap files as represented in Figure 5.4. We also can compare netflows as represented in Listing 2 and Listing 3. The attack label is finally manually added in the netflow of Figure 5.4 because ports and protocols obviously match, IP source and destination are the same and the length of the flow is comparable.

Listing 2 Example of netflow from baseline


```
{
"StartTime": "15:45:03.794101",
"Dur": 0.009609,
"Proto": "tcp",
"SrcAddr": "192.168.56.3",
"Sport": 39486,
"Dir": "->",
"DstAddr": "192.168.56.4",
"Dport": 5432,
"State": "RST",
"sTos": 0,
"dTos": 0,
"TotPkts": 30,
"TotBytes": 4014,
"SrcBytes": 1651,

}
```


5.6.3 Browser logs labelling

Attacker's actions were recorded by the browser monitoring tool into a Mongo database. We recorded the actions performed in the web browser by players and the visited web pages. Nevertheless, the labelling uses only the recorded actions. In this labelling process, some attack steps can be labelled and others cannot. Indeed, if the attacker setups a reverse shell, the new established connection is external to the web browser. As a consequence, only the actual command injection of step τ_0 can be labellised.

Similarly to network logs, we consider malicious the succeeding actions that lead to the exploitation of the command injection. The other lines of the logs were labelled as attempts because it corresponds to research of vulnerabilities, attempts of attacks or

Listing 3 Example of netflow for the instance 7


```
{
"StartTime": "17:33:36.832586",
"Dur": 0.02053,
"Proto": "tcp",
"SrcAddr": "10.35.56.12",
"Sport": 55976,
"Dir": "->",
"DstAddr": "10.35.56.13",
"Dport": 5432,
"State": "RST",
"sTos": 0,
"dTos": 0,
"TotPkts": 27,
"TotBytes": 4239,
"SrcBytes": 1875,
 "Label": "Attack",
 "Label description": "T1021"
}
```


documentation. Depending on the scenario, the command injection was either directly exploitable or needed to be prepared by uploading a binary and executing it. As a consequence, depending on the variation of scenario, one or two lines are labelled as a successful attack.

A labelled action is a JSON object that contains the type of action, the argument of the action and our label. For example in Listing 4, the JSON object shows the executing of an injection command using a call to the python interpreter. We can observe the script used by the attacker in the argument of the command.

5.7 Conclusion

Throughout this chapter we have presented the CERBERE cybersecurity exercise and its resulting dataset. CERBERE has the particularity of offering both a red team and a blue team exercise with a variety of difficulty levels. The scenario and architecture of this exercise are given and can be modified for generating a new playground. Three types of logs can be automatically extracted from a played exercise which is particularly interesting for intrusion detection evaluations.

Along with the exercise itself, we provide the dataset resulting from the first edition of the exercise. This dataset contains the logs of the 13 red teamers for the three types of captured logs. Additionally, we labelled the dataset after the exercise in order to spot the ground truth about line logs that are successful attacks and make it as usable as possible for future research, regardless of the application.

Listing 4 Example of web action for a command injection

```
{
"at_time_stamp": "1681310750.539",
"by_user": "649984c67f6052c4c05c9486",
"from_page": "649984c07f6052c4c05c9484",
"type": "SendHTTPForm",
"args": [
{
"name": "inputs",
"value": {
  "q": "hop | python3 -c \"import socket,os,pty;
        s=socket.socket(socket.AF_INET,socket.
        SOCK_STREAM);s.connect((\\\"7.tcp.eu.
        ngrok.io\\\",11162));os.dup2(s.fileno(),0);
        os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);
        pty.spawn(\\\"/bin/sh\\\")\""
}
},
{
"name": "xpath",
"value": "/html/body/main/div/div/div[2]/div[1]
          /div[2]/form"
}
],
"Label": "Attack"
}
```

This contribution, despite being less related to the topic of this thesis compared to the previous two chapters, was essential to the evaluation of MIMIR and to show how evaluations can be improved when enough data is available (**R3**). But most importantly, it is a contribution about a matter we think is essential for all areas of cybersecurity whether in research or in the industry: in order to build efficient and accurate tools, we need reliable data. We hope that CERBERE was the first iteration of a future platform where any kind of data can be generated for our research. Additionally, we were still able to gather some qualitative feedback about recommendations, that will help us improve both recommender systems when we prepare the next experiment.

In the introduction we discussed how we would not be able to address all the challenges brought by this problematic, and the one we wish to address next is the question of fake life. In order for the dataset to, not only represent realistic attacks but also be representative of what the logs of a real information system is, we need to implement normal behaviours. It will be our next challenge!

Conclusion

Never confuse education with intelligence, you can have a PhD and still be an idiot.

— Richard P. Feynman

The introductory chapters of the thesis highlighted challenges in the cybersecurity domain, specifically those faced by analysts. After presenting contextual elements such as analyst types, explored data, and the incident response process, we focused on the investigative phase, particularly exploring data with potential threats and attacks to discover. This contextualization revealed challenges stemming from the increasing volume of logs, inadequate helper tools, and a shortage of analysts due to the demanding nature of the work. Consequently, we delved into the literature to explore how the research community addresses these challenges.

We focused on recommender systems and their applications in cybersecurity, aiming to enhance analysts' efficiency by providing exploration recommendations. In our literature review, we identified a gap: while expert cybersecurity knowledge is utilized for technical tasks in various papers, it's not employed to complement analysts' expertise during investigations. As a response, we designed KRAKEN, a recommender system leveraging expert knowledge for recommendations. This initial system deepened our understanding of analysts' needs during investigations, leading to the design of MIMIR. This second recommender system captures user actions, detects intentions, infers the next probable intention, and suggests follow-up actions. Finally, our exploration of the literature highlighted a data problem, both in terms of availability and usefulness for evaluation. To address this, we introduced CERBERE, a red and blue team cybersecurity exercise, enabling us to generate data for evaluating and feeding our recommender systems.

Contribution made during this thesis

Our initial contribution, KRAKEN, is a knowledge-based recommender system that leverages the Mitre ATT&CK project and the Elastic Common Schema to associate attack

techniques with relevant log fields. It employs two scoring methods for recommendations. The first, Multi-Attribute Decision-Making (MADM), ranks techniques by conducting pairwise comparisons of attributes, computing global weights, and scoring techniques. The second method, Similarity scoring, capitalizes on the fixed nature of fields defined by ECS, calculating distances between fields to identify similar ones for exploration as recommendations. Evaluation results demonstrated relevant recommendations and introduced a novel approach for analysts to acquire knowledge for their investigations. However, challenges emerged in triggering recommendations at the appropriate time and for the right reasons, prompting an investigation into the diverse goals of analysts during their investigative processes.

Our second contribution, MIMIR, is a recommender system that utilizes past investigation traces to model analyst intentions, associating actions with specific intentions and identifying typical sequences of analyst actions using a Markov chain. During live investigations, MIMIR tracks analyst actions in real-time, detects intentions, infers the next most probable intention, and recommends relevant actions accordingly. We conducted theoretical experiments to assess the coverage of analyst behaviour captured in the Markov chain and a user evaluation to gauge the effectiveness of MIMIR's recommendations in real investigations. While the results of the user evaluation are promising, they highlight the need to refine the translation of recommendations into actionable steps for faster analyst adoption during investigations.

Our final contribution, CERBERE, addressed the challenge of evaluating recommender systems in the cybersecurity domain, which often faces issues related to data sensitivity and confidentiality. Given these challenges, we designed a red and blue team exercise played by 22 participants, that allows for the generation of adaptable and controlled datasets. CERBERE serves as a unique and modifiable blue and red team exercise, offering a scenario and architecture tailored to our evaluation requirements. The inclusion of a defensive component, the blue team phase, is unconventional and aims to encourage similar initiatives in the future. The first edition of the exercise was a success and allowed us, and three other PhD candidates from our team to gain data to work and further evaluate their prototypes. CERBERE also helped us quantitatively evaluate our recommender system MIMIR and get qualitative feedback about recommendations from both recommender systems contributed.

Perspectives & Future works

Even though three years of PhD feels like a long time when it starts, there is never enough time to try out all of our ideas. We have two kinds of perspectives for the work we have presented in this manuscript: direct improvements of our contributions and complementary contributions.

Our first recommendation for KRAKEN involves expanding its knowledge sources to include Cyber Threat Intelligence (CTI), which has witnessed significant growth in recent years. The structured nature of projects like STIX v2 [5] in structuring CTI knowledge bases makes it a valuable input for KRAKEN's recommendations. CTI reports, often referencing ATT&CK, provide detailed insights into attacks and attackers compared to the ATT&CK matrix. However, implementing CTI comes with challenges related to accessibility and logistics. Many companies producing CTI reports do not disclose them, and the available free feeds are often outdated and in less exploitable formats (STIX v1).

Secondly, addressing one of the limitations discussed in Chapter 4 for MIMIR, our focus should be on automating the pattern selection and detection process. A thorough exploration of different pattern mining algorithms would be beneficial. This improvement has the potential to reduce the cost of rebuilding the Markov chain and enhance the system's generalization by making the pattern mining method data-agnostic. However, recognizing relevant patterns poses a significant challenge, since the relevance of a pattern is established using cybersecurity knowledge.

Thirdly, for CERBERE, our key focus is on data-related enhancements. Increasing data collection can be achieved by conducting the exercise in new instances and improved scenarios, with the first one planned for 2024. Alternative approaches involve presenting the red team part as a honeypot. Additionally, we aim to structure the collected data more effectively for easier exploitation. Ongoing efforts include implementing a more structured monitoring system into the architecture and adopting user-friendly data formats in order for the resulting data to be as usable as possible for the research community.

A more significant prospect involves the development of a hybrid recommender system, combining features of both KRAKEN and MIMIR. KRAKEN, while not susceptible to the cold-start problem, struggles to adapt to new situations without regular maintenance of its knowledge base. Conversely, MIMIR exhibits flexibility by utilizing user actions for recommendations but is hindered by its abstract recommendations. A hybridized approach could

5. https://oasis-open.github.io/cti-documentation/stix/intro.html

mitigate these challenges. Our proposed solution involves integrating commonly executed user actions during investigations into KRAKEN's knowledge base, providing a foundational recommendation that remains available. This approach also allows for refinement through various methods such as direct feedback, Cyber Threat Intelligence (CTI), and discovery methods based on user actions. Furthermore, the hybridization method could enable us to prioritize either recommendation technique to better cater to the analyst's needs. For instance, a novice analyst might derive more benefit from a recommendation grounded in the behaviour and investigative methods of an experienced analyst. In contrast, a seasoned analyst could benefit more from a knowledge-based recommendation that validates their instincts.

Following this thesis, several underexplored topics emerge, promising significant contributions to the field. Firstly, time emerges as a crucial component in investigations, encompassing both the duration of an analyst's inquiry and the temporal framework represented in the logs. An intriguing prospect lies in the linkage of timestamps within the logs with killchain steps, allowing for the identification of attack campaigns and paving the way for new recommendations. This approach could involve suggesting previous or subsequent steps in the killchain, unveiling the progression of a user's intentions in alignment with the discovered killchain steps. Such a contribution holds the potential to substantially aid analysts in reconstructing attackers' pathways during investigations.

Furthermore, while this thesis concentrates on the investigative facet of incident response, we acknowledge that incident reporting and triage are well-explored domains, as discussed in Chapters 1 and 2. Yet, the topics of refined remediation and mitigation solutions remains a challenge. Exploring solutions like Mitre D3FEND (refer to Section 1.2.2.1) to establish connections between investigation findings and utilizing them for recommending mitigations and remediation represents a promising trajectory for advancement in this area.

BIBLIOGRAPHY

[1] Abdullah Abuhussein, Sajjan Shiva, and Frederick T Sheldon, « CSSR: cloud services security recommender », *in*: *2016 IEEE world congress on services (SERVICES)*, IEEE, 2016, pp. 48–55, DOI: https://doi.org/10.1109/services.2016.13.

[2] Gediminas Adomavicius and Alexander Tuzhilin, « Context-aware recommender systems », *in*: *Recommender systems handbook*, Springer, 2010, pp. 217–253, DOI: https://doi.org/10.1007/978-1-4899-7637-6_6.

[3] Gediminas Adomavicius and Alexander Tuzhilin, « Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions », *in*: *IEEE transactions on knowledge and data engineering* 17.*6* (2005), pp. 734–749, DOI: 10.1109/TKDE.2005.99.

[4] Alireza Afshari, Majid Mojahed, and Rosnah Mohd Yusuff, « Simple additive weighting approach to personnel selection problem », *in*: *International journal of innovation, management and technology* 1.*5* (2010), p. 511.

[5] Rabaa Alabdulrahman and Herna Viktor, « Catering for unique tastes: Targeting grey-sheep users recommender systems through one-class machine learning », *in*: *Expert Systems with Applications* 166 (2021), p. 114061, DOI: https://doi.org/10.1016/j.eswa.2020.114061.

[6] Carlos Ayala et al., « A Hybrid Recommender for Cybersecurity Based on Rating Approach », *in*: *Advances in Cybersecurity Management*, Springer, 2021, pp. 445–462, DOI: https://doi.org/10.1007/978-3-030-71381-2_20.

[7] Carlos Ayala et al., « A hybrid recommender system for cybersecurity based on a rating approach », *in*: *Advances in Security, Networks, and Internet of Things: Proceedings from SAM'20, ICWN'20, ICOMP'20, and ESCS'20*, Springer, 2021, pp. 397–409, DOI: https://doi.org/10.1007/978-3-030-71017-0_28.

[8] James Bennett, Stan Lanning, et al., « The netflix prize », *in*: *Proceedings of KDD cup and workshop*, vol. 2007, New York, 2007, p. 35.

[9] Michal Beran et al., « Exploratory Analysis of File System Metadata for Rapid Investigation of Security Incidents », *in*: *2020 IEEE Symposium on Visualization for Cyber Security (VizSec)* (2020), pp. 11–20, DOI: 10.1109/VizSec51108.2020.00008.

[10] Jacques Bertin and Marc Barbut, *Semiology of Graphics: Diagrams, Networks, Maps*, Paris: Ed. de l'EHESS, 2005, ISBN: 978-2-7132-2027-2.

[11] Pierre-Victor Besson et al., « CERBERE: Cybersecurity Exercise for Red and Blue team Entertainment, REproducibility », *in*: *CyberHunt 2023-6th Annual Workshop on Cyber Threat Intelligence and Hunting*, IEEE Computer Society, 2023.

[12] Pierre-Victor Besson et al., *URSID: Using formalism to Refine attack Scenarios for vulnerable Infrastructure Deployment*, 2023, arXiv: 2303.17373 [cs.CR].

[13] Romain Brisse et al., « KRAKEN: a knowledge-based recommender system for analysts, to kick exploration up a notch », *in*: *International Conference on Information Technology and Communications Security*, Springer, 2021, pp. 1–17, DOI: https://doi.org/10.1007/978-3-031-17510-7_1.

[14] Romain Brisse et al., « MIMIR: Modelling user Intentions with Markov chains for Intention Recommendations », *in*: *International Conference on Digital Forensics*, Springer, 2024.

[15] Robin Burke, « Hybrid recommender systems: Survey and experiments », *in*: *User modeling and user-adapted interaction* 12 (2002), pp. 331–370.

[16] Robin Burke, « Hybrid web recommender systems », *in*: *The adaptive web: methods and strategies of web personalization* (2007), pp. 377–408.

[17] Robin Burke, « Knowledge-based recommender systems », *in*: *Encyclopedia of library and information systems* 69.*Supplement 32* (2000), pp. 175–186.

[18] Fidel Cacheda et al., « Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems », *in*: *ACM Transactions on the Web (TWEB)* 5.*1* (2011), pp. 1–33, DOI: https://doi.org/10.1145/1921591.1921593.

[19] Canadian Institute for Cybersecurity, *Canadian Institute for Cybersecurity datasets*, 2023, URL: https://www.unb.ca/cic/datasets/index.html.

[20] Bram CM Cappers and Jarke J van Wijk, « SNAPS: Semantic network traffic analysis through projection and selection », *in*: *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, IEEE, 2015, pp. 1–8, DOI: 10.1109/VIZSEC.2015.7312768.

[21] Damien Cremilleux et al., « VEGAS: Visualizing, exploring and grouping alerts », *in*: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2016, pp. 1097–1100, ISBN: 978-1-5090-0223-8, DOI: 10.1109/NOMS.2016.7502968, (visited on 03/03/2020).

[22] Zhe Cui et al., « DataSite: Proactive visual data exploration with computation of insight-based recommendations », *in*: *Information Visualization* 18.*2* (2019), pp. 251–267, DOI: 10.1177/1473871618806555.

[23] Arthur de Moura Del Esposte et al., « A Collaboration Model to Recommend Network Security Alerts Based on the Mixed Hybrid Approach », *in*: ().

[24] Dorothy E. Denning, « An Intrusion-Detection Model », *in*: *IEEE Transactions on Software Engineering* SE-13 (1987), pp. 222–232, DOI: 10.1109/TSE.1987.232894.

[25] Alexandre Dey, « Datascience in support of cybersecurity operations: Adaptable, robust and explainable anomaly detection for security analysts », PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique, 2022.

[26] Gints Engelen, Vera Rimmer, and Wouter Joosen, « Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study », *in*: *SPW*, 2021, pp. 7–12, DOI: 10.1109/SPW53761.2021.00009.

[27] Leonardo Ferreira, Daniel Castro Silva, and Mikel Uriarte Itzazelaia, « Recommender Systems in Cybersecurity », *in*: *Knowledge and Information Systems* (2023), pp. 1–37, DOI: https://doi.org/10.1007/s10115-023-01906-6.

[28] Fabian Fischer and Daniel A. Keim, « NStreamAware: real-time visual analytics for data streams to enhance situational awareness », *in*: *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, ACM, 2014, pp. 65–72, ISBN: 978-1-4503-2826-5, DOI: 10.1145/2671491.2671495.

[29] Fethi Fkih, « Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison », *in*: *Journal of King Saud University-Computer and Information Sciences* 34.*9* (2022), pp. 7645–7669, DOI: https://doi.org/10.1016/j.jksuci.2021.09.014.

[30] Stefano Foresti and James Agutter, « Visalert: From idea to product », *in*: Springer, 2008, pp. 159–174, DOI: https://doi.org/10.1007/978-3-540-78243-8_11.

[31] Muriel Figueredo Franco, Bruno Rodrigues, and Burkhard Stiller, « MENTOR: the design and evaluation of a protection services recommender system », *in*: *2019 15th international conference on network and service management (CNSM)*, IEEE, 2019, pp. 1–7, DOI: https://doi.org/10.23919/cnsm46954.2019.9012686.

[32] Guido Grillenmeier, « Protecting Active Directory against modern threats », *in*: *Network Security* 2021.*11* (2021), pp. 15–17, DOI: https://doi.org/10.1016/s1353-4858(21)00132-x.

[33] Kevin Hu et al., « VizML: A Machine Learning Approach to Visualization Recommendation », *in*: *Proceedings of the 2019 Conference on Human Factors in Computing Systems (CHI)*, ACM, 2019, DOI: 10.1145/3290605.3300358.

[34] Philip Huff et al., « A recommender system for tracking vulnerabilities », *in*: *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–7, DOI: https://doi.org/10.1145/3465481.3470039.

[35] Christopher Humphries et al., « CORGI: Combination, Organization and Reconstruction through Graphical Interactions », *in*: *2014 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2014 IEEE Symposium on Visualization for Cyber Security (VizSec), IEEE, 2014, pp. 57–64, DOI: 10.1145/2671491.2671494, (visited on 03/03/2020).

[36] Christopher Humphries et al., « ELVIS: Extensible Log VISualization », *in*: *Proceedings of the Tenth Workshop on Visualization for Cyber Security*, VizSec 2013, Association for Computing Machinery, 2013, pp. 9–16, ISBN: 9781450321730, DOI: 10.1145/2517957.2517959.

[37] Martin Husák, « Towards a Data-Driven Recommender System for Handling Ransomware and Similar Incidents », *in*: *2021 IEEE International Conference on Intelligence and Security Informatics (ISI)*, IEEE, 2021, pp. 1–6, DOI: https://doi.org/10.1109/isi53945.2021.9624774.

[38] Martin Husák and Milan Čermák, « SoK: applications and challenges of using recommender systems in cybersecurity incident handling and response », *in*: *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–10, DOI: https://doi.org/10.1145/3538969.3538981.

[39] A Ishizaka, D Balkenborg, and T Kaplan, « Influence of aggregation and measurement scale on ranking a compromise alternative in AHP », *in*: *Journal of the Operational Research Society* 62.4 (2011), pp. 700–710, DOI: 10.1057/jors.2010.23.

[40] Dietmar Jannach et al., « Recommender Systems: An Introduction », *in*: *Recommender Systems: An Introduction* (Jan. 1, 2010), DOI: 10.1017/CBO9780511763113.

[41] Peter E Kaloroumakis and Michael J Smith, « Toward a knowledge graph of cybersecurity countermeasures », *in*: *The MITRE Corporation* 11 (2021).

[42] Linus Karlsson, Pegah Nikbakht Bideh, and Martin Hell, « A Recommender System for User-specific Vulnerability Scoring (full version) », *in*: (2019), DOI: https://doi.org/10.1007/978-3-030-41568-6_23.

[43] Parneet Kaur and Shivani Goel, « Shilling attack models in recommender system », *in*: *2016 International conference on inventive computation technologies (ICICT)*, vol. 2, IEEE, 2016, pp. 1–5, DOI: https://doi.org/10.1109/inventive.2016.7824865.

[44] Salman Khaliq, Zain Ul Abideen Tariq, and Ammar Masood, « Role of user and entity behavior analytics in detecting insider attacks », *in*: *2020 International Conference on Cyber Warfare and Security (ICCWS)*, IEEE, 2020, pp. 1–6, DOI: https://doi.org/10.1109/iccws48432.2020.9292394.

[45] Gang Kou et al., « Pairwise comparison matrix in multiple criteria decision making », *in*: *Technological and economic development of economy* 22.5 (2016), pp. 738–765, DOI: 10.3846/20294913.2016.1210694.

[46] Maxime Lanvin et al., « Errors in the CICIDS2017 Dataset and the Significant Differences in Detection Performances It Makes », *in*: *Risks and Security of Internet and Systems*, ed. by Slim Kallel et al., Cham: Springer Nature Switzerland, 2023, pp. 18–33, ISBN: 978-3-031-31108-6, DOI: https://doi.org/10.1007/978-3-031-31108-6_2.

[47] Philip A Legg et al., « Automated insider threat detection system using user and role-based profile assessment », *in*: *IEEE Systems Journal* 11.*2* (2015), pp. 503–512, DOI: https://doi.org/10.1109/jsyst.2015.2438442.

[48] Laetitia Leichtnam, « Detecting and visualizing anomalies in heterogeneous network events: Modeling events as graph structures and detecting communities and novelties with machine learning », PhD thesis, CentraleSupélec, 2020.

[49] Tianyi Li et al., « What data should I protect?: recommender and planning support for data security analysts », *in*: *Proceedings of the 24th International Conference on Intelligent User Interfaces*, IUI '19: 24th International Conference on Intelligent User Interfaces, ACM, 2019, pp. 286–297, DOI: 10.1145/3301275.3302294.

[50] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades, « Facing the cold start problem in recommender systems », *in*: *Expert systems with applications* 41.*4* (2014), pp. 2065–2073, DOI: https://doi.org/10.1016/j.eswa.2013.09.005.

[51] Lisa Liu et al., « Error Prevalence in NIDS datasets: A Case Study on CIC-IDS-2017 and CSE-CIC-IDS-2018 », *in*: *2022 IEEE Conference on Communications and Network Security (CNS)*, IEEE, 2022, pp. 254–262, DOI: https://doi.org/10.1109/cns56114.2022.9947235.

[52] Jock Mackinlay, « Automating the design of graphical presentations of relational information », *in*: *ACM Transactions On Graphics (Tog)* 5.*2* (1986), pp. 110–141, DOI: 10.1145/22949.22950.

[53] Jock Mackinlay, Pat Hanrahan, and Chris Stolte, « Show Me: Automatic Presentation for Visual Analysis », *in*: *IEEE Transactions on Visualization and Computer Graphics* 13.*6* (2007), pp. 1137–1144, DOI: 10.1109/TVCG.2007.70594.

[54] Serena McDonnell et al., « CyberBERT: a deep dynamic-state session-based recommender system for cyber threat recognition », *in*: *2021 IEEE aerospace conference (50100)*, IEEE, 2021, pp. 1–12, DOI: https://doi.org/10.1109/aero50100.2021.9438286.

[55] Stephen Moskal and Shanchieh Jay Yang, « Translating intrusion alerts to cyber-attack stages using pseudo-active transfer learning (PATRL) », *in*: *2021 IEEE conference on communications and network security (CNS)*, IEEE, 2021, pp. 110–118, DOI: https://doi.org/10.1109/cns53000.2021.9705037.

[56] Fitzroy D Nembhard, Marco M Carvalho, and Thomas C Eskridge, « Towards the application of recommender systems to secure coding », *in*: *EURASIP Journal on Information Security* 2019.*1* (2019), p. 9, DOI: https://doi.org/10.1186/s13635-019-0092-4.

[57] European Network and Information Security Agency - ENISA, *Good Practice Guide for Incident Management*, ENISA, 2010, URL: https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management (visited on 05/10/2023).

[58] Troy Nunnally et al., « *NAVSEC*: a recommender system for 3D network security visualizations », *in*: *Proceedings of the Tenth Workshop on Visualization for Cyber Security - 2013 IEEE Symposium on Visualization for Cyber Security (VizSec)*, ACM Press, 2013, ISBN: 978-1-4503-2173-0, DOI: 10.1145/2517957.2517963.

[59] Troy Nunnally et al., « P3D: A parallel 3D coordinate visualization for advanced network scans », *in*: *2013 IEEE International Conference on Communications (ICC)*, 2013, pp. 2052–2057, DOI: 10.1109/ICC.2013.6654828.

[60] Ranjit Panigrahi and Samarjeet Borah, « A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems », *in*: *International Journal of Engineering & Technology* 7.*3.24* (2018), pp. 479–482.

[61] Seung-Taek Park and Wei Chu, « Pairwise preference regression for cold-start recommendation », *in*: *Proceedings of the third ACM conference on Recommender systems - RecSys '09*, the third ACM conference, ACM, 2009, p. 21, ISBN: 978-1-60558-435-5, DOI: 10.1145/1639714.1639720.

[62] Aleksandra Pawlicka et al., « A systematic review of recommender systems and their applications in cybersecurity », *in*: *Sensors* 21.*15* (2021), p. 5248, DOI: 10.3390/s21155248.

[63] Vern Paxson, Scott Campbell, Jason Lee, et al., *Bro intrusion detection system*, tech. rep., Lawrence Berkeley National Laboratory, 2006.

[64] Nikolaos Polatidis and Christos K Georgiadis, « A dynamic multi-level collaborative filtering method for improved recommendations », *in*: *Computer Standards & Interfaces* 51 (2017), pp. 14–21, DOI: https://doi.org/10.1016/j.csi.2016.10.014.

[65] Nikolaos Polatidis et al., « From product recommendation to cyber-attack prediction: Generating attack graphs and predicting future attacks », *in*: *Evolving Systems* 11 (2020), pp. 479–490, DOI: https://doi.org/10.1007/s12530-018-9234-z.

[66] Nikolaos Polatidis et al., « Recommender systems meeting security: From product recommendation to cyber-attack prediction », *in*: *Engineering Applications of Neural Networks: 18th International Conference, EANN 2017, Athens, Greece, August 25–27, 2017, Proceedings*, Springer, 2017, pp. 508–519, DOI: https://doi.org/10.1007/978-3-319-65172-9_43.

[67] Nicolas Privault, *Understanding Markov chains: examples and applications*, Springer, 2018, DOI: https://doi.org/10.1007/978-981-13-0659-4.

[68] Pearl Pu, Li Chen, and Rong Hu, « Evaluating recommender systems from the user's perspective: survey of the state of the art », *in*: *User Modeling and User-Adapted Interaction* 22 (2012), pp. 317–355, DOI: https://doi.org/10.1007/s11257-011-9115-7.

[69] Francesco Ricci, Lior Rokach, and Bracha Shapira, « Introduction to recommender systems handbook », *in*: *Recommender systems handbook*, Springer, 2010, pp. 1–35, DOI: https://doi.org/10.1007/978-0-387-85820-3_1.

[70] Martin Roesch et al., « Snort: Lightweight intrusion detection for networks. », *in*: *Lisa*, vol. 99, 1, 1999, pp. 229–238.

[71] Rosa Romero-Gomez, Yacin Nadji, and Manos Antonakakis, « Towards designing effective visualizations for DNS-based network threat analysis », *in*: *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2017 IEEE Symposium on Visualization for Cyber Security (VizSec), IEEE, 2017, pp. 1–8, ISBN: 978-1-5386-2693-1, DOI: 10.1109/VIZSEC.2017.8062201.

[72] Florian Roth, *Sudo Privilege Escalation CVE-2019-14287*, 2019, URL: https://github.com/SigmaHQ/sigma/blob/master/rules/linux/process_creation/proc_creation_lnx_sudo_cve_2019_14287.yml (visited on 09/25/2023).

[73] Thomas L Saaty, « A scaling method for priorities in hierarchical structures », *in*: *Journal of mathematical psychology* 15.*3* (1977), pp. 234–281, DOI: 10.1016/0022-2496(77)90033-5.

[74] Carla Sayan, Salim Hariri, and George Ball, « Cyber security assistant: Design overview », *in*: *2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, IEEE, 2017, pp. 313–317, DOI: https://doi.org/10.1109/fas-w.2017.165.

[75] Z. Cliffe Schreuders et al., « Security Scenario Generator (SecGen): A Framework for Generating Randomly Vulnerable Rich-scenario VMs for Learning Computer Security and Hosting CTF Events », *in*: *2017 USENIX Workshop on Advances in Security Education (ASE 17)*, Vancouver, BC: USENIX Association, Aug. 2017, URL: https://www.usenix.org/conference/ase17/workshop-program/presentation/schreuders.

[76] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, « Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization », *in*: *ICISSP*, 2018, DOI: https://doi.org/10.5220/0006639801080116.

[77] Chris Simmons et al., « AVOIDIT: A cyber attack taxonomy », *in*: *University of Memphis, Technical Report CS-09-003* (2009).

[78] Fabio Soldo, Anh Le, and Athina Markopoulou, « Predictive blacklisting as an implicit recommendation system », *in*: *2010 Proceedings IEEE INFOCOM*, IEEE, 2010, pp. 1–9, DOI: https://doi.org/10.1109/infcom.2010.5461982.

[79] National Institute of Standards and Technology, *CVE-2019-14287*, 2019, URL: https://nvd.nist.gov/vuln/detail/CVE-2019-14287 (visited on 09/13/2023).

[80] National Institute of Standards and Technology, *CVE-2021-4034*, 2021, URL: https://nvd.nist.gov/vuln/detail/cve-2021-4034 (visited on 09/13/2023).

[81] Cliff Stoll, *The cuckoo's egg: tracking a spy through the maze of computer espionage*, Simon and Schuster, 2005.

[82] Zarrin Tasnim Sworna, Chadni Islam, and Muhammad Ali Babar, « Apiro: A framework for automated security tools api recommendation », *in*: *ACM Transactions on Software Engineering and Methodology* 32.*1* (2023), pp. 1–42, DOI: https://doi.org/10.1145/3512768.

[83] Roberto Theron et al., « Network-wide intrusion detection supported by multivariate analysis and interactive visualization », *in*: *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*, IEEE, 2017, pp. 1–8, ISBN: 978-1-5386-2693-1, DOI: 10.1109/VIZSEC.2017.8062198.

[84] Rafael Uetz et al., « Reproducible and adaptable log data generation for sound cybersecurity experiments », *in*: *Annual Computer Security Applications Conference*, 2021, pp. 690–705, DOI: https://doi.org/10.1145/3485832.3488020.

[85] Alex Ulmer, David Sessler, and Jörn Kohlhammer, « NetCapVis: Web-based Progressive Visual Analytics for Network Packet Captures », *in*: *2019 IEEE Symposium on Visualization for Cyber Security (VizSec)*, 2019, pp. 1–10, DOI: 10.1109/VizSec48167.2019.9161633.

[86] Manasi Vartak et al., « SEEDB: automatically generating query visualizations », *in*: *Proceedings of the VLDB Endowment* 7 (2014), pp. 1581–1584, DOI: 10.14778/2733004.2733035.

[87] Alex Vieane et al., « Addressing human factors gaps in cyber defense », *in*: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 60, 1, SAGE Publications Sage CA: Los Angeles, CA, 2016, pp. 770–773, DOI: https://doi.org/10.1177/1541931213601176.

[88] wagga40, *Zircolite*, 2021, URL: https://github.com/wagga40/Zircolite (visited on 09/15/2023).

[89] Leland Wilkinson, *The grammar of graphics*, Springer, 2012.

[90] Kanit Wongsuphasawat et al., « Voyager 2: Augmenting Visual Analysis with Partial View Specifications », *in*: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 2017, pp. 2648–2659, ISBN: 9781450346559, DOI: 10.1145/3025453.3025768.

[91] Kanit Wongsuphasawat et al., « Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations », *in*: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 649–658, DOI: 10.1109/TVCG.2015.2467191.

[92] Chen Zhong et al., « A cyber security data triage operation retrieval system », *in*: *Computers & Security* 76 (2018), pp. 12–31, DOI: https://doi.org/10.1016/j.cose.2018.02.011.

[93] Chen Zhong et al., « Studying Analysts' Data Triage Operations in Cyber Defense Situational Analysis », *in*: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Journal Abbreviation: Lecture Notes in Computer Science (including subseries Lecture

Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), July 7, 2017, pp. 128–169, ISBN: 978-3-319-61151-8, DOI: 10.1007/978-3-319-61152-5_6.

[94] Carson Zimmerman, *The strategies of a world-class cybersecurity operations center*, The MITRE Corporation, 2014.

List of Figures

List of Tables

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

CentraleSupélec

Titre : Recommandations d'exploration pour l'investigation d'incidents de sécurité

Mot clés : cybersécurité, investigation, systèmes de recommandation, réponse à incident

Résumé : Ces dernières années, les analystes en cybersécurité doivent faire face à des obstacles grandissants dans leur activité. Non seulement les données à investiguer sont hétérogènes, contiennent trop de dimensions, ou sont simplement incomplètes, mais aussi les attaques et attaquants se multiplient, créant une pénurie d'experts du domaine. De nombreux outils visent à soulager leur charge de travail, notamment pendant la réponse à incident, mais ce n'est pas suffisant.

Les travaux de la thèse réalisée par Romain Brisse consistent à trouver des méthodes pour faciliter la phase investigative de la réponse à incident. Ils se focalisent notamment sur l'utilisation de systèmes de recommandation proposant des chemins d'exploration dans les journaux d'événements à investiguer. Les contributions de la thèse comportent deux systèmes de recommandation. Le premier, KRAKEN, repose sur l'utilisation de connaissances expertes de la communauté cyber, permettant de reconnaître l'attaque observée dans les données et de recommander les champs les plus pertinents à explorer. La seconde contribution s'inscrit dans une certaine continuité avec la première, car ayant remarqué la difficulté pour un système de recommandation à comprendre l'intention d'un analyste, un deuxième système de recommandation (MIMIR) se base sur une modélisation de ces intentions pendant une investigation afin de recommander la marche à suivre dans la suite de celle-ci. Finalement, s'intéressant aux problématiques d'évaluation et de manque de données cyber, une dernière contribution est faite sous la forme d'un exercice (CERBERE) pendant lequel des données permettant non seulement l'évaluation mais aussi l'amélioration des systèmes de recommandation sont générées et investiguées par les participants.

Title: Exploration recommendations for the investigation of security incidents

Keywords: cybersecurity, investigation, recommender systems, incident response

Abstract: In recent years, cybersecurity analysts have encountered growing challenges in their field. Not only are the data they investigate heterogeneous, multidimensional or simply incomplete, but also the number of attacks and attackers is increasing, leading to a shortage of experts in the domain. While numerous tools aim to alleviate their workload, particularly during incident response, they fall short.

Romain Brisse's thesis work focuses on developing methods to facilitate the investigative phase of incident response, specifically leveraging recommendation systems that propose exploration paths in event logs. The thesis contributions include two recommendation systems. The first, KRAKEN, relies on expert knowledge from the cyber community to recognize attacks in data and recommend the most relevant fields to explore in order to identify them. The second contribution aligns with the first, as it addresses the challenge of recommendation systems understanding an an-

alyst's intent. The second system, MIMIR, is based on modelling these intentions during an investigation to suggest the subsequent investigation steps. Finally, addressing evaluation challenges and the lack of cyber data in the field, a final contribution takes the form of an exercise (CERBERE) during which data for the evaluation and improvement of recommendation systems are generated and investigated by participants.