UNIVERSITY OF PASSAU

DOCTORAL THESIS

User-centered intrusion detection using heterogeneous data

Author:
Mathieu Garchery

Advisors:
Prof. Dr. Michael Granitzer
Ulrich-Anton Häring

A thesis submitted in partial fulfillment of the requirements for the degree of doctor of natural sciences

in the

Faculty of Computer Science and Mathematics

Passau, December 2020

Abstract

With the frequency and impact of data breaches raising, it has become essential for organizations to automate intrusion detection via machine learning solutions. This generally comes with numerous challenges, among others high class imbalance, changing target concepts and difficulties to conduct sound evaluation. In this thesis, we adopt a user-centered anomaly detection perspective to address selected challenges of intrusion detection, through a real-world use case in the identity and access management (IAM) domain. In addition to the previous challenges, salient properties of this particular problem are high relevance of categorical data, limited feature availability and total absence of ground truth.

First, we ask how to apply anomaly detection to IAM audit logs containing a restricted set of mixed (i.e. numeric and categorical) attributes. Then, we inquire how anomalous user behavior can be separated from normality, and this separation evaluated without ground truth. Finally, we examine how the lack of audit data can be alleviated in two complementary settings. On the one hand, we ask how to cope with users without relevant activity history ("cold start" problem). On the other hand, we seek how to extend audit data collection with heterogeneous attributes (i.e. categorical, graph and text) to improve insider threat detection.

After aggregating IAM audit data into sessions, we introduce and compare general anomaly detection methods for mixed data to a user identification approach, designed to learn the distinction between normal and malicious user behavior. We find that user identification outperforms general anomaly detection and is effective against masquerades. An additional clustering step allows to reduce false positives among similar users. However, user identification is not effective against insider threats. Furthermore, results suggest that the current scope of our audit data collection should be extended.

In order to tackle the "cold start" problem, we adopt a zero-shot learning approach. Focusing on the CERT insider threat use case, we extend an intrusion detection system by integrating user relations to organizational entities (like assignments to projects or teams) in order to better estimate user behavior and improve intrusion detection performance. Results show that this approach is effective in two realistic scenarios.

Finally, to support additional sources of audit data for insider threat detection, we propose a method representing audit events as graph edges with heterogeneous attributes. By performing detection at fine-grained level, this approach advantageously improves anomaly traceability while reducing the need for aggregation and feature engineering. Our results show that this method is effective to find intrusions in authentication and email logs.

Overall, our work suggests that masquerades and insider threats call for different detection methods. For masquerades, user identification is a promising approach. To find malicious insiders, graph features representing user context and relations to other entities can be informative. This opens the door for tighter coupling of intrusion detection with user identities, roles and privileges used in IAM solutions.

Acknowledgements

First, I would like to thank my supervisors for their continuous guidance which made this dissertation possible. Michael, I am deeply grateful for insightful feedback which kept me on track during this interesting yet challenging research experience. Your contagious scientific curiosity has been a great motivator to complete this thesis.

Ulli, many thanks for your daily support, your guidance from a business perspective and especially for the great deal of freedom I enjoyed in my research investigations. Your open-minded and practical attitude made our collaboration pleasant at any time. Unfortunately, you could not see this research project completed. You will remain forever in my memory as a very inspiring and helpful friend.

Furthermore, I want to greatly thank Harald, Morwenna, Axelle, Ingrid as well as the rest of the chair team for constant organizational and administrative support since the beginning of my thesis. I thank Konstantin for his support during the first year spent choosing between research directions. Thanks to Roland, whose engagement made this collaboration between Atos and the University of Passau possible. Moreover, I wish to express my gratitude towards Fazzi, Amogha, Cristina and Saber who contributed to my research investigations during their Master thesis.

Next, I would like to thank colleagues and friends at Atos, university and Irixys for their valuable advice. It has always been a pleasure to meet and exchange with you in a convivial atmosphere.

Finally, I want to express my gratitude to Nadja for supporting me throughout ups and downs of this personal journey.

Contents

List of Figures

List of Tables

List of Abbreviations

ADSAGE	**A**nomaly **D**etection in **S**equences of **A**ttributed **G**raph **E**dges (see 6.1.2)
AUPRC	**A**rea **U**nder **P**recision-**R**ecall **C**urve
CERT (dataset)	public insider threat dataset released by Carnegie-Mellon University's **C**omputer **E**mergency **R**esponse **T**eam (see 3.2)
CR-k	**C**umulative **R**ecall at budget **k** (see 6.1.3)
FFNN	**F**eed-**f**orward **N**eural **N**etwork
IAM	**I**dentity and **A**ccess **M**anagement
IDS	**I**ntrusion **D**etection **S**ystem
LANL (dataset)	cyber-security dataset from **L**os **A**lamos **N**ational **L**aboratory (see 3.3)
LDAP	**L**ightweight **D**irectory **A**ccess **P**rotocol
RNN	**R**ecurrent **N**eural **N**etwork
(ROC) AUC	(**R**eceiver **O**perating **C**haracteristic) **A**rea **U**nder **C**urve
RQ	**R**esearch **Q**uestion
SIEM	**S**ecurity **I**nformation and **E**vent **M**anagement
UEBA	**U**ser and **E**ntity **B**ehavior **A**nalytics
UCIBID	**U**ser **C**luster **I**dentification **B**ased **I**ntrusion **D**etection (see 4.2.2)
UIBID	**U**ser **I**dentification **B**ased **I**ntrusion **D**etection (see 4.2.2)

Chapter 1

Introduction

1.1 Motivation and scope

The growing threat of data breaches In our ever more digitalized world, data breaches pose a rising threat to public and private organizations. According to the Cost of Data Breach study by IBM [1], the global average cost of a data breach has reached $3.94 million in 2019. In addition to their high financial impact, data breaches are increasingly frequent: the likelihood for an organization to experience a leak of at least 10 000 data records in the next two years reaches 29.6%, a 31% increase compared to 2014. Moreover, the IBM study only considers medium-sized breaches (up to 100 000 data records) and does not encompass mega breaches, like the Equifax incident (2017). In this latter case, sensitive personal data such as social security numbers, addresses and birth dates from over 147 million people were stolen from the credit risk assessment company. The incident had a significant financial impact on Equifax, whose stock price lost approximately one third of its value in the weeks following the announcement; recently the company agreed to pay *at least* $575 million to settle subsequent lawsuits [2]. Despite the far-reaching consequences, many organizations are not prepared against data breaches. As suggested in the IBM study, their containment takes as long as 279 days on average, even reaching 314 days for malicious incidents, which represent around a half of all breaches.

Intrusion detection systems (IDS) Therefore, robust protection systems are becoming vital to mitigate the risk of data breaches and other intrusions in digital infrastructure. Due to the increasing data volumes produced by organizations, automated solutions are required, as the effort needed for exhaustive monitoring of an organization exceeds by far available human resources. This is where intrusion detection systems come in. The American National Institute of Standards and Technology (NIST) defines an intrusion detection system as "software that automates the intrusion detection process, [i.e.] monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices" [3]. In view of the large amounts of events to monitor, an effective IDS must be capable of finding a needle in a haystack, whenever such needle is present. Human resources are usually still allocated to a security operations center, in charge of receiving the alerts and taking counteractions to contain the threat. In this thesis, we adopt this perspective as well.

Traditionally, intrusion detection systems can be classified into two broad categories: signature-based and anomaly-based IDS [4]. Signature-based IDS rely on known patterns of malicious activity (so-called signatures) to be detected. This approach is very effective (in terms of detection and false positive rate), as long as the signatures are up-to-date. However, signature-based methods are helpless against

unseen, i.e. zero-day threats. Nevertheless, these intrusions can be detected with anomaly-based IDS. Their principle is to build profiles of normal behavior, and assume significant deviations from normality to represent malicious activities. This last assumption is critical for the performance of anomaly-based IDS: if anomalies are not malicious, false alerts will be raised. On the contrary, malicious activities falling within the scope of normal behavior will lead to false negatives, i.e. not raising alerts when necessary. Methods presented in this thesis fall under anomaly-based intrusion detection.

Furthermore, intrusion detection systems can be classified based on where they operate. Host-based intrusion detection systems run on users' computer where activity is inspected locally. On the contrary, network-based intrusion detection systems rely on analyzing traffic between hosts of an organization's network to detect attacks. As both approaches can be complementary, the distinction between host-based and network-based is not always crisp; hybrid systems have been developed as well [5]. In this thesis, we adopt a user-centered perspective which exceeds this distinction. Indeed we try to leverage as many audit data sources as available to describe user activities, regardless of whether these are performed locally or remotely. By doing so, we assume that behavior profiles required for anomaly-based intrusion are best built at user level.

Relation to SIEM and UEBA While this thesis focuses on intrusion detection, security information and event management (SIEM) as well as user and entity behavior analytics (UEBA) are two closely related concepts. SIEM designates the process of monitoring security events across information systems, in order to generate and respond to alerts [6]. Thus SIEM encompasses intrusion detection, which should be integrated in this process. UEBA corresponds to anomaly-based profiling of users and entities (i.e. diverse devices connected to an IT system) in order to detect fraudulent behavior from an internal perspective [7]. Thus, methods presented in this thesis are highly relevant to UEBA.

Intrusion types Various types of intrusions have been described in literature, including network-based attacks, malware, exploitation of hardware and software vulnerabilities [5]. The landscape of real-world threats is now so vast that specialized knowledge bases have emerged to keep trace of possible attack techniques [8]. Given the diversity of attack patterns, it would be naive to think a one-fits-all detection solution can be found. In order to provide a clear boundary to this thesis, we decide to focus on two types of attacks, for which modeling user behavior is key to applying anomaly-based detection: masquerades and insider threats. Masquerade refers to credentials being used by someone else as the legitimate owner, usually without his consent. In this case, credentials are often stolen by the attacker, who can be internal or external to the organization. On the contrary, insider threats designate illegitimate and potentially harmful activities conducted by rightfully authorized (i.e. internal) users.

Machine learning for intrusion detection As the distinction between normal and anomalous behavior is often non-trivial and difficult to formalize, machine learning can be used to learn these two concepts using examples (see chapter 2 for a more detailed literature review). Machine learning methods can be classically divided into supervised and unsupervised approaches, depending on ground truth availability.

In the intrusion detection domain, the latter are usually preferred for several reasons. First, manual annotation requires prohibitive human effort, due to the large volume and variety of audit logs. Second, class imbalance (i.e. scarceness of intrusions compared to normal samples) aggravates this problem: in order to find only one anomalous example, a human annotator would have to go through thousands if not millions of normal observations. Finally, providing a clear definition of what constitutes an intrusion is not always straightforward: it can depend on the operational context [9] and can evolve over time (i.e. concept drift [10]). In this thesis, we address a real-world intrusion detection problem where ground truth is not available (more details in section 1.2), thus we use unsupervised learning techniques.

Whether addressed through supervised or unsupervised learning, intrusion detection is generally regarded as a binary classification (in the former setting) or an anomaly detection task (in the latter). In any case, observations are to be assigned a binary label (normal or malicious). However, in this thesis we adopt a different perspective and focus on anomaly ranking. Instead of assigning labels, our aim is to provide scores quantifying how anomalous audit records are. This presents several advantages compared to the classification perspective. First, it is more general, as one can always revert back to binary decisions by applying some threshold on the anomaly scores. Second, ranking allows to choose a threshold dependent on the operational environment. Lastly, anomaly scores can be used for prioritization of alerts by security analysts.

Heterogeneous data Finally, it is obvious that intrusion detection systems are to be applied on real-world audit data, which is often heterogeneous. By heterogeneous data we refer to data with attributes (or features, following the machine learning terminology) of different types: numeric, ordinal, categorical, text, image, sound, video, graph... It appears that the majority of existing methods for anomaly detection in general, and intrusion detection in particular, are designed for numeric data only. As a consequence, other data types are often simply discarded when using these methods. This is not desirable: by ignoring some features, we may loose useful information or context relevant to our problem. Thus we argue that heterogeneous data should not be discarded without further analysis. In this thesis, we use the terms *mixed data* for the combination of numeric and categorical features, and *heterogeneous data* for any combination of several feature types.

1.2 Real-world intrusion detection use case

The research work presented in this thesis addresses particular aspects of a real-world intrusion detection use case proposed by the company Atos, and which constitutes the background of our study. The long term goal of this use case consists in laying foundations for the development of an intrusion detection module within an identity and access management framework. In the following, we first provide some details about this application context. Then we proceed to outline key characteristics of this real-world use case.

1.2.1 Application context: identity and access management

To put it very broadly, the goal of identity and access management is to ensure that the right persons access the right resources at the right time and for the right

reasons. To achieve this goal, IAM classically relies on three main functions: authentication, authorization and audit. An access to a protected resource consists of an authentication, where the identity claimed by a user is verified followed by an authorization, where his right to access said resource is verified. IAM solutions notably allow organizations to manage large number of users along with their credentials and roles required for authentication and authorization. The third main component of IAM – audit – is mostly concerned with a posteriori monitoring and analysis of access logs. Audit is critical to guarantee the proper functioning of IAM solutions, as well as to comply with legal regulations.

Adding intrusion detection capabilities to IAM systems appears as a natural extension for several reasons. First, traditional credentials based on *something you know* (e.g. password) or *something you have* (e.g. smart card) are subject to theft and misuse. Thus methods based on *something you are* can provide an additional authentication layer to reduce risk of illegitimate access. Anomaly-based intrusion detection systems fall into this last category. Secondly, IAM systems already collect user activity traces for audit purposes, providing a data source to build anomaly-based intrusion detection models. Last but not least, integrating intrusion detection within IAM systems allows for automatic counteraction when an intrusion is detected, i.e. denying access and reporting to the security operations center. This is a crucial first step for timely containment of intrusions.

1.2.2 Use case particularities

In order to evaluate possible intrusion detection solutions, a real-world dataset is provided by Atos. It consists of anonymized access logs from the company's web single sign-on portal for employees. Through this gateway, numerous online business applications and resources are available, such as the employee portal (for time sheets, leave and travel requests, etc.), the company's internal social network, HR and support pages, and many more. These audit logs are in form of authentication and authorization events; more details can be found in section 3.1.

This dataset has three particular characteristics which we address in our research. First, it contains mixed (i.e. numeric and categorical) features; other data types could be added later. Secondly, it was not collected for intrusion detection purposes and only has a limited number of features to describe user activity. Thirdly, ground truth is completely absent. Due to the large data size, it is very likely that the Atos audit logs contain at least some anomalies, but there are no labels of normal or abnormal behavior. At the time where this work has been conducted, no expert knowledge was available to annotate particular data samples either: under these conditions, validation of any detection result is challenging.

1.3 Research objectives

Our first goal is to address particular aspects of the real-world intrusion detection use case described previously: support of mixed data, limited features and total absence of ground truth.

However to this aim deploying "off-the-shelf" solutions is impossible for several reasons. Some are general challenges of intrusion detection: for instance, in this application domain there is no standard feature set (unlike in image classification or credit card fraud detection). Moreover, definitions of what is to be considered an intrusion or not largely depends on the organization's security policy, and therefore on

the application environment. These two points make intrusion detection solutions unique and hinder reusing existing methods.

In addition to these general issues, properties of the real-world use case we study bring specific requirements. Regarding the support of mixed data, most anomaly-based intrusion detection methods work only on numeric data, hence using them implies discarding categorical attributes. Concerning features available to describe user activity, they are limited because the real-world audit logs we use were not collected specifically for intrusion detection purposes. Thus substantial preprocessing and feature engineering effort is required. This motivates us to formulate the following research question:

RQ. 1: How to apply anomaly detection methods to real-world authentication and authorization audit logs containing mixed data and limited features?

Regarding the absence of ground truth, the challenge is not only that we have to restrict ourselves to unsupervised learning methods, but also that detection results should be evaluated without further gold standard. This constitutes our second research question:

RQ. 2: In total absence of ground truth, how to learn the distinction between normal and anomalous user behavior and evaluate the results?

We address the absence of ground truth through user identification (see chapter 4). This approach is effective against masquerades but not insider threats; therefore we lend our full attention to the latter in the remaining chapters of this thesis. Additionally, in real-world settings – including ours – the lack of audit data sources available to describe and characterize user behavior can represent a significant challenge. This is an important shortcoming affecting anomaly-based intrusion detection methods in general, as they rely on audit data to build profiles of normal user behavior.

The "cold start" problem represents a first variant of this challenge: it corresponds to relevant activity data being inherently unavailable for particular users (e.g. new recruits of an organization who do not have an activity history yet). Our third research question addresses this setting:

RQ. 3: How to address the "cold start" problem (i.e. absence of activity data) in anomaly-based intrusion detection?

The cold start problem typically concerns a subset of all users within an organization. However, the lack of audit data can be more pervasive, in particular when the scope of collected audit data is just not sufficient to detect intrusions effectively. This raises the question of how to extend data collection in the future. As discussed in related work, we have identified that graph and text features are commonly neglected in intrusion detection, which motivates us to investigate how they could be better integrated in this domain:

RQ. 4: Which heterogeneous data features can be useful to detect intrusions, in particular insider threats, and how to leverage them?

1.4 Contributions

In response to the previous research questions, this thesis brings following contributions. Regarding **RQ. 1**, we show that authentication and authorization events from our real-world dataset can be aggregated into user sessions, which represent consistent and semantically meaningful timespans of user activity. In order to characterize these sessions, we extract diverse features from their corresponding audit log events. As these sessions contain mixed (numeric and categorical) attributes, we propose three new unsupervised anomaly detection methods supporting this type of heterogeneous data. We benchmark these methods on public outlier detection datasets, before applying them to our audit sessions.

However, in our real-world intrusion detection use case, ground truth is completely unavailable, including for evaluation. To address **RQ. 2** we frame intrusion detection as user identification task, in order to investigate whether users are distinguishable based on their sessions. Our user identification based intrusion detection approach is usable with any classifier and outperforms previous anomaly detection baselines for masquerade detection on our real-world dataset. Still, this approach yields many false positives when users have similar behavior. Using clustering, we are able to improve masquerade detection by greatly reducing false positives. However, an evaluation on an insider threat dataset shows that these intrusions must be addressed differently. Additionally, a feature importance study of the user identification classifier shows that most session features are not informative to distinguish users from each other in our real-world dataset.

These findings motivate us to address the insufficiency of audit data required to characterize user behavior in anomaly-based intrusion detection. Focusing on insider threats, we address two variants of this problem. In both cases, our angle of attack consists in leveraging heterogeneous audit data sources.

Regarding **RQ. 3**, we show how graph features can be used to tackle the "cold start" problem in intrusion detection systems: whenever historical data is too scarce, anomaly-based IDS will have trouble defining normal user behavior. We show that this can be addressed through zero-shot learning, a technique where semantic representations of classes compensates the lack of examples thereof. In particular, we extend an existing intrusion detection system, in which we integrate information from the organization's structure in form of vector representations (graph embeddings). This provides more context about users. We compare our extended system against the existing one in two scenarios where the cold start problem arises: a. when a new user joins the organization (i.e. history unavailable) and b. when a user changes working project (i.e. history present but obsolete). Our results show that taking into account the organization structure graph significantly improves insider threat detection in both scenarios.

Concerning **RQ. 4**, we examine how different heterogeneous attributes can be leveraged and assess precisely which ones are useful for insider threat detection. We remove the usual data aggregation step: using only one data source (authentications, emails, web traffic, etc.) at a time, we perform detection at fine-grained (i.e. log line) level. In order to support graph and text features, we introduce ADSAGE (Anomaly Detection in Sequences of Attributed Graph Edges). Our evaluation shows that ADSAGE ranks among the best performing methods for authentication and email logs; it also appears that simple rule-based classifiers perform surprisingly well. Our results show that detection at fine-grained audit event level is feasible.

1.5 Structure

The rest of this thesis is structured as follows. In chapter 2 we review the state-of-the-art regarding several topics relevant to this work, outlining existing developments and research gaps. In chapter 3, we describe the different datasets used in this study. Chapter 4 addresses research questions **RQ. 1** and **RQ. 2** in the context of our real-world intrusion detection use case. In chapter 5, we address research question **RQ. 3** by using zero-shot learning to overcome the "cold start" problem, i.e. absence activity data for some users. In chapter 6, we address **RQ. 4** and leverage heterogeneous audit log data in the CERT insider threat use case. Finally, chapter 7 concludes this thesis by summarizing our main findings and outlining further development perspectives.

1.6 Publications

Parts of this thesis' content have been published as follows.

- M. Garchery and M. Granitzer, "On the influence of categorical features in ranking anomalies using mixed data," *Procedia Computer Science*, vol. 126, pp. 77–86, 2018 (section 4.1);
- M. Garchery and M. Granitzer, "Identifying and clustering users for unsupervised intrusion detection in corporate audit sessions," in *2019 IEEE International Conference on Cognitive Computing (ICCC)*, IEEE, 2019, pp. 19–27 (section 4.2);
- S. Zerhoudi, M. Garchery, and M. Granitzer, "Improving intrusion detection systems using zero-shot recognition via graph embeddings," in *IEEE Annual Computer Software and Applications Conference, COMPSAC 2020*, to appear, IEEE, 2020 (section 5.2);
- M. Garchery and M. Granitzer, "Adsage: Anomaly detection in sequences of attributed graph edges applied to insider threat detection at fine-grained level," *arXiv preprint arXiv:2007.06985*, 2020 (section 6.1, non peer-reviewed).

Chapter 2

Related work

Research on intrusion detection has attracted a lot of attention since the publication of the seminal paper by Denning [15] in 1987. Early surveys by Lunt [16] and Axelsson [4] have provided a first overview of detection techniques, measures and intrusion types. However the space of existing intrusion detection settings and systems designed to address them is extremely vast. A synthetic classification mainly based on detection approach and type of audit data sources used can be found in [17]. Hindy *et al.* [5] provide a more recent classification of IDS including extensive taxonomies of detection systems and threats. Additionally, some surveys focus on specific perspectives, like network intrusion detection [18] or the use of data mining and machine learning in this domain [19], [20].

In this chapter, our aim is to review literature concerning selected aspects of intrusion detection, which are most relevant to this thesis. First, in section 2.1 we focus on masquerades and insider threats, the two types of intrusions addressed in this work. We provide definitions for both intrusion types and review existing methods to detect them from a data-centered perspective. That is, we emphasize which audit data sources are most relevant to detect masquerades and insider threats.

Second, in section 2.2 we outline challenges in evaluating intrusion detection systems, which is particularly difficult for several reasons. This includes lack of public evaluation datasets and inherent difficulties in aligning evaluation methodologies. To illustrate this last point, we examine the state-of-the-art on the CERT insider threat problem, which we address in this thesis.

Next, in section 2.3 we review the (rare) use of heterogeneous data in unsupervised anomaly detection. We detail general methods to perform anomaly ranking in mixed data, as required to address our real-world intrusion detection problem in chapter 4. We then review the use of text and graph data for intrusion detection, suggesting that they constitute promising features. Nevertheless, we observe that these heterogeneous features have been largely ignored by IDS addressing the CERT insider threat problem. Our method ADSAGE, described in chapter 6, fills this research gap, as well as our system proposed in chapter 5, though in a more specific setting. We also review anomaly detection methods suited for sequences of graph edges with heterogeneous attributes, eventually finding that ADSAGE is the first method to address this particular setting.

Finally, in section 2.4 we provide background on zero-shot learning and its application to intrusion detection. The few existing works in this domain concern network intrusion detection; our system described in chapter 5 is the first to use zero-shot learning against insider threats.

2.1 Intrusion types and relevant sources of audit data

In the following, we review existing methods to detect two types of intrusions relevant to this work: masquerades (addressed in chapter 4) and insider threats (chapters 4, 5 and 6). In addition to providing definitions for these intrusion types, our focus is on giving an overview of which audit data sources are relevant to detect them.

2.1.1 Masquerade detection and user identification

In intrusion detection, masquerades refer to a specific scenario, where credentials are used by a different person than the one they were originally assigned to. Note that although masquerades are often conducted with stolen credentials, this definition includes "friendly" masquerades, i.e. when a user shares his credentials with someone else, for legitimate or illegitimate purposes.

Several research works have addressed masquerade detection since the seminal work by Schonlau *et al.* [21], which introduced a dataset of Unix commands for evaluation. It is noteworthy that this dataset does not contain real masquerades. Instead, user command histories are contaminated with traces from different users in order to simulate intrusions. Therefore when considering users individually, masquerades are expected to differ from (own) normal behavior. However, this does not model the adversarial setting: in reality, attackers could attempt to cover malicious traces, which is not the case here. Note that we use a similar approach to inject masquerades in a real-world dataset of audit sessions (see section 4.2.3).

Using the dataset introduced by Schonlau *et al.* [21], different methods were proposed to perform masquerade detection in Unix commands. Maxion and Townsend [22] show that usage of rare commands allows to distinguish between users. Later approaches focus on taking into account contextual information like the sequence of commands [23]–[25] or command arguments [26] to extract more precise user profiles. In addition to user command logs, masquerade detection methods have been proposed for process [27] and search [28] logs, as well as network flows [29], [30].

Note that masquerade detection is by nature tightly linked to user profiling, which consists of constructing descriptions of normal user behavior [31]. If virtually all anomaly-based intrusion detection systems rely on some form of user profiling, it is of primary importance in masquerade detection: user identification is used to determine if a user's claimed identity (i.e. corresponding to his credentials) matches his real identity (as estimated through behavior analysis). Various approaches to user profiling and identification can be found in literature, for example via process logs [32], graphical user interface and application usage [33], [34] and web browsing [35].

In chapter 4, we address masquerade detection in a real-world and a synthetic dataset (see dataset descriptions in sections 3.1 and 3.2) through user identification. User profiles of normal behavior are implicitly learned by the classifier used for user identification. The novelty of our method resides in using clustering on the user identification confusion matrix to reduce false positives. Other applications of clustering in intrusion detection include feature extraction [36], discovery of unseen threats [37] and profile construction [38], [39].

2.1.2 Insider threats

Whereas masquerades correspond to a well delimited intrusion scenario, insider threats encompass a much broader range of malicious activities. Hunker and Probst [40] state that "an insider threat is [posed by] an individual with privileges who misuses them or whose access results in misuse". This rather general definition encompasses for example credential theft and misuse (i.e. masquerades), sabotage of IT systems, theft and public release of private information, and in general any usage of an organization's IT infrastructure to conduct illegal activities, possibly for personal gain. One common point of different insider incidents is that an insider possesses advanced knowledge about the environment in which he will commit fraud. According to Homoliak *et al.* [41], "insiders are authorized users that have legitimate access to sensitive/confidential material, and they may know the vulnerabilities of the deployed systems and business processes". This knowledge is critical as it can help the attacker reach his malicious aim, increase attack impact and evade detection. For this reason, and because malicious insiders possess legitimate access by definition, their detection is an extremely hard problem. In the following, we review counteraction approaches from a data-centered perspective, detailing which sources of audit data to collect for insider threat detection.

An early survey by Salem, Hershkop, and Stolfo [42] insists on the importance to "derive user intent" in order to detect and characterize insider threats. Indeed, such intrusions are hidden in the mass of normal activities and individual audit data collections might not be sufficient to characterize whether an action is malicious or not. For example, a user accessing sensitive documents from the organization he belongs to cannot be considered anomalous per se, if we admit that he is authorized to do so. Nevertheless, copying those documents to a personal data storage device or sending them to an external recipient would probably constitute fraud. This shows the importance of combining different audit data sources – in this case document repository access, file system and communication logs – in addressing insider threats.

Two common audit data source categories are host-based and network-based logs, which respectively correspond to activity data collected locally on and remotely between user workstations [42]–[44]. Host-based audit traces relevant to insider threat detection include authentication logs [45], [46], user and system issued commands [47], process logs, program/application usage, file system activity [45], usage of removable devices [45], [46] and database accesses [48]. Network-based audit data sources can represent web browsing [46], [49]–[51], phone and email communications [45], [50], [52], printer usage [53], but also lower level network traffic [54]. Of course these lists are not exhaustive, as research efforts continue to craft and evaluate new forms of data collection for insider threat detection. Examples of features that can be extracted from these audit data sources can be found in table 2 from [44].

Besides diverse activity logs, another source of information highly relevant to insider threat detection is user "meta-data", also referred to as "contextual data" by Liu *et al.* [43]. This includes knowledge collected independently of user activity, such as HR records or psychological profiles. As Gheyas and Abdallah [44] argue, these features are helpful to characterize motives of malicious insiders such as predispositions to malicious behavior, personality traits and emotional state. User roles within the organization are the most commonly used form of contextual data [52], [55], [56], as they are much easier to collect than psychometric measures for example.

In this thesis, we adopt a user-centered perspective and try to leverage all activity features available. Thus we use a mix of host and network-based activity features

to first detect masquerades (chapters 4) then insider threats (chapters 4, 5 and 6). A method to leverage contextual data when activity data is missing is presented in chapter 5.

2.2 Challenges in intrusion detection evaluation

Evaluation of intrusion detection systems faces significant challenges. As shown next, two main difficulties are the lack of realistic public datasets (section 2.2.1) and the variety of operational requirements (section 2.2.2). But even for the CERT insider threat problem, which does not suffer from these two shortcomings, no standard benchmark methodology has emerged (section 2.2.3). This hinders straightforward comparison with existing systems.

2.2.1 Data (un)availability

A significant – if not the first – challenge of intrusion detection research is the lack of public datasets for evaluation. An important reason for that is the sensitivity of audit records, whose goal is precisely to trace user activities. The considerable effort required to remove personal identifiable information from audit traces, along with security by obfuscation strategies, explains why organizations are overwhelmingly reluctant to release such data.

Furthermore, audit traces alone are not sufficient for sound intrusion detection evaluation: ground truth, i.e. annotations of anomalies must be provided as well. This is another blocking point. If some intrusions into an organization were blocked by detection methods in place, no new development is required and the incentive to release corresponding traces is low. Conversely, if malicious activities slipped through existing protection mechanisms, releasing audit traces show vulnerability, which can damage the organization's image.

To alleviate these issues, synthetic datasets can be generated. This however raises the question of their realism. Beyond simulating normal behavior, the main problem lies in the adversarial setting. In a real-world context, intruders will try to evade detection. This cannot be reasonably simulated when audit logs data are generated by the same people who develop intrusion detection systems. One possible solution, although not perfect, is to collect audit traces representing normal behavior and hire a "red team" to inject malicious activities. This approach is notably used for the CERT [57], [58] and LANL authentication [59], [60] datasets in this thesis.

Table 2.1 lists the main public intrusion detection datasets with available ground truth. Since the publication of the KDD cup dataset in 1999, research interest in intrusion detection has increased significantly. The KDD cup 99 dataset [61] has long been the most widely used by the community, despite numerous flaws questioning its realism [62], [63]. An improved version (NSL-KDD [63]) was later proposed to address some of these shortcomings. However both datasets contain traces collected at the end of the 1990s which are now considered outdated [64], [65]. The fact that both variants are still largely used [20], [64] is quite symptomatic for the lack of test data in the field.

Additionally, the problem of data scarcity is more or less striking depending on the type of targeted intrusions: network attacks, masquerades and insider threats. For network-based intrusion detection, several realistic collections of traces have been released in recent years [66]–[68].

In masquerade detection, research has been fostered by the release of the Schonlau dataset [69] in 2000, which contains histories of Unix commands issued by users. However this dataset does not contain realistic intrusions. Subsequent works released audit logs from different sources to tackle masquerade detection, using process traces [70] and file system navigation behavior [71].

Since the 2010s, research has turned to the detection of insider threats, which cover a broad scope of malicious activities (including masquerades). As malicious insiders possess authorized access and often advanced knowledge of IT systems they attack, their detection is very difficult. Audit logs from multiple data sources such as email, file and browsing activities are usually required to spot them within the majority of normal users. For this reason, insider threat detection blurs the distinction between network-based and host-based intrusion detection. To this date, the CERT datasets [57], [58] represent the most extensive data collection for evaluation of insider threat detection. We describe their content in detail in section 3.2. Another recent data collection for insider threat detection is TWOS [72]; nevertheless, intrusions are expected to be less realistic as the dataset was collected in a student competition.

Dataset name	Release year	Type of intrusions	Audit log sources	Realism, modifications
KDD cup 99 [61]	1999	network intrusions	TCP dumps	collected in testbed environment injected attacks
LBNL/ICSI [73]	2006	network scanning	TCP dumps	anonymized, only packet headers
NSL-KDD [63]	2009	network intrusions	TCP dumps	(same as KDD cup 99)
UNB ISCX IDS [67]	2012	network intrusions	flows from several protocols: http, email, ssh, ftp	synthetic, generated from profiles built upon real-world traces
UNSW-NB15 [66]	2015	network intrusions	TCP dumps	real-world normal traces, synthetic attacks
UNB CICIDS [68]	2017	network intrusions	flows from several protocols: http, email, ssh, ftp	synthetic, generated from profiles built upon real-world traces
Schonlau dataset [69]	2000	masquerades	Unix commands	anonymized command arguments removed synthetically injected intrusions
RUU (Are You You?) [70]	2011	masquerades	Windows processes logs	real-world normal traces simulated intrusions
WUIL [71]	2014	masquerades	file system navigation traces	real-world normal traces simulated intrusions
ADFA-LD, ADFA-WD [74]	2013	remote attacks, privilege escalation	process logs	real traces, collected in test environment
CERT datasets [57], [58]	2013	insider threats	desktop, email and web logs	synthetic, red team intrusions
LANL cybersecurity events [59], [60]	2015	compromised authentications	authentications, processes network flows	real-world, red team authentications
TWOS [72]	2017	insider threats, masquerades	desktop and network logs	collected in student competition

TABLE 2.1: Overview of public intrusion detection datasets with available ground truth.

2.2.2 Operational environment: specificity versus comparability

Besides the lack of evaluation datasets, comparing the performance of intrusion detection systems is inherently difficult due to the variability of operational environments. First, unlike other anomaly detection application domains like credit card fraud or spam detection, intrusion detection can be performed using various types of audit data sources. The nature of collected traces depends not only on the targeted types of intrusions, but also on the sensors available in practice. In some cases, and as discussed later in this thesis, available data sources are not sufficient, raising the question of how to extend the audit data collection. Anyway, there is no standard set of feature shared in all IDS operational environments. Thus in real-world intrusion detection problems, simply applying existing methods without adaptation is often not possible, which ultimately discourages comparison with state-of-the-art.

In addition to available audit data sources, the definition of what constitutes an intrusion is very much context-dependent. Indeed, from the perspective of an organization deploying an intrusion detection system, ideally all violations of security policies observable within collected audit data sources should be detected. But the definition of security policies – and therefore of what constitutes a violation of such policies – is set by the organization and subjective. For example, should users be allowed to access their emails at night? Different organizations might provide different answers. Drawing the line between acceptable and forbidden behavior can be even more difficult for insider threats: is sending documents to one's private email address acceptable? The answer is that it highly depends on the context (content of said documents, confidentiality classification, etc.).

Meanwhile, anomaly-based intrusion detection methods rely on building profiles of normal (i.e. frequently observed) behavior and flagging deviations from this normality. No wonder then if a significant mismatch appears between anomalies – in the sense of infrequent patterns according to the anomaly detection method – and intrusions, as defined by the operational environment. This problem, referred to as "semantic gap" [9], [75], not only leads to poor detection performance in practice; it also poses a significant challenge to sound evaluation.

Even when applying the same detection methods on the same audit data sources, in two different operational environments one could get identical anomaly detection results, but those results would have to be interpreted differently, in light of possibly different definitions of malicious behavior. To make it worse, the cost of errors – which, by the way, is very difficult to quantify in practice – also depends on the operational environment. Using custom evaluation metrics can give a clearer picture of detection performance in a specific context; however at the price of reducing comparability with existing works.

2.2.3 Lacking benchmark standardization: the example of the CERT insider threat problem

We have previously seen that lack of test data and specific operational environment requirements are two major problems in comparing intrusion detection systems. Still, there are some public evaluation datasets available, which can be seen as specific intrusion detection problems in their respective operational environments. Although many approaches have been proposed to address those use cases (in majority through machine learning solutions), the community lacks standardized benchmark methodologies in most cases. Across the literature, various changes in

Ref.	Dataset version	ROC AUC	Detection rate (recall)	False positive rate	Other metrics
[76]	r6.2				CR-1000 = 35.7/40 = 0.89
[52]	Vegas	0.77			
[50]	r2		0.50		precision = 0.08
[77]	r4.2		0.40		precision = 0.81
[78]	r4.2	0.83	0.85	0.20	
[79]	r4.2	0.95			
[80]	r4.2		0.81	0.12	
[81]	r6.2?		0.60		precision = 0.84
[82]	r4.2	0.86	0.86	0.20	
[83]	r4.2		0.81/0.74 (at 25%/5% budget)		
[84]	r4.2	0.99			

TABLE 2.2: Detection performance reported by various systems addressing the CERT insider threat detection use case.

evaluation methodologies, including metrics choice, data selection and preprocessing among other factors hinder straightforward comparison of experimental results.

To illustrate this, we give an overview of results reported on the CERT insider threat datasets (which are the most relevant to this thesis, see section 3.2) in table 2.2. In most cases, the ROC AUC score (area under the receiver operating characteristic curve, obtained by plotting the true positive rate against the false positive rate) is used. It reflects detection performance across all decision thresholds and can be interpreted as the probability that an anomaly is ranked higher than a normal observation. For a single decision threshold, detection and false positive rate can be used. Nevertheless it should be noted that performance at one decision threshold is not representative of overall system performance. Some systems report detection performance metrics at both single and multiple thresholds.

Considering ROC AUC, Hall *et al.* [84] report the highest score of 0.99. However their data selection is possibly biased, as they train their system exclusively on users with device activity and consider only one month of data. Furthermore, they rely on features indicating whether users leave the organization shortly after their logs were collected, but in practice such information would not be available during audit log collection. Therefore their evaluation setting is unrealistic and whether their method would perform as well on a broader scope of threats remains an open question. Second best performance in terms of ROC AUC is reported by Yuan *et al.* [79] (0.95). They heavily rely on specific feature engineering to distinguish activities associated with higher level of insider threat risk: actions performed after hours, usage of unassigned computers and browsing of sensitive websites among others. Their system uses a recurrent neural network to extract matrix representations of activity sequences and a convolutional neural network to classify extracted matrices. They claim using around 70% of the dataset for training, and the rest for evaluation, but give no further details about how the split was performed. Rashid, Agrafiotis, and Nurse [78] report a ROC AUC of 0.83, again without mentioning which portion of data was used for evaluation.

When considering detection results at single decision threshold (i.e. detection and false positive rate), the best performance is reported by Le and Zincir-Heywood [82], achieving 86% of detected threats with 20% false positives (and 0.86 ROC AUC).

Unfortunately, because of the very high class imbalance, the false positive rate is much too high for the system to be useful in practice (e.g. with 1 million samples it would generate as much as 200 000 false alarms). Lin *et al.* [80] report a detection rate of 0.81 along with a false positive rate of 0.12, but provide no further information about the data selection used for evaluation. Le *et al.* [83] report detection rate along with accuracy, a questionable choice given the high class imbalance.

In such imbalanced class setting, it is generally more meaningful to use the (area under the) precision recall (PR) curve [85]. Böse *et al.* [50] report precision and recall plots (against decision threshold) equivalent – but not directly comparable – to a classical precision-recall curve. Similarly, Lu and Wong [81] plot precision and recall against a parameter influencing the decision threshold. Lv *et al.* [77] report recall and precision at fixed threshold but not the full precision-recall curve.

An interesting alternative to precision and recall consists in using a business metric like Tuor *et al.* [76]. Here a certain budget represents the resources available to investigate alerts raised by the intrusion detection system. This budget corresponds to the number of cases to be investigated per day, therefore setting a realistic decision threshold. Then the corresponding recall over all days is reported; cumulative recall values can be used as global detection performance indicator across all thresholds (see more details in section 6.1.3).

In conclusion, despite several research works addressing the CERT insider threat use case, no standard evaluation setting has emerged. However the evaluation methodology proposed by Tuor *et al.* [76] stands out through its comprehensive description encouraging reproducibility, and the realism of its recall-based metrics. For these reasons we rely on this evaluation setting in chapters 5 and 6.

2.3 Unsupervised anomaly detection in heterogeneous data

Heterogeneous data has been widely ignored in unsupervised anomaly detection, as existing methods mostly support numeric attributes. We summarize existing approaches from three perspectives related to this thesis. First, in section 2.3.1 we review anomaly ranking methods applicable to mixed data (i.e. containing numeric and categorical features). Then we discuss the usage of text and graph features in intrusion detection (section 2.3.2), with a particular emphasis on the CERT insider threat use case (section 2.3.3). Finally, in section 2.3.4 we review approaches for sequences of graph edges with attributes, as addressed by our method ADSAGE.

2.3.1 Unsupervised anomaly ranking in mixed data

We here review methods for anomaly ranking (i.e. outputting continuous anomaly scores) in mixed data (i.e. containing categorical and numeric attributes). Taha and Hadi [86] provide a comprehensive overview of anomaly detection methods for categorical data, focusing on input parameters and time complexity. However few methods fit our requirements of performing unsupervised anomaly ranking and supporting mixed data. In particular, many methods require to provide the number of anomalies among other parameters, which supposes knowing the optimal decision threshold beforehand (see table 2 in [86], decision parameter "M"). Similarly, LOADED [87] and ODMAD [88] require parameters to determine the decision threshold indirectly. This is fundamentally incompatible with ranking. K-LOF [89] is designed for categorical data, and numeric data is supported through an additional preprocessing step, but authors do not mention whether it is simply applicable to

mixed data. In the end, remaining options are count-based method SPAD [90], FRaC [91] and the method proposed by Rashidi, Hashemi, and Hamzeh [92].

When evaluating our user identification approach for unsupervised intrusion detection (see section 4.2), we use SPAD as baseline anomaly ranking method, which we selected for its appealing simplicity and performance reported in [90]. We discarded the method from Rashidi, Hashemi, and Hamzeh [92] due to its high time complexity and unfortunately, we were unaware of FRaC [91] at the time of conducting experiments.

2.3.2 Heterogeneous features for intrusion detection: graph and text

After reviewing general anomaly ranking methods for mixed data, let us now narrow our scope to intrusion detection. In the following, we review how two types of heterogeneous features, text and graph, are used in this domain.

Anomaly detection based on graphs and graph features has been successfully applied to insider threat detection. In many cases, graph features prove useful to highlight local deviations from normal behavior which would remain undetected with more simple statistical (i.e. numeric) features. Graph features are used to model relationships between users and physical devices of information systems [53], user accesses to resources in collaborative systems [93], [94] or flows from business [95] and system [96] processes. Interactions among users, such as email communications, are also of particular interest to detect insider threats [95], [97], [98]. Some works use graphs to represent links between abstract entities like tasks corresponding to work roles [99] or knowledge acquired by members of an organization and relevant to perpetration of malicious activities [100]. As insider threats are often correlated with employee quitting [52], attrition prediction is sometimes addressed as proxy task for insider threat detection; social graphs often are informative in this case [56], [98]. Besides insider threats, graph features can be used to detect intrusions in Internet of Things (IoT) [101] and physical security networks [102] or for malware propagation [103], [104].

In addition to graphs, the pervasiveness of text in real-world data has motivated researchers to leverage these features for diverse tasks. In recent years, breakthroughs in the domain of language models based on neural networks and semantic vector representations of words (*embeddings*) [105] have found many practical applications, including in intrusion detection. Even though the idea of using natural language processing on system call logs is not new [106], it was recently revived by using recurrent neural network language models to find anomalies in log sequences [107], [108]. Complementing sequence modeling, convolutional neural networks can be used to extract informative features from text data at character [109] or word level [110]. Public collections of word embeddings (e.g. [111]–[114]) allow to easily leverage text sources by converting them to numeric representations without further training [115]. An example in network intrusion detection can be found in [116]. Furthermore, specific embeddings can be learned for intrusion detection [117], although it requires a large source of audit logs and sufficient computing resources. For detecting insider threats, text features can be very useful as writing style and usage of particular words can reveal personality traits and sentiments of the writer, which are informative markers [55], [118], [119].

In conclusion, heterogeneous features such as graph and text data are promising for intrusion detection in general, and to better characterize insider threats in particular.

2.3.3 Leveraging heterogeneous features in the CERT datasets

In the previous section, we have provided an overview of how heterogeneous features can be useful for intrusion detection. We now focus on the CERT insider threat use case by reviewing how previous works use the different data sources present in the datasets.

The CERT datasets contain multiple sources of audit logs relevant to characterize malicious insider activities: authentication events, usage of removable devices, web and email usage as well as file operations. In addition to audit logs, descriptions of user roles, positions and their assignments to teams and projects are provided. Psychometric profiles quantifying personality traits of users are also available. We refer the reader to section 3.2 for a detailed description, our current focus being on the usage of different data components in existing works, and in particular heterogeneous features.

As shown in table 2.3, most existing systems addressing the CERT insider threat use case rely primarily on numeric data. This includes statistical features obtained directly from audit log events (e.g. time of authentication, file type, size of email and attachments, etc.). Some systems rely on feature engineering to extract more complex numeric features (e.g. count of visited URLs per website category [49], [52], [83], count of emails sent and received in some time interval [76]). These features are referred to as metadata in table 2.3 and are often used in literature. On the contrary, heterogeneous features – in particular graph and text attributes – have been widely ignored despite their pervasiveness in the CERT datasets.

Text data can be found in the content of web pages, email messages and file browsed by users. Among existing insider threat detection systems, only the one introduced by Legg *et al.* [120] makes use of web page and file contents, by extracting bag-of-words count features. Regarding email text content, Gavai *et al.* [52] use simple statistical features (like count of punctuation symbols and average word length). Legg *et al.* [120] rely on LIWC features (Linguistic Inquiry and Word Count [121]) assigning words to pre-defined sentiment analysis categories.

In addition to text, the CERT datasets contains another type of heterogeneous data: graph representations. They can model relations between users and resources (user to computer links in authentications, user to web pages) or among users (email communications, organization's hierarchy described as LDAP records). Like text features, graph attributes are rarely taken into account. For example Gamachchi and Boztaş [49] are the only ones to use the web page graph. The email communication graph is used in some cases but not fully exploited: Rashid, Agrafiotis, and Nurse [78] and Le *et al.* [83] only distinguish internal from external recipients. Agrafiotis *et al.* [122] claim using recipient information to characterize email behavior but do not give further details. Only one system uses the graph of user to computer relations [123].

The most exploited source of graph data is probably the organization's structure, which comprises hierarchical relations between users and their assignments to projects, teams and departments. Tuor *et al.* [76] integrate these contextual attributes as categorical values but report negative influence on detection performance. Others use role information to build specific profiles [120], to perform data selection [123] or to analyze results a posteriori [49]. Note that the two last approaches use LDAP features indirectly, which is why they are marked as ignoring LDAP features in table 2.3. Le and Zincir-Heywood [82] and Le *et al.* [83] claim using role features but remain vague about how these features are used. Finally, Hall *et al.* [84] use LDAP

features only to determine whether a user has left the organization shortly after his activity was recorded.

Ref.	Authen-tications (logon)	Removable device usage (device)	Web usage (http) URL metadata	Page content	Page graph	Email usage Email metadata	Email content	Email graph	File operations File metadata	File content	Positions and roles (LDAP)	Psychometric user profiles
[76]	■	■	□	□	□	■	□	□	■	□	■	□
[52]	■	?	■	□	□	■	■	□	?	?	□	□
[120]	■	■	□	■	□	■	■	□	?	■	■	□
[123]	■	■	■	□	□	□	□	□	□	□	□	■
[45]	■	■	■	□	□	■	□	□	■	□	□	□
[50]	■	■	■	□	□	■	□	□	■	□	□	□
[77]	■	■	□	□	□	□	□	□	■	□	■	□
[49]	■	?	■	□	■	?	?	?	?	?	□	□
[122]	■	■	■	□	□	■	□	■	■	□	□	□
[78]	■	■	■	□	□	■	□	■	■	□	□	?
[46]	■	■	■	□	□	□	□	□	□	□	□	□
[79]	■	■	■	□	□	■	□	□	■	□	□	□
[80]	■	■	■	□	□	■	□	□	■	□	□	□
[81]	■	■	■	□	□	■	□	□	■	□	□	□
[83]	■	?	■	?	?	?	?	■	?	?	■	■
[82]	■	■	■	□	?	■	□	?	■	□	■	■
[84]	■	■	□	□	□	□	□	□	□	□	■	■

TABLE 2.3: Usage of the different data sources from the CERT datasets in existing systems, which include audit logs (logon, device, web, email and file), description of user positions and roles and psychometric profiles indicating user personality traits. The ■ symbol indicates that corresponding data source is taken into account, □ indicates that corresponding data source is ignored. Question marks indicate unknown data source usage. Text features are highlighted in blue, graph features in orange.

To summarize, existing insider threat detection systems have widely discarded graph and text data available in the CERT datasets. However, as suggested in section 2.3.2, these heterogeneous features could be very informative to characterize malicious insider behavior. This thesis aims to tighten this research gap from two perspectives. First, by leveraging user information independent from activity logs (such as role and project assignments) to better estimate user behavior when historical data is unavailable (chapter 5). Second, by proposing a framework to support heterogeneous data at line level in audit logs, while reducing the need for feature engineering (chapter 6).

2.3.4 Anomaly detection in sequences of graph edges with attributes

In section 6.1, we address the CERT insider threat use case via unsupervised graph anomaly detection at edge level. Our method ADSAGE supports sequences of edges and edge attributes. Moreover the corresponding graphs are dynamic, changing over time as new edges are added. We here review related methods form literature.

Akoglu, Tong, and Koutra [124] survey anomaly detection methods for graphs, but they report being unable to find methods for anomaly detection in dynamically changing attributed graphs. The vast majority of surveyed methods output anomaly scores for nodes in static graphs. Some methods support dynamic graphs, but focus on detecting if the whole graph becomes anomalous at some point (and not on individual graph edges). In another survey, Ranshous *et al.* [125] more specifically focus on methods for dynamic graphs. They report finding several methods for detection at edge level, but these methods support neither sequences of edges nor edge attributes.

Since the publication of the two previous surveys, graph anomaly detection has attracted much attention in the machine learning community and more methods were released. We briefly outline the most relevant ones in the following. StreamSpot [126] and SpotLight [127] detect anomalies in graph streams, however detection is not performed at edge level. Although EdgeCentric [128] supports edge attributes, it detects anomalies at node level only and does not support dynamic graphs.

The body of work most similar to ADSAGE consists in a set of recent methods addressing unsupervised anomaly detection at edge level in dynamic graphs: SedanSpot [129], NetWalk [130], AddGraph [131], AnomRank [132] and the method introduced by Ranshous *et al.* [133]. Unfortunately none of these methods supports edge attributes, which limits their usability in our use case. There is no benchmark comparing all methods at once. Authors of AddGraph claim to outperform NetWalk [131]. Eswaran and Faloutsos [129] report that SedanSpot outperforms the method from Ranshous *et al.* [133]. Yoon *et al.* [132] claim that AnomRank is significantly faster than SedanSpot, although anomaly detection effectiveness is similar for both methods.

In the end, our literature review reveals that there is currently no unsupervised anomaly detection method in sequences of graph edges with attributes. Our method ADSAGE, introduced in section 6.1, fills this gap. We compare it to SedanSpot [129], which provides properly documented implementation. Lastly, note that AddGraph [131] uses a neural network with negative sampling to generate artificial anomalous edges in the training phase, like ADSAGE.

2.4 Zero-shot learning for intrusion detection

In the following, we introduce zero-shot learning, its applications to intrusion detection and position our system described in chapter 5.

2.4.1 Introduction to zero-shot learning

Zero-shot learning refers to a learning setting designed to allow classification of unseen class examples. As noted in [134], this ability is important in situations where sufficient amounts of observations cannot be collected for some classes. This might be due to intrinsic rarity of said classes, or because labeling is too expensive, e.g. due to very high total number of classes or concepts changing over time.

Although traditional (i.e. non zero-shot learning) classification methods can only classify observations of seen classes, it has been noted that humans can easily overcome this limitation. The archetypal intuitive example of zero-shot learning is that a person who has never seen a zebra might be able to recognize one, if told that a zebra looks like a horse with black and white stripes [135], [136].

Wang *et al.* [134] provide a formal definition of the zero-shot learning setting as follows. Let S be the set of seen classes, U the set of unseen classes and X the feature space (typically, a multidimensional real number space). The training set, corresponding to observations for seen classes x^{tr} associated pairwise with their corresponding labels y^{tr} is denoted $D^{tr} = \{(x^{tr}, y^{tr}) \in X \times S\}$. The test set consists of test observations $X^{te} = \{x^{te} \in X\}$ and their corresponding labels $Y^{te} = \{y^{te} \in U\}$, which are to be predicted. In this setting, the aim of zero-shot learning is to learn a classifier $f : X \to U$ to classify test observations X^{te} into unseen classes U, given the training set, i.e. labeled observations D^{tr} corresponding to seen classes S.

In order to overcome the absence of training observations for unseen classes, a different source of description is required, which Wang *et al.* [134] call "auxiliary information". In the case of human learning (as in the previous zebra example), this is typically done via semantic description features (e.g. "How many legs does it have?" or "What color is it?") [137], [138]. These features are represented in a semantic space and, crucially, must be available for both seen and unseen classes.

More formally, Wang *et al.* [134] extend their previous definition as follows to account for the role of the semantic space. Let T denote the semantic space (typically, a real multidimensional space), T^S and T^U the sets of semantic representations for seen and unseen classes respectively (called class prototypes or embeddings). Then zero-shot learning implies using the class prototypes T^S and T^U in addition to the labeled observations D^{tr} in order to obtain the classifier f.

This quite general definition however leaves many possibilities open as to the nature of the semantic space, how to obtain it and how the class prototypes are used in learning the zero-shot classifier. In their survey, Wang *et al.* [134] provide an overview and categorization of existing semantic spaces and zero-shot learning methods.

As pointed out in [134], zero-shot learning can be seen as a special case of transfer learning, in which source and target feature spaces are the same, but source and target label spaces differ. Note that in "pure" zero-shot learning settings, source and target label spaces (corresponding to seen and unseen observations respectively) are disjoint; however in practice they may overlap: this last setting is known as generalized zero-shot learning [139]. One-shot [140], [141] and few-shot [142] learning techniques are also closely related; they relax the assumption that unseen classes

have zero examples. In practice, a small number of instances might be available, though too few to apply traditional classification methods.

2.4.2 Application to intrusion detection

The most common applications of zero-shot learning are computer vision and natural language processing tasks, such as image and video recognition or machine translation [134]. However the usage of zero-shot learning is still rare in the security domain, and in particular for intrusion detection.

Chowdhury *et al.* [143] use few-shot learning for network intrusion detection. By using a convolutional neural network for feature extraction in the attribute learning stage, they improve recognition for rare classes of attacks. Some works focus on the attribute learning step only: Li *et al.* [144] combine feature selection based on random forest and clustering while Pérez and Ribeiro [145] learn class attributes in the form of rules. Rivero *et al.* [146] extend the last approach to include an inference step, by using a nearest neighbor method in projection space to identify new classes of network attacks. Zhuang *et al.* [147] target network intrusion detection with a particular data format they call "attributed sequences", which consists of a profile containing diverse attribute value and a sequence of categorical values. Their one-shot learning framework allows to learn relationships between profile and sequences while addressing data scarcity.

Further loosely related works include zero-shot learning applied in reinforcement learning context [148] and for other security applications, like device fingerprinting [149] and malware classification [150].

2.4.3 Positioning our approach

To the best of our knowledge, there is no previous work addressing insider threat detection through zero-shot learning. In chapter 5 of this thesis, we design and evaluate such a system. According to the classification presented in [134], we use a learned semantic space obtained via label embedding. More precisely, our class prototypes are obtained via graph embedding techniques applied on organizational relations. However classifying our zero-shot learning method into the categories defined in [134] is not as straightforward. The particularity of our system is that it predicts user activity while using this same feature space as input, i.e. it works as an auto-encoder. Hence it does not perform classification and therefore does not fit into the previous categorization. In our setting, semantic features (user embeddings) are used as additional input to the auto-encoder to compensate the lack of relevant description in (user activity) feature space.

Chapter 3

Datasets

This chapter describes the different intrusion detection evaluation datasets used in our experiments. First, we describe a real-world, private dataset of user accesses to business applications over a web portal (section 3.1). We explain how authentication and authorization events are aggregated into user web sessions to obtain our audit-sessions dataset. Second, we describe the CERT synthetic datasets for insider threat detection evaluation (section 3.2). Combining different data sources (authentication, email, web, file and removable device usage) from the CERT datasets, we create a collection of user sessions (cert-sessions) similar to the previous audit-sessions. Finally, we describe real-world LANL authentication logs which include red team malicious events (section 3.3).

Datasets mentioned above are used as follows in our experiments. In chapter 4, we use audit-sessions and cert-sessions for masquerade and insider threat detection respectively. In chapter 5, we evaluate our extension of an existing insider threat detection system for the CERT audit data sources. To this effect we reuse the data aggregation scheme of the baseline system. This data aggregation scheme for the CERT dataset is not a contribution of this thesis, which is why it is not described here; the interested reader can refer to [76] for more details. In chapter 6 we focus on event level insider threat detection, thus our primary data source for evaluation are the CERT datasets. We use authentication, email and web data sources with minimal preprocessing, as one of our goals is to reduce feature engineering and data aggregation. Additionally, we compare results obtained on (synthetic) CERT authentication logs with those from the (real-world) LANL dataset.

3.1 Real-world audit dataset

We describe a real-world, private dataset of user access logs. This dataset was collected within the IT company Atos from April 2016 to March 2017. It contains around 75 million audit events generated by over 100 000 users. Events represent users logging in (authentication events) or being granted access to a resource (authorization events). These logs were collected in the context of a web single sign-on portal which delivers access to several online business applications, such as the employee portal (for time sheets, leave and travel requests), the company's internal social network or human resources pages.

In order to understand the nature of the audit logs at hand, important aspects must be noted. First, for confidentiality reasons the dataset is anonymized. Real user names are mapped to fictional aliases and other identifiers are removed. Moreover, users identifiable indirectly with quasi-identifiers are removed as well (e.g. users working from countries where the company employs only a few people). Second, note that the dataset encompasses only a fraction of daily activities of a typical employee. Only accesses to online resources behind the company authentication web

portal are logged, which excludes all host-based audit traces, but also email and web traffic to external domains. One exception is VPN (Virtual Private Network) usage, for which authentication over the single sign-on portal is required (though only the initial authentication is logged). Third, the single sign-on setting is of great importance to explain the structure of the dataset: a user will typically authenticate once then access one or several resources. More technically, this means that one (successful) authentication event will lead to the creation of a new session, in which several authorization events can take place.

In the following, we describe the raw dataset as provided to us (section 3.1.1). We then detail how we augment it with a geolocation database for IP addresses to better characterize user locations (section 3.1.2). Finally, we motivate and describe our preprocessing approach which consists in aggregating audit events into sessions (section 3.1.3).

3.1.1 Raw dataset description

The raw audit dataset is provided as a database, whose structure is reproduced in figure 3.1. The main table (audit_messages) contains all authentication and authorization events. As a consequence, some attributes which are only relevant to one of both event types are sometimes left empty. Three extension tables (named "identification", "wherefrom" and "who") provide additional context for each event. They have unspecific columns (type and val), which are filled with the name and values of additional attributes for each event from the main table. Table 3.1 describes the content of columns relevant to this work. Note that certain columns present in the database schema are absent from this table; the reason is that they either do not contain any relevant information for our purpose or are redundant with some other column listed in the table.

Attributes with "identification" prefix characterize the event itself, for example its category (authentication or authorization), outcome (success or failure), session identifier, authentication method used or the target resource for which access is being requested. Columns with "wherefrom" prefix characterize the service requiring authentication, from a system architecture perspective; they are not relevant for our work. "Who" attributes provide information about the user requesting authentication or authorization, comprising mainly the user name, the country where this user is employed and the IP address from which the user is connecting.

Moreover, figure 3.2 shows the relationships between entities represented in the audit dataset: users, sessions, authorization and authentication events as well as some of their attributes. In particular, it should be noted that user IP address and authentication methods used are only present in authentication events; authorization events contain the accessed application. A given event can be associated to a unique user and a unique session; this session can be associated to a unique user as well and can be seen as a web activity session. Note also that user information (such as the country of employment) is part of the events and not accessible independently.

The raw audit dataset contains 31.7 million authentication and 43.8 million authorization events. The distribution of these events with respect to their time of occurrence is represented in figure 3.3. Unsurprisingly, most events occur on weekdays, with a peak of activity around 9:00 in the morning. Figure 3.4 shows the distribution of total number of authentications performed for each user. One can observe that a small number of accounts are significantly more active than the majority of users: they correspond to special credentials (e.g. system administrator or automated jobs). Figure 3.5 shows the number of authorization events (i.e. application

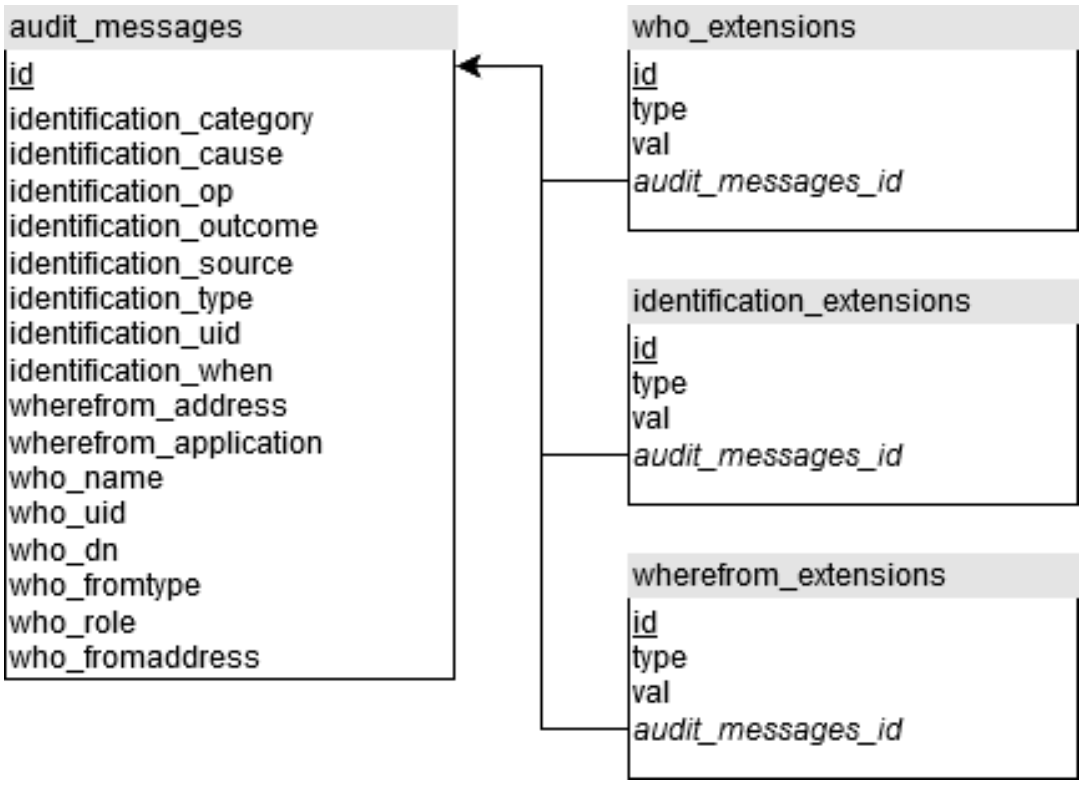


FIGURE 3.1: Database schema of the raw audit dataset. The main table (audit_messages) describes authentication and authorization events. Extension tables provide additional information for each event.

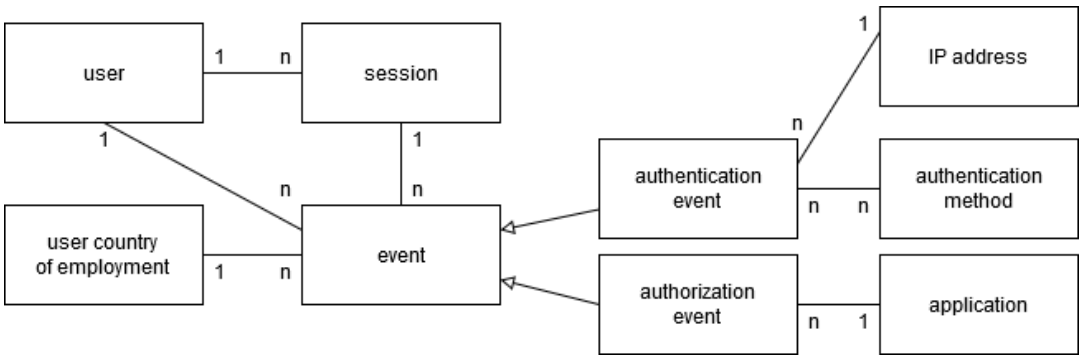


FIGURE 3.2: Entity relationship model of events in the real-world audit dataset. Note that certain attributes are specific to authentication or authorization events, and that associations between user and country of employment are only defined indirectly over events.

Table	Column	Description
audit_messages	id	event identifier
	identification_category	authentication or authorization
	identification_cause	session identifier
	identification_op	session creation, update or error
	identification_outcome	success or failure of authentication
	identification_type	authentication result and type
	identification_when	date and time of event
	wherefrom_address	address of server
	who_name	user name asking for authentication
	who_fromaddress	user IP address
who_extensions	country	user country of employment
	company	company or subsidiary employing user
identification_ extensions	AuthnMethod	authentication method
	AuthnMethodType	authentication method type
	SAML2:Audience	target resource or application

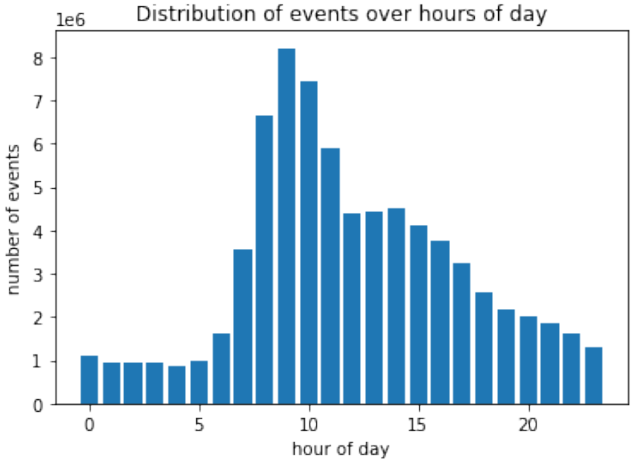
TABLE 3.1: Description of attributes present in the audit dataset. Note that certain columns present in the database schema are omitted as they either contain no relevant information or are redundant with other columns described in this table.

accesses) following an authentication event in the same session. One can note that many sessions have few authorization events, and the vast majority of sessions have less than a few tens events. The authentication failure rate is around 3.3%, which corresponds to around 1 million events. Figure 3.6 shows the proportions of different authentication methods by type (simple password, one-time password, smart card/certificate and one-time password via SMS). The majority of authentications are performed with smart card. Figure 3.7 shows the distribution of applications accessed. It appears that most common applications (1 and 2) are respectively the company's internal social network and the employee portal; both represent two thirds of all accesses. Finally, figure 3.8 shows the distribution of user countries, the most common ones being France, India, Germany, the United Kingdom and the United States.

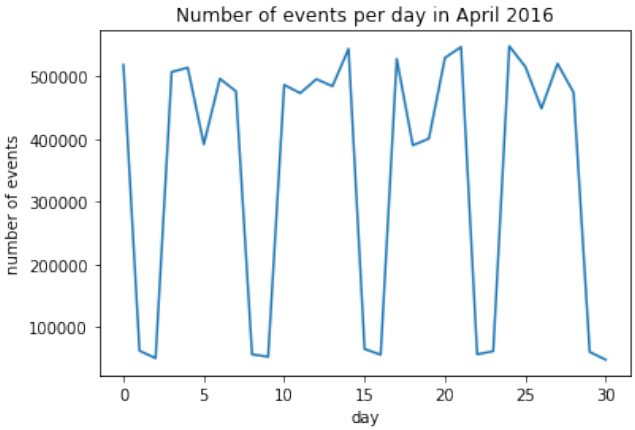
3.1.2 Data augmentation

In order to characterize user location more precisely, we augment the database of audit events. We use the free GeoLite2 City databases from Maxmind [151] which provide location information for IP addresses. From these databases we extract a list of IP address ranges with their corresponding country, region, city, latitude and longitude. Each audit event can then be matched to its corresponding IP address range in order to extract the location associated with the user IP address of the event. Note that this concerns only authentication events, as IP addresses are not logged at authorization time.

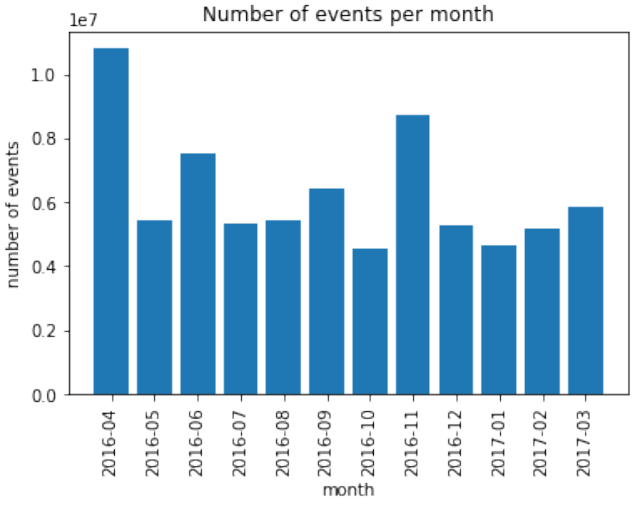
The audit dataset spans over a period of one year, during which assignments of IP address ranges change. For this reason we have used the Internet Archive [152] to retrieve several versions of the GeoLite2 City database (corresponding to March



(A) Number of events per hour



(B) Number of events per day during April 2016



(C) Number of audit events per month

FIGURE 3.3: Distribution of audit events per hour, day and month in the raw audit dataset.

2016, August 2016 and March 2017). For each authentication event, its location is determined using the most recent database available at the time of the event.

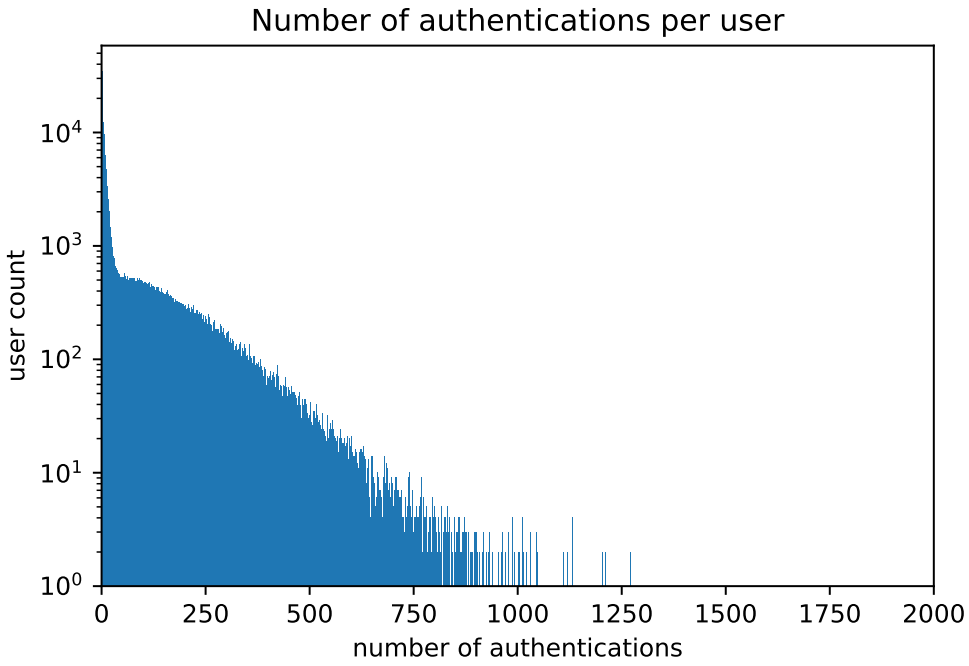


FIGURE 3.4: Distribution of the total number of authentication events per user.

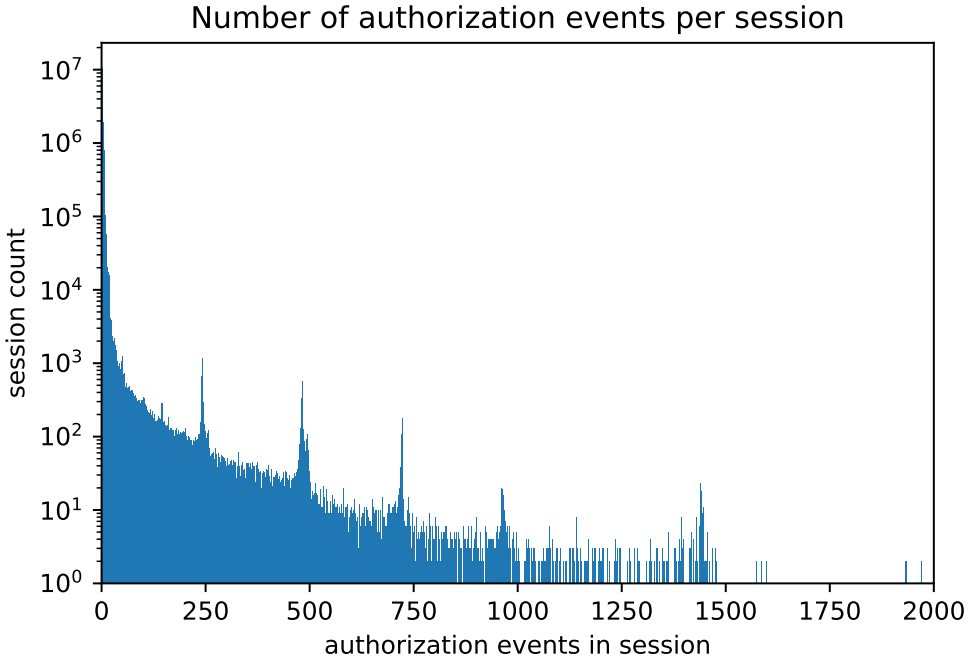


FIGURE 3.5: Distribution of the number of authorization events following authentication in the same session.

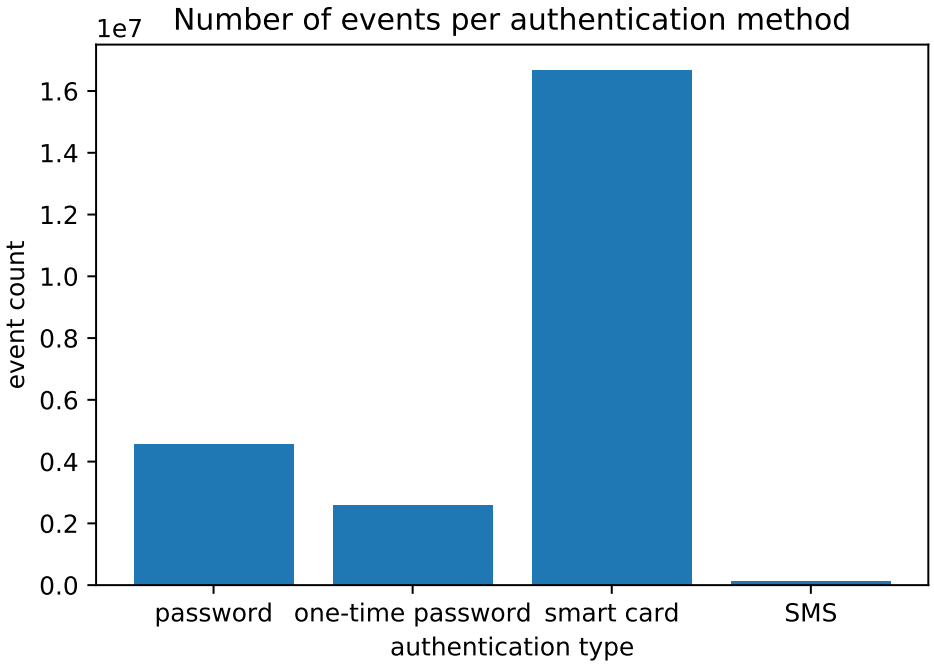


FIGURE 3.6: Distribution of authentication methods in raw audit dataset.

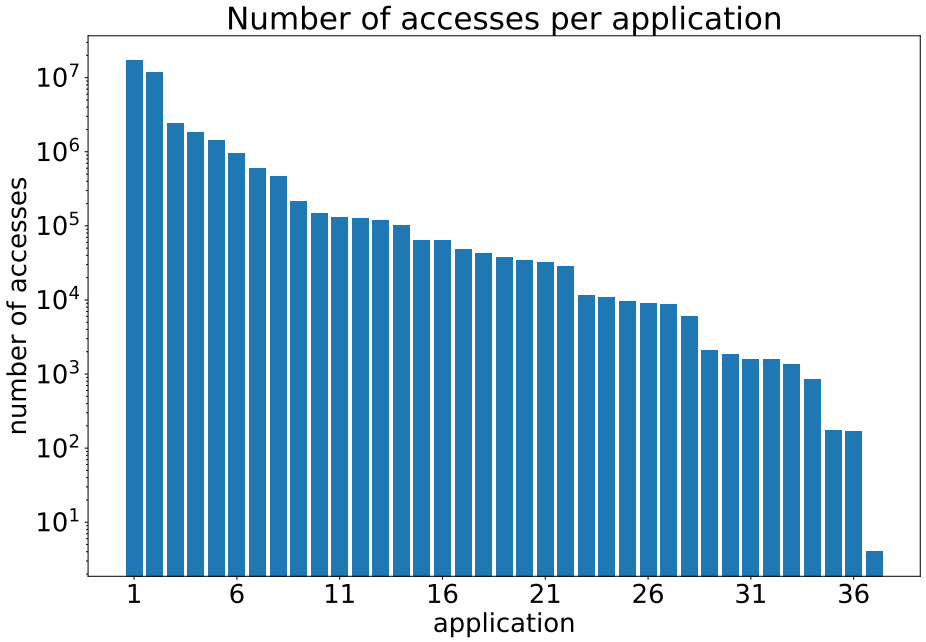


FIGURE 3.7: Distribution of application accesses in raw audit dataset.

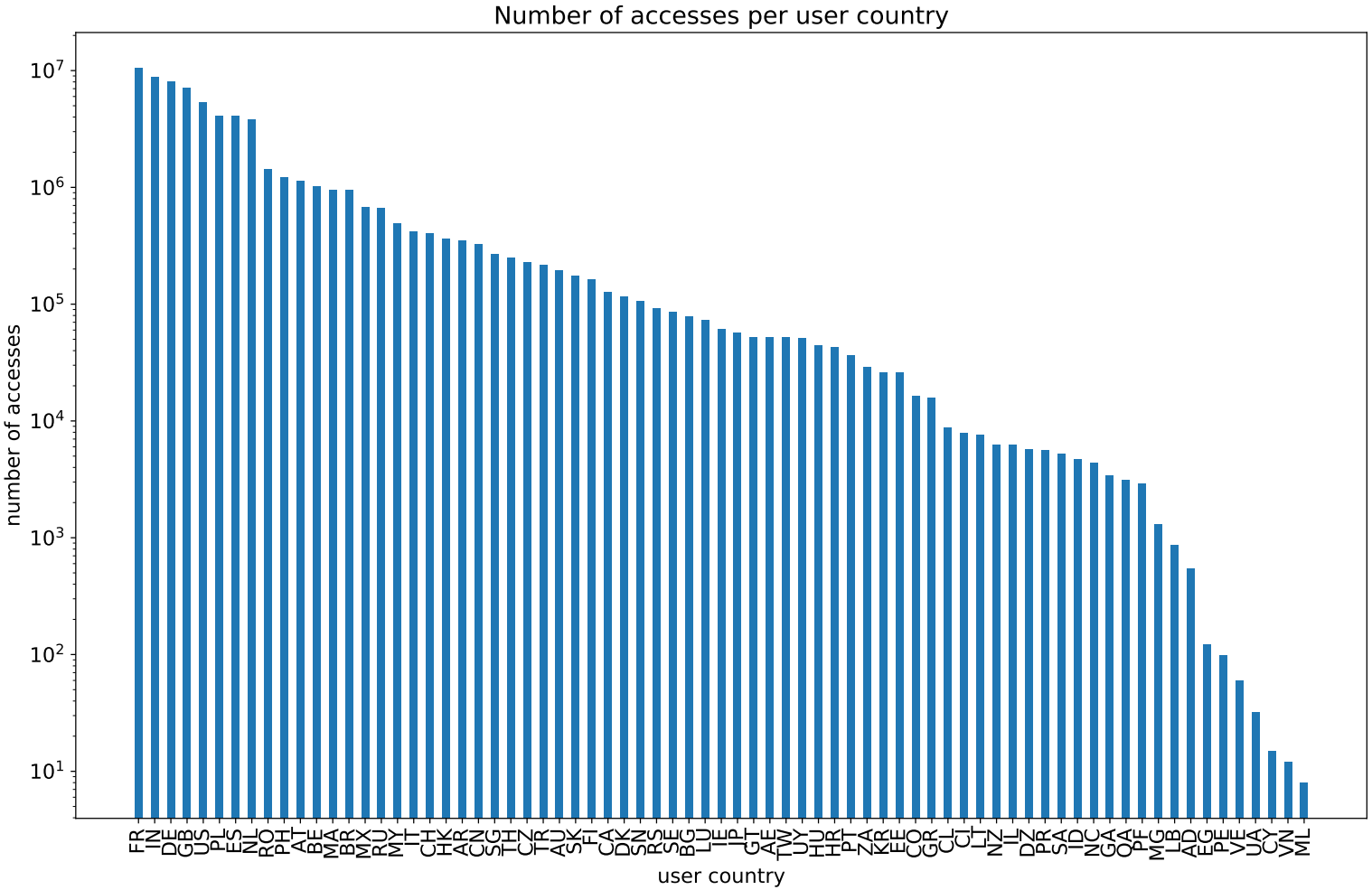


FIGURE 3.8: Distribution of user countries in raw audit dataset.

3.1.3 Session aggregation (audit-sessions dataset)

In the following, we describe why and how we aggregate authentication and authorization events into sessions.

Our first motivation is to enrich events to provide context, because events alone contain few relevant attributes, as shown previously. We argue that session aggregation is a simple yet powerful manner to do so. First, session aggregation is simple to realize technically, thanks to the session identifiers present in audit events, allowing to associate individual events to sessions. Second and more importantly, sessions represent an intuitive delimitation of user activities, for user intention is expected to be consistent within a session. Moreover, as audit events represent browser-based accesses, session aggregation is able to separate activities simultaneously performed in multiple browser instances, which would be impossible with aggregation based on time windows.

The first step for session aggregation is to collect successful authentication events, as they represent session beginnings. From these successful authentications, session identifiers can be retrieved, which in turn allows us to collect all events corresponding to said session. Using events within a session, we extract following features:

- user identifier,
- activity volume: number of events observed during each hour of the day and during the whole session;
- time features: session length, time of session start and end (month, week, day, hour, minute, second);
- application accesses: number of accesses to each application, access duration (until next application access);
- transitions between applications: number of transitions for each application pair and corresponding duration;
- time elapsed since last previous activity, last failed and last successful authentication for current user;
- usage of authentication methods: number of authentications with each method (several methods can be used within same session in case of multiple factor authentication);
- user country of employment;
- IP address at authentication time and its geolocation (country, region, city, latitude, longitude);
- and a boolean value indicating whether user country of employment matches country derived from IP address.

As users sometimes do not close their session properly (i.e. closing their browser before logging out), for each session we consider the most recent event to represent the session end and to compute the session length. Note that failed authentication attempts are discarded (except when computing duration since last activity and last failed authentication). This is because failed authentication leads to denied access thus no intrusion is possible. On the technical level, no session is created so session aggregation is meaningless in that case.

	Sessions with successful authentication	Sessions with failed authentication	Authentication failure rate
smart card	437923	32769	6,96%
password	57729	14349	19,9%
one-time password	31043	33771	52,1%
SMS	1277	3095	70,8%

TABLE 3.2: Usage of authentication methods and authentication failure rate in the audit-sessions dataset.

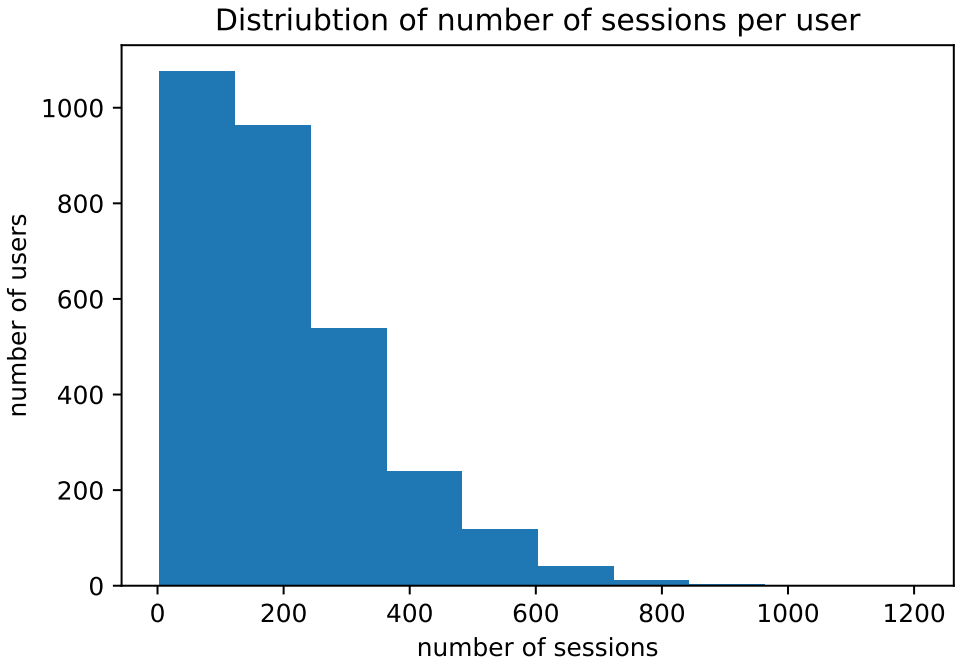


FIGURE 3.9: Distribution of the number of sessions per user in the audit-sessions dataset.

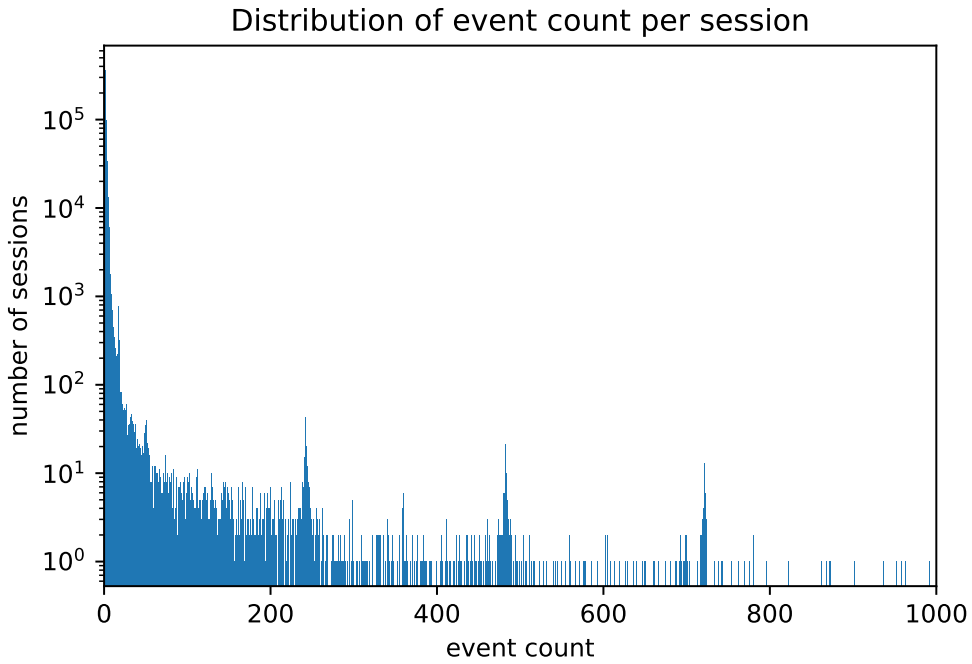


FIGURE 3.10: Distribution of the number of events per session in the audit-sessions dataset.

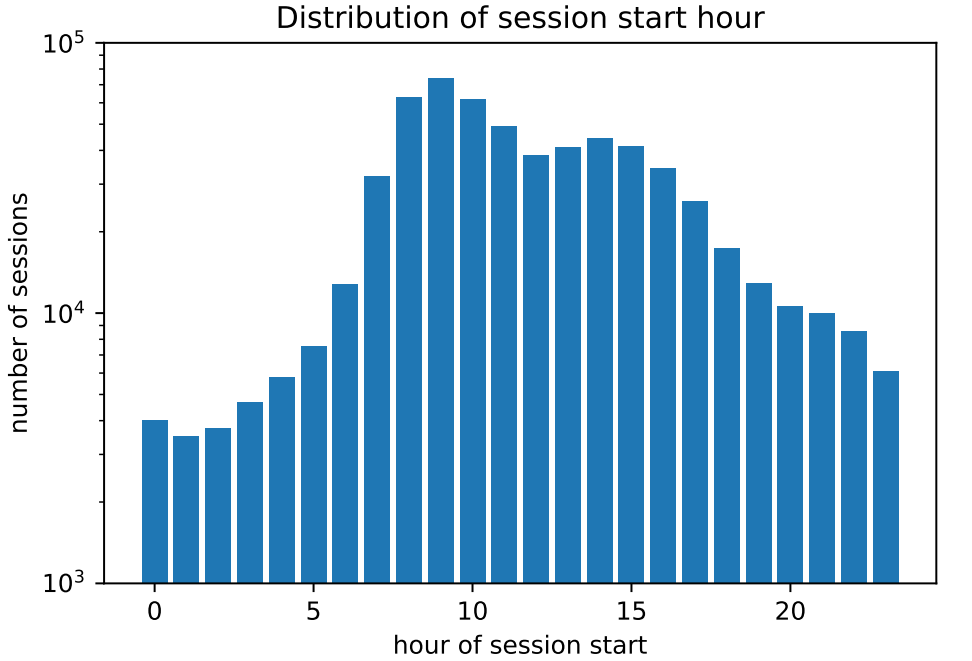


FIGURE 3.11: Distribution of the hour of session start in the audit-sessions dataset.

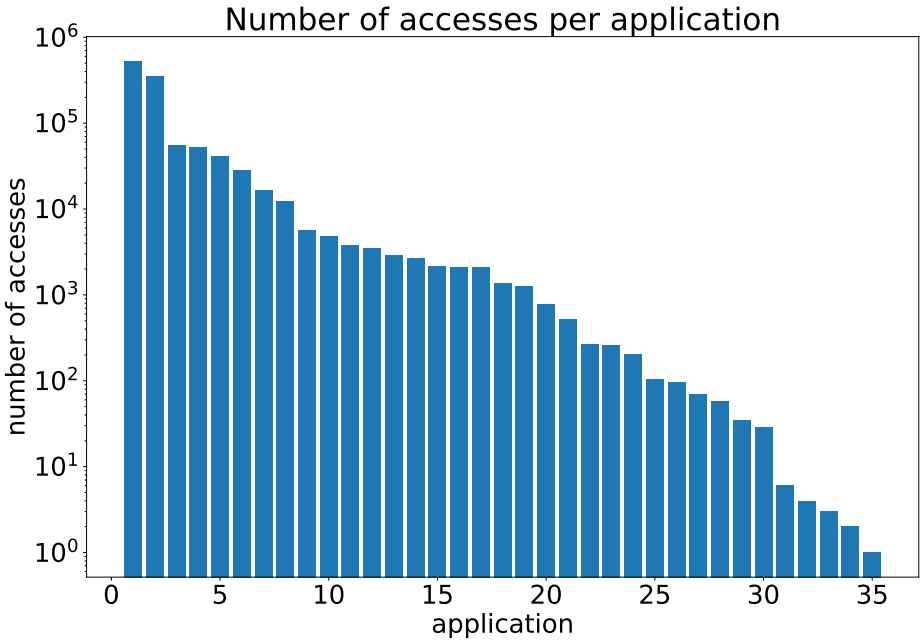


FIGURE 3.12: Distribution of application accesses in the audit-sessions dataset.

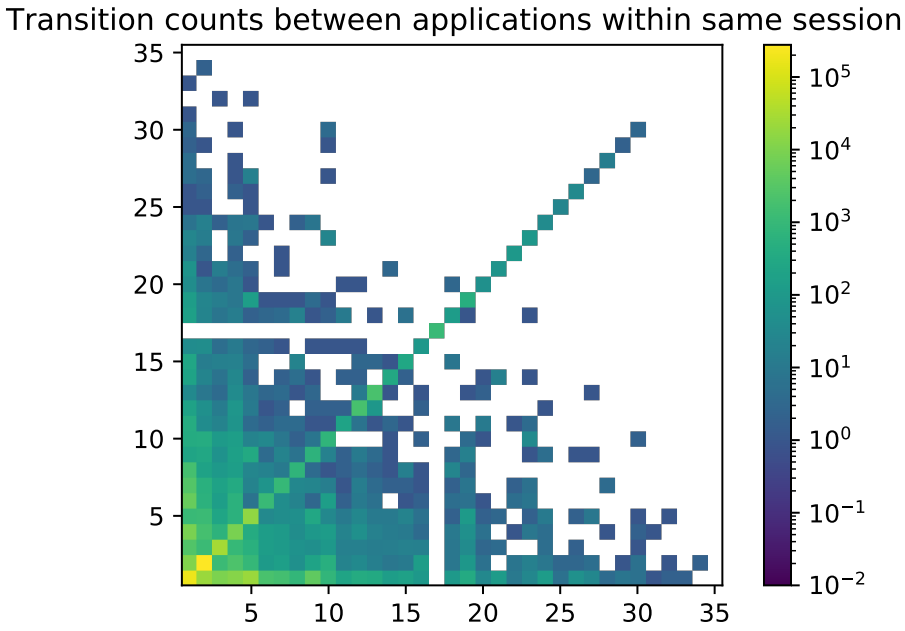


FIGURE 3.13: Number of transitions between application pairs in the audit-sessions dataset.

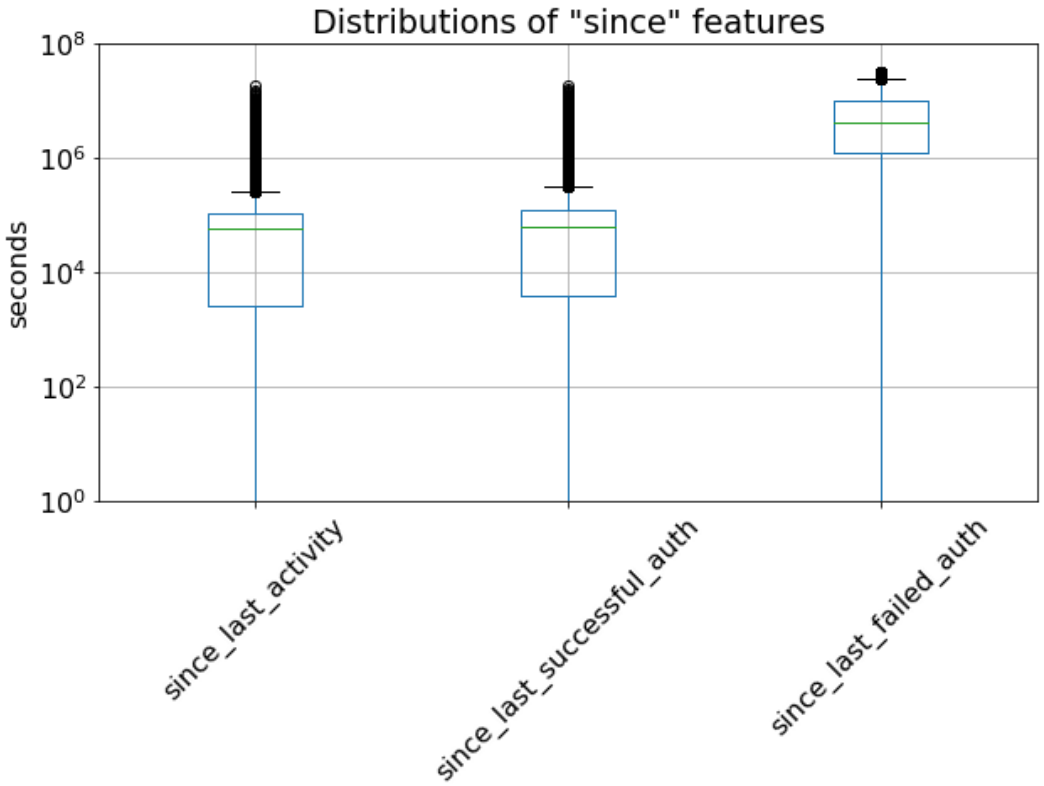


FIGURE 3.14: Boxplots for features representing time elapsed since last activity.

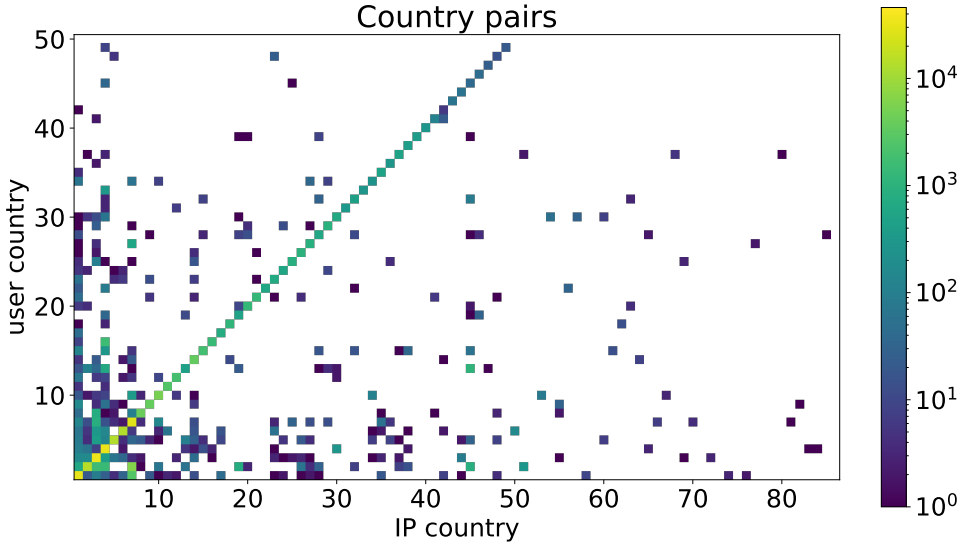


FIGURE 3.15: Distribution of user and IP country co-occurrences in the audit-sessions dataset.

In the following, we describe properties of the audit-sessions dataset, consisting of sessions from April to December 2016 as training set and sessions from January to March 2017 as test set, which represent around 612 000 sessions combined. This dataset contains activity for a subset of 3000 randomly chosen users among those with at least 20 sessions in the training set. Special user accounts like system administrators are excluded.

Figure 3.9 shows that most users have a few hundreds sessions. As shown on figure 3.10, most sessions consist of up to a few tens of events; sessions with hundreds of events are rarely observed. Figure 3.11 indicates that most sessions start between 7:00 and 18:00, with a peak between 8:00 and 10:00. Two applications represent the vast majority of all user accesses (see full distribution in figure 3.12, note the logarithmic scale). These two applications are the internal social network and the employee portal, as mentioned earlier. This prominence also appears in transitions between two applications, i.e. sequences of two successive application accesses within a same session. The occurrence of these transitions is displayed in figure 3.13. Most transitions concern one of the two most common applications, or occur within the same application (diagonal values in figure 3.13). As a consequence, application transition features are very sparse, since most pairs are very rarely observed. Figure 3.14 shows box plots for the following session features: duration since last activity of current user (at session start), since last successful authentication and since last failed authentication. The average duration since last activity is around 80 000 seconds, which corresponds to roughly one day. However, there is an important number of outlying high values, i.e. user accounts used far less often. Similar values are observed for the duration since the last successful authentication. However, the typical duration since the last failed authentication is significantly larger (around 45 days). As shown in table 3.2, authentication failure is relatively uncommon with the most widely used authentication method (smart card) but more common with other authentication methods. Distributions of countries associated with user accounts and IP addresses (logged at authentication time) are similar to the raw dataset (see figure 3.8) and are not included here. However the co-occurrence of these two country features is represented in figure 3.15. As expected, it shows the countries associated with user and IP address match in most sessions (diagonal line in figure 3.15).

3.2 CERT insider threat datasets

In the following, we describe the CERT insider threat datasets [57], [58], a collection of synthetic audit data sources released by Carnegie Mellon University's Software Engineering Institute to facilitate research on insider threat detection. The CERT datasets represent audit traces of users in a large organization and contain labelled examples of insider threats.

In chapter 4, we use these datasets to complete our evaluation on the Atos audit real-world dataset which unfortunately has no ground truth. Furthermore, in chapters 5 and 6, CERT datasets allow us to evaluate which heterogeneous data sources (graph, text) are relevant to intrusion detection and how to take them into account.

We first give an overview of available data sources (section 3.2.1) and insider threat scenarios (section 3.2.2) present in the CERT datasets. We then show how we aggregate different audit log sources to obtain sessions similar to the real-world audit dataset described previously (section 3.2.3).

3.2.1 Audit log sources

The CERT datasets include different audit data sources. In each data source, a log line represents an event or individual user action. We here describe events in the last version of the CERT dataset (6.2), used in chapters 5 and 6. Experiments in chapter 4 use version 4.2 of the dataset which has slightly different features.

All events contain the identifier of the user performing the action and the corresponding date and time. We list available audit data sources along with their additional attributes in the following:

- authentication logs ("logon") additionally contain for each event the used computer identifier and a value indicating login or logoff;
- email events contain the currently used computer identifier, email addresses of sender and receivers (split by email header field: "from", "to", "cc" or "bcc"), a value indicating whether the email is being sent or viewed, the size of the message, a list of attachments (file names and sizes) and the content of the message (text);
- web events represent browsing activities and contain computer identifier, URL visited, page content (text) and activity type (visit, download and upload);
- removable device usage logs ("device") represent a device being connected or disconnected to a given computer and includes the file system structure present on the device;
- file events are described by the name of the concerned file, an activity type (open, write, copy or delete), boolean values indicating whether this activity was performed from and to a removable device as well as the file content (text). Note that some files are decoys; the list of corresponding file names is also provided.

These data sources representing the activity of users are complemented with several static resources. In the context of this thesis, the most important is the LDAP repository representing the organization's hierarchy and user roles. Each user is described by a record indicating his user identifier, email, role (i.e. job title), the project and organizational units he is currently assigned to. Organizational units (in decreasing size order) are: business unit, functional unit, department, team and supervisor. The LDAP repository provides monthly snapshots of the organization, which allows to determine changes in the organization such as employees joining, leaving or being assigned to new projects.

Other static resources in the CERT datasets (though not used in our experiments) include a list of decoy files present on the organization's computers and psychometric user profiles representing personality traits.

3.2.2 Insider threat scenarios

The CERT datasets contain examples for five insider threat scenarios, each consisting of several malicious activities detectable across different audit data sources. Insider threat scenarios are described as follows in the dataset documentation [57]:

1. User who did not previously use removable drives or work after hours begins logging in after hours, using a removable drive, and uploading data to wikileaks.org. Leaves the organization shortly thereafter.

2. User begins surfing job websites and soliciting employment from a competitor. Before leaving the company, they use a thumb drive (at markedly higher rates than their previous activity) to steal data.

3. System administrator becomes disgruntled. Downloads a keylogger and uses a thumb drive to transfer it to his supervisor's machine. The next day, he uses the collected keylogs to log in as his supervisor and send out an alarming mass email, causing panic in the organization. He leaves the organization immediately.

4. A user logs into another user's machine and searches for interesting files, emailing to their home email. This behavior occurs more and more frequently over a three month period.

5. A member of a group decimated by layoffs uploads documents to Dropbox, planning to use them for personal gain.

Version 6.2 of the dataset contains one instance of each of the five scenarios, while version 4.2 contains multiple examples of scenarios 1 to 3.

3.2.3 Session aggregation (cert-sessions dataset)

In this section, we describe how we aggregate audit events from the CERT insider dataset (version 4.2) into sessions, as was done previously with the real-world audit dataset. The goal is to obtain a second evaluation dataset similar to audit-sessions described in section 3.1.3. A significant difference is that for cert-sessions, ground truth about intrusions is available.

However, the process of aggregating events into sessions is more complex than with the previous dataset because no session identifiers are present in the CERT dataset. Therefore session boundaries are determined by authentication events. A session consists of all audit events generated by a given user on a given computer between a login and the next logout event of said user on said computer. Based on session start (login) and end (logout), user and computer, corresponding audit events are retrieved and following session features are extracted:

- user identifier;
- computer identifier;
- time features: session length, time of start and end (month, week, day of week, day, minute, second);
- duration since last user activity (at session start);
- usage of removable devices: number of uses, average duration and total duration;
- web usage: number of pages visited, average duration until next page visit;
- file copies to removable devices: number of copies for each file extension and in total;
- email usage: number of emails sent for each receiver field (to, cc, bcc, all) and for each type of receiver (internal or external to the organization, as determined by email address domain), average and sum of email sizes, total number of attachments sent, ratio of attachment presence.

Anomaly labels (representing ground truth) are also extracted, a session is considered anomalous if it contains at least one malicious audit event.

A difficulty when assigning audit events to sessions is that some authentication events are missing in the CERT datasets (to represent noise in the audit data). For this reason it is possible to have a logout without preceding login. In that case the session will only contain the logout event, however the duration since last user activity is still set. The case where logout is missing (but login present) is less problematic: the last audit event in the session can be used to compute session length. Additionally, there can be several login events (corresponding to screen unlocks, as per the CERT documentation [57]) in a single session; in this case events are added to the session until logout occurs.

3.3 LANL authentication dataset

In addition to the synthetic CERT datasets, for evaluation of intrusion detection methods at event level presented in chapter 6, we use real-world authentication logs from Los Alamos National Laboratory (LANL) multi-source cybersecurity events [59], [60]. This repository contains different types of audit data sources such as Windows authentications, process traces, DNS data and network flows. These logs were collected from LANL's internal network over 58 days and represent the activity of over 12 000 users. We use only authentication logs, as other data sources do not contain ground truth about anomalies. In this context, anomalies are malicious authentications injected by a red team.

Our goal is to obtain a collection of authentication logs similar to the logon data source in the CERT datasets. The major problem with the raw LANL authentication dataset is that features of normal and anomalous events are not exactly the same. Indeed, anomalous events do not have all the features of normal events.

To solve this, we align attributes of normal and red team events by keeping only the timestamp, user, source and destination computer attributes. Normal authentications events have two user fields: source and destination user; we keep only the first one. This is justified by the fact that the values of these two attributes are the same most of the time: source and destination user only differ when user A (source) authenticates as user B (destination). In this case, by applying our preprocessing, A will be seen as B after such authentication (which is meaningful in the context of our application).

After feature alignment, we merge normal and red team events and remove authentications from special aliases and system accounts.

Chapter 4

Unsupervised intrusion detection through user identification

In this chapter, we address specific aspects of the real-world intrusion detection use case presented earlier in sections 1.2 and 3.1. From a high level perspective, our goal is to detect anomalous user behavior in a dataset of audit logs collected by an IAM (identity and access management) solution used to manage accesses to web business applications within a large IT company. Particularities of this task include a specific structure of audit records (authentication and authorization events, presence of numeric and categorical features) and more importantly the total absence of ground truth. An additional challenge is that this real-world dataset contains limited features to describe user activity (see section 3.1 for more details), hence it is unclear whether these attributes are informative enough to detect anomalous user behavior. These stringent constraints have lead us to define following general research questions:

RQ. 1: How to apply anomaly detection methods to real-world authentication and authorization audit logs containing mixed data and limited features?

RQ. 2: In total absence of ground truth, how to learn the distinction between normal and anomalous user behavior and evaluate the results?

We have previously described how to aggregate authentication and authorization events into user web sessions (section 3.1.3). Session aggregation allows to set context for individual audit events, whose attributes are limited, by grouping them into meaningful activity spans. Hence this preprocessing approach can be seen as a first step toward answering RQ. 1: by providing a well-defined, grounded method to represent user activity extracted from audit logs for intrusion detection purposes. Anomaly detection methods can now be applied to this representation.

However evaluating the outcome of anomaly detection methods is not possible without simultaneously addressing the problem of total ground truth unavailability (RQ. 2). Note that this goes further than unsupervised learning: in our case, labels are not only unavailable during the learning process, but also in the evaluation stage. Hence how can we check that activities flagged as anomalous indeed pose a security problem? Or to rephrase RQ. 2: how can we be sure that the distinction between normal and anomalous behaviors, as learned by some anomaly detection model, is meaningful?

Our general approach to answer these questions is to reformulate the intrusion detection task as user identification task. Given some audit session, we directly predict which user has generated this session, instead of assessing whether it falls within normal user behavior. Therefore this approach strongly relies on learning

the relation between user identifier (prediction target) and other attributes of audit sessions. This setting allows to turn an unsupervised problem into a supervised one. Because prediction targets are user identifiers, sound evaluation is possible. On the downside, this approach introduces its own new challenges (large number of classes and need for enough user examples), which are more widely discussed in section 4.3.

In section 4.1 we use the user identification proxy task to seek answers to following specific research questions:

RQ. a: Are available features informative enough to characterize user behavior, and distinguish one user from others?

RQ. b: Which features are useful to characterize user behavior in the real-world audit dataset?

After gaining insights on the real-world audit sessions dataset, we return to the unsupervised intrusion detection problem while keeping the user identification perspective in mind, focusing on following research questions:

RQ. c: How to turn user identification into intrusion detection?

RQ. d: How to evaluate intrusion detection models without ground truth?

These research questions are addressed in section 4.2, where we propose and evaluate methods for intrusion detection via user identification.

4.1 Audit session features for user identification

4.1.1 Problem setting

In the following experiments, our goal is to find answers to research questions a (feature importance) and b (user identification feasibility). We will do so by analyzing the outcome of a classification model trained to perform user identification on the audit-sessions dataset.

We start by formulating the user identification problem: our goal is to learn a prediction function f such that $f(session) = user$, where session is an audit session as described in section 3.1.3, without its *user* identifier (which is used as prediction target).

4.1.2 Methods

The previous problem definition as well as our application context bring several challenging requirements for the classification method used. First, it should support mixed data, as the audit-sessions dataset contains numeric and categorical attributes, which we want to take into account. Second, it should scale well with respect to the number of features (in the order of 1000 for audit-sessions). However, we cannot assume that all features extracted from sessions are informative. Hence the employed classification method should not be too affected by noisy or uninformative attributes. A moderately large number of target classes should also be supported (at least a few hundreds). Last but not least, the classification method must provide a simple way to evaluate feature importance, in order to provide an answer to RQ. b.

Given these requirements, we choose to resort to a random forest classifier [153]. Based on decision trees, random forest is suited for mixed data. As trees are built on random feature subsets, the classifier scales well with respect to the number of features while being relatively insensitive to uninformative ones. The random forest classification method is also sufficiently fast to handle some hundreds of target classes in reasonable computation time. According to Fernández-Delgado *et al.* [154], it ranks among top performing methods in a large classification benchmark over diverse domains. Finally, random forest provides a simple mechanism to evaluate feature importance via permutation.

Concerning this last point, we use permutation importance based on accuracy as introduced by Breiman [153]. For each feature (or group of features), we report the difference in accuracy between the model built on training data where the values of said feature are intact or randomly shuffled respectively. This allows to characterize how informative this feature is, with respect to the given classification task (here, user identification) using the random forest classifier. We also report scaled permutation importance, where the accuracy difference is expressed as ratio of the maximum accuracy value.

4.1.3 Evaluation setting

We use the random forest implementation from H2O (version 3.22.0.1) [155] with following hyperparameters: number of trees = 200, maximum depth = 5, maximum number of categories = 65535, one-hot categorical encoding set to "enum", score tree interval = 5, stopping rounds = 3 and histogram type set to "auto".

We use the audit-sessions dataset described in section 3.1.3. The period from April to December 2016 is used as training set and the period from January to March 2017 as test set. We choose to restrict ourselves to a limited number of users to assess the feasibility of the user identification approach. Hence for each experimental setting, we use sessions of n users randomly drawn from a pool of 1000 users. We then report results averaged over several runs.

Session features correspond to the ones described in section 3.1.3. Because computing feature importance for a large number of features is time consuming, we first report importance for groups of features. Features are split into following feature groups:

- application access counts ("app_count");
- application access duration ("app_duration");
- usage of authentication methods ("auth");
- count of transitions between application pairs ("bigram");
- duration of transitions between application pairs ("transition_delta");
- count of events per hour ("n_events");
- IP address and its derived geolocation ("IP");
- time since previous activity, successful and failed authentication ("since");
- event volume;
- session start and end time features ("session");

- user country identifier and whether it matches IP country ("user_country").

After having identified the most important feature groups, we report more detailed results for individual features in these groups.

4.1.4 Results

Figure 4.1 shows the user identification accuracy from 100 up to 900 users, as well as the distribution of per user accuracy for 100, 500 and 900 users. User identification accuracy is encouraging: it reaches around 0.7 for 100 users then drops (as expected) slightly below 0.5 for 900 users. The distribution of per user accuracy (corresponding to the number of rightly attributed sessions for each user) shows that although some users are not identified at all, the majority of them have some of their sessions correctly identified as the distribution spreads over all values between 0 and 1. Hence, user identification is feasible with reasonable accuracy in the audit-sessions dataset.

We now turn to feature importance. Figure 4.2 shows the feature permutation importance for groups of features when performing user identification with 100 users. Unsurprisingly, the most important feature group is the IP address and its geolocation, whose random permutation leads to losing around two thirds of the attainable accuracy. User country features are the second most important group (representing 21.8% of attainable accuracy) and "since" features come next (with 3.9% of attainable accuracy). Surprisingly, all other feature groups are very weak predictors for user identification.

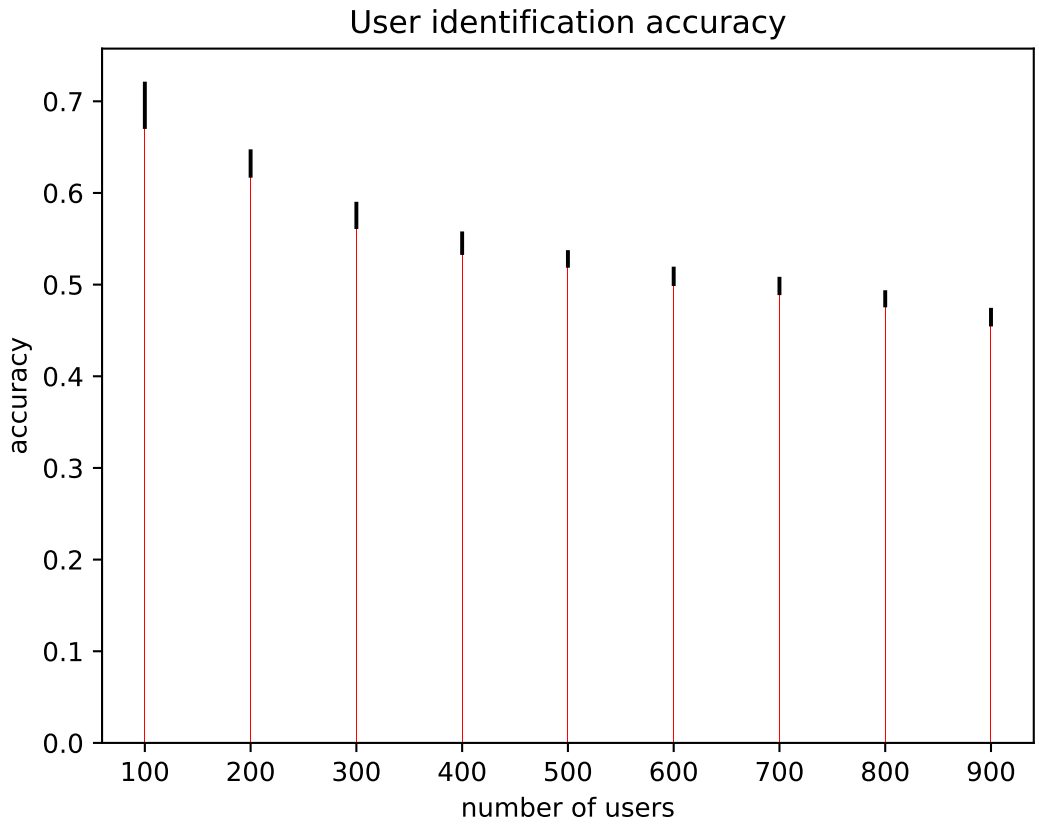
Figure 4.3 details the individual permutation importance of features within the three most important features groups: "ip", "user_country" and "since". This shows that the distribution of permutation importance is unequal among each feature group. The IP address, the user country identifier and the time since last failed authentication are the most important from the "ip", "user_country" and "since" groups respectively. This result is particularly surprising for the IP feature group, in which the contribution of the derived location (city, country and coordinates) is minor compared to the IP address itself. The figure also shows that when considering individual features, the feature importance gap between IP address and user country is not as large as when considering their respective feature groups.

As the IP address is by far the most important feature for user identification, how would the results change in case this feature would be unavailable or unreliable? For instance, in the Atos setting, this can happen if the user accesses applications over VPN connection.

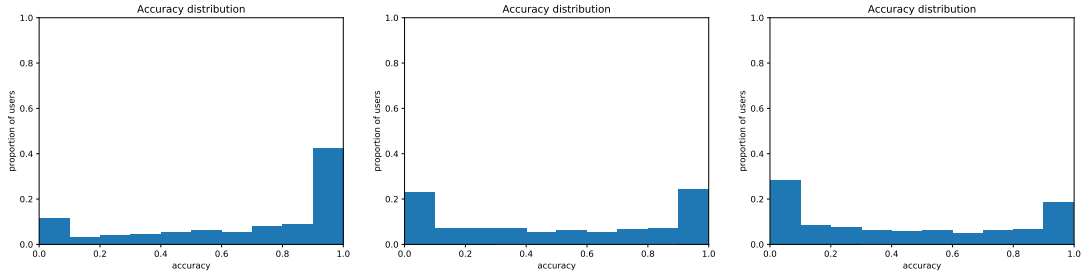
In order to answer this question, we compute feature permutation importance when including and excluding the IP feature group, for 100 and 500 users. Results are shown in table 4.1. When the IP information is removed, user identification drops significantly (from 0.70 to 0.47 with 100 users and from 0.52 to 0.20 for 500 users). At the same time, the random forest classifier relies much more on features from the "user_country" group, whose relative importance jumps (from the 14-22% range to around 77-78%). Other features groups see their relative importance increase as well, however not so dramatically.

4.1.5 Discussion

Based on previous results, we have gained important insights about the audit sessions dataset. We are now able to answer our two first specific research questions.



(A) User identification accuracy with respect to number of users



(B) Per user accuracy distribution, 100 users

(C) Per user accuracy distribution, 500 users

(D) Per user accuracy distribution, 900 users

FIGURE 4.1: A: User identification accuracy from 100 up to 900 users with 95% confidence intervals over 5 runs. B-D: Distribution of per user accuracy for 100, 500 and 900 users respectively.

Concerning RQ. a, user identification is indeed feasible using a random forest classifier: we obtain around 70% accuracy with 100 users, and around 50% with 500 users. This means that users can be distinguished by their behavior, as represented in audit sessions.

Regarding RQ. b, we have found that using a random forest classifier the IP address is by far the most informative feature for user identification. This is hardly surprising: within some limited time frame, the IP address can almost be seen as a personal identifier. Whenever this feature is unavailable, the user country attribute gains importance for user identification. More surprising is that other session features, and in particular those representing application usage, have very low importance in user identification.

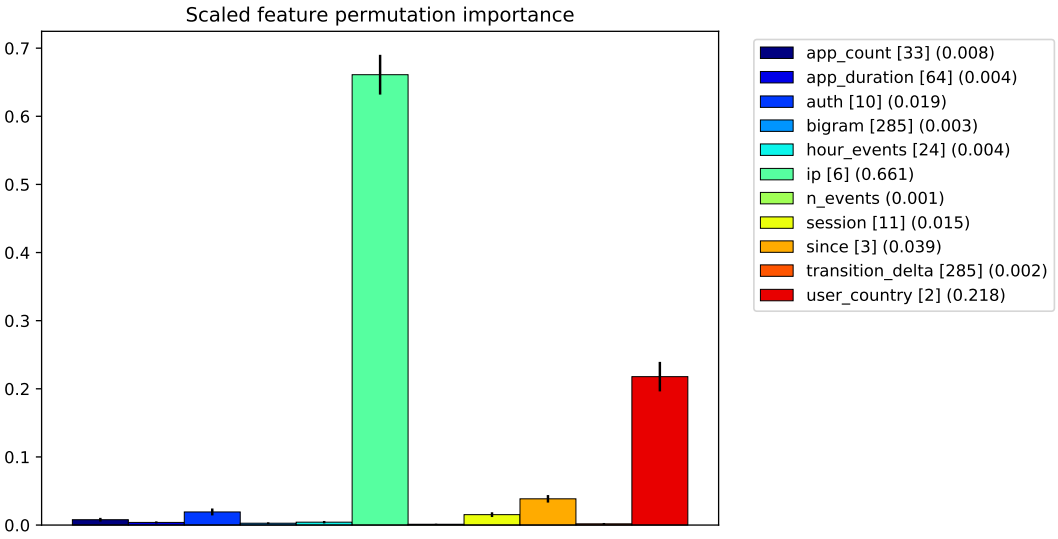


FIGURE 4.2: Average permutation importance with 95% confidence intervals for groups of features in the user identification task (n=100 users, 20 runs). Integers in brackets represent the number of features in each feature group. Numbers in parentheses represent feature group permutation importance as ratio of highest accuracy (when no feature is permuted).

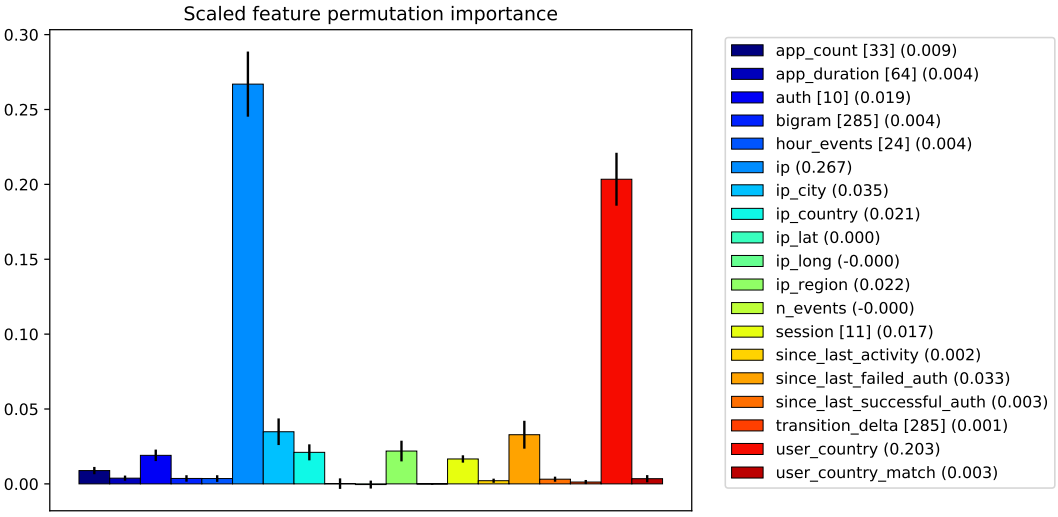


FIGURE 4.3: Average permutation importance with 95% confidence intervals for groups of features in the user identification task (n=100 users, 20 runs), detailing individual feature contributions for "IP", "since" and "user_country" feature groups. Integers in brackets represent the number of features in each feature group. Numbers in parentheses represent feature group permutation importance as ratio of highest accuracy (when no feature is permuted). Note that features are represented in same order on the chart (left to right) as in the legend (top to bottom), which may be helpful to distinguish features with low importance.

Feature group	Feature count	100 users (20 runs) with IP (acc = 0.70)	100 users (20 runs) without IP (acc=0.47)	500 users (5 runs) with IP (acc = 0.52)	500 users (5 runs) without IP (acc=0.20)
app_count	33	0.005 (0.8%)	0.021 (4.4%)	0.007 (1.3%)	0.016 (7.9%)
app_duration	64	0.003 (0.4%)	0.004 (0.9%)	0.003 (0.6%)	0.006 (3.2%)
auth	10	0.013 (1.9%)	0.042 (8.9%)	0.010 (2.0%)	0.022 (10.9%)
bigram	285	0.002 (0.3 %)	0.0005 (1.0%)	0.002 (0.5%)	0.006 (3.2%)
hour_events	24	0.003 (0.4%)	0.007 (1.5%)	0.003 (0.6%)	0.005 (2.6%)
ip	6	0.463 (66.1%)	-	0.440 (84.0%)	-
n_events	1	0.001 (0.1%)	0.000 (0.0%)	0.000 (0.0%)	0.001 (0.3%)
session	11	0.011 (1.5%)	0.033 (7.1%)	0.015 (2.8%)	0.024 (12.0%)
since	3	0.027 (3.9%)	0.052 (11.0%)	0.015 (2.8%)	0.019 (9.4%)
transition_delta	285	0.001 (0.2%)	0.002 (0.4%)	0.002 (0.4%)	0.004 (1.9%)
user_country	2	0.152 (21.8%)	0.367 (77.7%)	0.074 (14.1%)	0.158 (78.6%)

TABLE 4.1: Permutation importance of feature groups including and excluding the "IP" feature group, for 100 and 500 users.

It is worth mentioning that random forest permutation accuracy may overestimate the importance of categorical attributes with large number of levels (i.e. distinct values) [156]. This issue comes from the variable selection bias in random forest: features with higher number of possible splits are more likely to be used. In this regard, our results must be interpreted carefully. Feature importance measures presented here are only valid in the context of a random forest classifier and cannot be generalized to other methods.

A further limitation of our experiment lies in the limited number of users (up to 900) supported, which might be insufficient in some real-world deployment settings (e.g. the number of users in our real-world audit dataset is in the order of 10^5). This type of setting could be addressed by adapting methods from extreme classification (i.e. supporting very large number of classes), which we further discuss in conclusion of this chapter (4.3).

Nevertheless, user identification with random forest is viable for up to some hundreds of users; we next explore how to leverage this approach for intrusion detection.

4.2 Intrusion detection based on user identification

Our previous experiments have shown that in the audit-sessions dataset, sessions can be attributed back to their corresponding user through user identification. Now our goal is to use this perspective to perform unsupervised intrusion detection.

4.2.1 Problem setting

Anomaly ranking for intrusion detection in audit sessions

More precisely, we address the problem of unsupervised anomaly-based intrusion ranking. Given an audit record, our goal is to assign a score quantifying its anomalousness, i.e. ideally all intrusions should have higher scores than normal observations. Anomaly ranking generalizes the binary classification perspective as the sensitivity of the system can be adapted using a threshold, depending on application

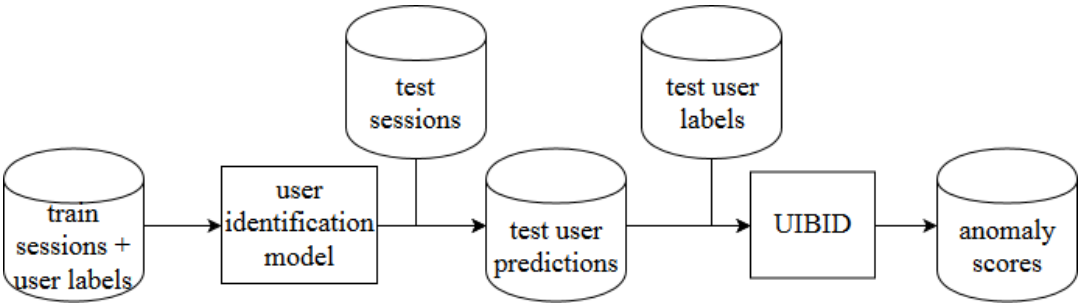


FIGURE 4.4: User Identification Based Intrusion Detection (UIBID). The anomaly score for a session is computed based on the probability of the **claimed user** being the author of the session.

domain requirements (mainly finding the right balance between false positives and false negatives).

We focus on user session records. A session s is composed of a user label u (the author of the session) and different activity features f_i: $s = \{u, f_1, ..., f_n\}$. As stated earlier in 3.1.3, sessions have natural activity boundaries (they start with login and end with logout) and therefore should represent consistent user activity segments. Based on this assumption we use sessions as elementary and indivisible audit records: they are either completely benign or completely malicious.

Leveraging the user identification perspective

We keep the user identification perspective introduced in 4.1. Thus we first train a user identification model to predict the discrete user label (user id) from a session record: $f(s) = u$, where s is a session record from user u (s does not contain a user label, which is the target of the classification task). Using the user identification model we can predict a probability distribution over users for a session: $p(u_1|s), p(u_2|s), ..., p(u_n|s)$ where $u_1, ..., u_n$ are all possible users.

4.2.2 Methods

In this section, we introduce two methods to derive intrusion detection anomaly scores from the user identification model. Hence we provide an answer to RQ. c: How to turn user identification into intrusion detection?

User identification based intrusion detection (UIBID)

Our first method, called User Identification Based Intrusion Detection (UIBID), is very straightforward. After building the user identification model, the anomaly score for a session is computed as follows:

$$score(s, cu) = 1 - p(cu|s). \tag{4.1}$$

where cu is the claimed user in session s and the probability p is obtained from the user identification model. Thus, the lower the probability that a session belongs to its corresponding user, the more anomalous it is. Figure 4.4 shows the full process of training the user identification model and using its predictions for intrusion detection with UIBID.

User cluster identification based intrusion detection (UCIBID)

Unfortunately, UIBID cannot cope with users having very similar sessions, for which it retrieves many false positives. Each session which cannot be attributed to its user with a high confidence will get a high anomaly score. However such sessions are not necessarily anomalous and corresponding false alarms should be discarded.

To alleviate this issue, we introduce User *Cluster* Identification Based Intrusion Detection (UCIBID), which relaxes the user identification problem. Instead of attributing sessions to individual users, UCIBID assigns sessions to clusters of users. Users are assigned to the same cluster if they are misclassified between each other, thus indirectly modeling session similarity. For instance, if users A and B are often misclassified between each other (many sessions of A attributed to B or inversely), then these users are assumed to be similar and therefore sessions of A attributed to B (and vice versa) should not be considered anomalous. By assigning both users to the same cluster, identification errors between A and B are tolerated and corresponding alarms will no longer be generated.

As pointed out in the example, UCIBID reduces detection sensitivity compared to UIBID: intrusions within the same user cluster can no longer be detected. We assume that such intrusions can be safely ignored because users in the same cluster (i.e. with very similar sessions) (a) cannot be distinguished from each other and (b) are expected to have similar access rights, thus making masquerade attacks unlikely. Another related assumption is that user clusters should be very compact: a high degree of similarity is required for users to be assigned to the same cluster. Therefore user clusters are expected to be small (in number of users) but numerous. Note that each user can be assigned to only one cluster or none (in the last case the user will be considered individually as in UIBID).

For UCIBID, the probability of a session belonging to a user cluster is computed as the sum of probabilities of users composing the cluster, allowing to re-use the existing user identification model. The anomaly score for a session is computed as follows:

$$score(s, cu) = 1 - \sum_{u \in C_{cu}} p(u)p(u|s). \tag{4.2}$$

where cu is the claimed user of the session, C_{cu} is the cluster of users containing cu and $p(u|s)$ is obtained from the user identification model. The term $p(u)$ represents the prior probability of the user and is ignored in practice (i.e. we assume a uniform distribution over all users). Figure 4.5 details the process of cluster construction and anomaly scores computation based on the user identification model in UCIBID.

User clusters are constructed from the confusion matrix of the user identification classifier. This classifier is assumed to optimize the distinction between users. Each row of the matrix represents one user as the identification prediction probabilities for all possible users. Thus each confusion matrix row can be used as a vector representation of the corresponding user. We perform clustering on these user vectors in order to obtain user clusters, hence characteristics of clusters are implicitly defined by the clustering process. Informally, due to the nature of user vectors, users are similar if they are often mistaken for one another or mistaken for a third common user by the user identification model (see figure 4.6).

4.2.3 Evaluation setting

In the following, we detail our evaluation setting in several aspects. First we introduce baseline anomaly detection models for mixed data. Second we explain

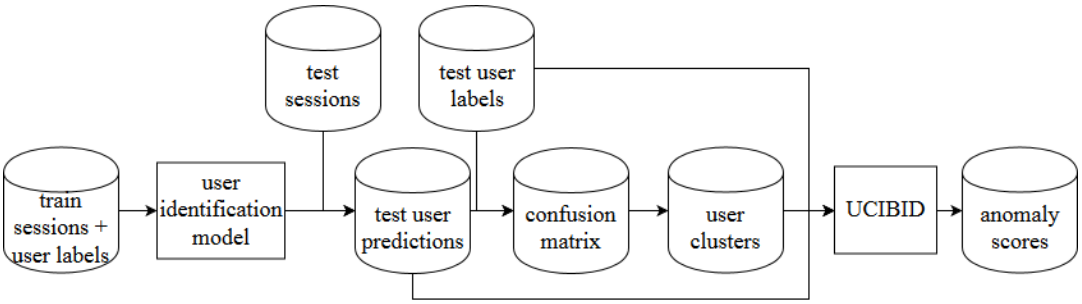


FIGURE 4.5: User Cluster Identification Based Intrusion Detection (UCIBID). The anomaly score of a session is computed as the probability of **any user in the same user cluster as the claimed session user** being the author of the session.

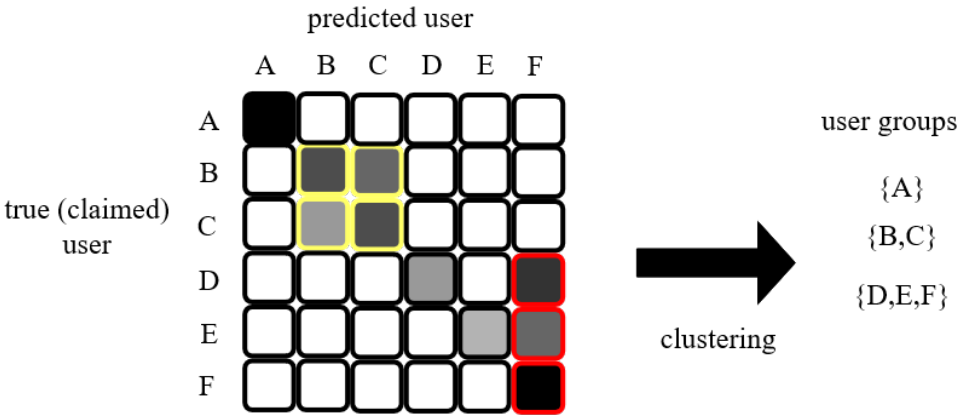


FIGURE 4.6: Clustering users represented as rows of the confusion matrix obtained from the user identification task. In this example, there are 6 users in total, light to dark shades represent low to high values in the confusion matrix. All sessions of user A are re-attributed to A (no user prediction error), thus A constitutes a cluster alone. Users B and C are partially misclassified for each other (yellow region) and are merged into a single cluster. Users D and E are assigned to the cluster of user F as they are often classified as such by the user identification model (red region).

our evaluation process for different intrusion types: masquerades (using the audit-sessions and cert-sessions datasets) and insider threats (cert-sessions only). Third, we describe classifier and clustering implementation details. Finally we present our evaluation metrics.

Anomaly ranking baselines for mixed data

As pointed out in section 2.2, the very specific structure of audit records makes the comparison of different intrusion detection systems very challenging. Due to the lack of intrusion detection systems operating on audit records similar to ours, we decided to compare our new methods UIBID and UCIBID to general unsupervised anomaly detection techniques.

An important constraint imposed by our datasets is that both numeric and categorical features should be supported, and that the anomaly detection algorithm scale to some hundreds of thousands of samples. We select following baseline methods:

- IndEnt [11],
- isolation forest [157],
- SPAD [90].

This choice is based on our preliminary benchmark of anomaly ranking methods for mixed data [11], in which we introduced two anomaly ranking methods based on entropy. We decided to keep only one (IndEnt) for the present evaluation as it obtained slightly better results in our previous benchmark. IndEnt relies on the individual entropy contribution of an observation to derive an anomaly score; the interested reader can refer to [11] for more details. A second baseline is isolation forest which has state-of-the-art performance according to [158]; we use our implementation for better support of mixed data described in [11]. Our last baseline is SPAD, introduced by Aryal, Ting, and Haffari [90], who claim that this method is as effective as state-of-the-art methods (including isolation forest) while running faster. SPAD and IndEnt have been shown to give similar results in our previous benchmark.

To allow a fair comparison of our user identification based intrusion detection methods with the general baseline models (IndEnt, isolation forest and SPAD), we use the latter as detectors in different configurations. As these methods are unsupervised, the first parameter is to use the training set or not: in the "train+test" setting, the models are fitted on the training set, while in the "test" setting models are fitted on the test set. The second parameter is whether to use one unsupervised anomaly detection model for all users or one model per user. Combining the configuration possibilities, we obtain four different intrusion detectors for each of the three baseline methods.

Datasets and intrusion types

We use audit-sessions and cert-sessions datasets for evaluation. Concerning the audit-sessions dataset (see section 3.1.3), we use sessions from April to December 2016 as training set, sessions from January to March 2017 as test set. As the random forest classifier used for user identification natively supports a moderate number of classes, we restrict our evaluation to a subset of 3000 users with at least 20 sessions in the training set. For the cert-sessions dataset (see section 3.2.3), we use sessions

Dataset	Unique users	Features	Train set records	Test set records	Intrusion rate (insider threats)
audit-sessions	3000	990	463K	148K	NA
cert-sessions	1000	70	150K	49K	train: $3.0 * 10^{-4}$ test: $6.9 * 10^{-3}$

TABLE 4.2: Characteristics of datasets used for evaluation of user identification based intrusion detection methods.

from January to June 2010 as training set and sessions from July to August 2010 as test set, for all 1000 users.

We first evaluate our methods in a masquerade detection scenario. As neither the audit-sessions nor the cert-sessions dataset contains masquerade samples, we resort to generating synthetic ones, similarly to Schonlau *et al.* [21]. Our assumption is that some activity of a user A should be considered as masquerade when conducted by another user B. In this regard we introduce synthetic masquerades by randomly changing the user label of some sessions so as to obtain an intrusion rate of 0.1%. As we cannot assume having access to an intrusion-free training set, both training and test sets are polluted with synthetic intrusions. Moreover, for cert-sessions the ground truth insider threats – which are a different type of intrusion – are not removed and considered normal observations, since they are indeed conducted by the rightful user. To summarize, in the masquerade detection scenario, our answer to RQ. d (How to evaluate intrusion detection methods without ground truth?) is: by generating synthetic intrusions.

Then we also evaluate our user identification based intrusion detection models on insider threat scenarios using the cert-sessions dataset, which has annotated intrusions. As it contains ground truth about insider threat activities conducted by users, we label any session containing at least one malicious event as anomalous. We use these labels to evaluate our methods for the insider threat scenario. Table 4.2 summarizes the number of unique users, features, total and anomalous records (for the insider threat case) in both datasets.

Classifier and clustering parameters

For the user identification model required by UIBID and UCIBID, we use a random forest [153] classifier with 100 trees, which we empirically determined to perform best. Note that any other supervised classification models could be used in general; nevertheless our datasets add two constraints in particular: (a) support of both numeric and categorical data and (b) scalability to a relatively large number of classes.

For UCIBID we perform clustering with scikit-learn [159] implementations of DBSCAN [160] (min_pts = 2, varying epsilon) and hierarchical agglomerative clustering [161] (average linkage, varying number of clusters). In both cases we use cosine similarity (commonly used for sparse data in high dimensional spaces) and report the best performing configurations.

Metrics

We compare all methods using area under the precision-recall curve (AUPRC), which is more informative than the ROC curve for strongly imbalanced problems like ours [85]. We perform cross-validation with 10 folds of 500 randomly selected

audit-sessions	user id. accuracy = 0.535 ± 0.010	
	overall AUPRC	mean AUPRC per fold
UIBID	0.055	0.092 ± 0.044
SPAD_test_per-user	0.004	0.004 ± 0.001
IndEnt_test_per-user	0.021	0.027 ± 0.014
ExtendedIsoForest_test_per-user	0.005	0.007 ± 0.003
UCIBID_dbscan_eps=0.384_minpts=2	0.102	**0.138 ± 0.061**
UCIBID_agg_120	**0.141**	**0.167 ± 0.033**
cert-sessions	user id. accuracy = 0.876 ± 0.008	
	overall AUPRC	mean AUPRC per fold
UIBID	**0.800**	**0.799 ± 0.042**
SPAD_test_per-user	0.014	0.016 ± 0.010
SPAD_train+test_per-user	0.044	0.054 ± 0.024
IndEnt_test_per-user	0.139	0.146 ± 0.041
IndEnt_train+test_per-user	0.004	0.004 ± 0.001
ExtendedIsoForest_test_per-user	0.222	0.233 ± 0.043
ExtendedIsoForest_train+test_per-user	0.051	0.065 ± 0.025

TABLE 4.3: User identification accuracy and masquerade detection scores (overall AUPRC, mean average AUPRC per fold with 95% confidence interval) for audit-sessions and cert-sessions datasets with 10 folds of 500 users.

users for each dataset. We report the overall AUPRC, computed from the full PR-curve over all, as well as the average AUPRC per fold and the user identification accuracy (proportion of sessions attributed to the claimed user). In the insider threat scenario, we similarly run methods on 10 folds of 500 users and report the area under their full precision-recall curve (AUPRC).

4.2.4 Results

Masquerade detection

For the masquerade detection setting, table 4.3 reports the overall AUPRC as well as the average AUPRC per fold and the user identification accuracy (proportion of sessions attributed to the claimed user). Figure 4.7 shows the full PR-curve, computed from all 10 runs.

On the audit-sessions dataset, UCIBID with agglomerative clustering (for n=120 clusters) performs best with 0.141 overall AUPRC. UCIBID with DBSCAN comes second with 0.102 overall AUPRC, although the difference is not significant when considering average AUPRC per fold. Both UCIBID models significantly outperform UIBID (0.055 overall AUPRC). Baseline methods SPAD, IndEnt and isolation forest are significantly worse. Results are only reported for the "test" configuration with one model per user, as other settings have poor performance (i.e. they do not perform significantly better than a model raising an alert for each session, which would achieve 0.001 AUPRC).

On the cert-sessions dataset, UIBID outperforms all other methods by far, with an overall AUPRC as high as 0.800. Results for UCIBID methods are not reported because we determined that the best configurations were those which lead to each user being assigned to a separate cluster, thus defaulting to the UIBID model. Concerning

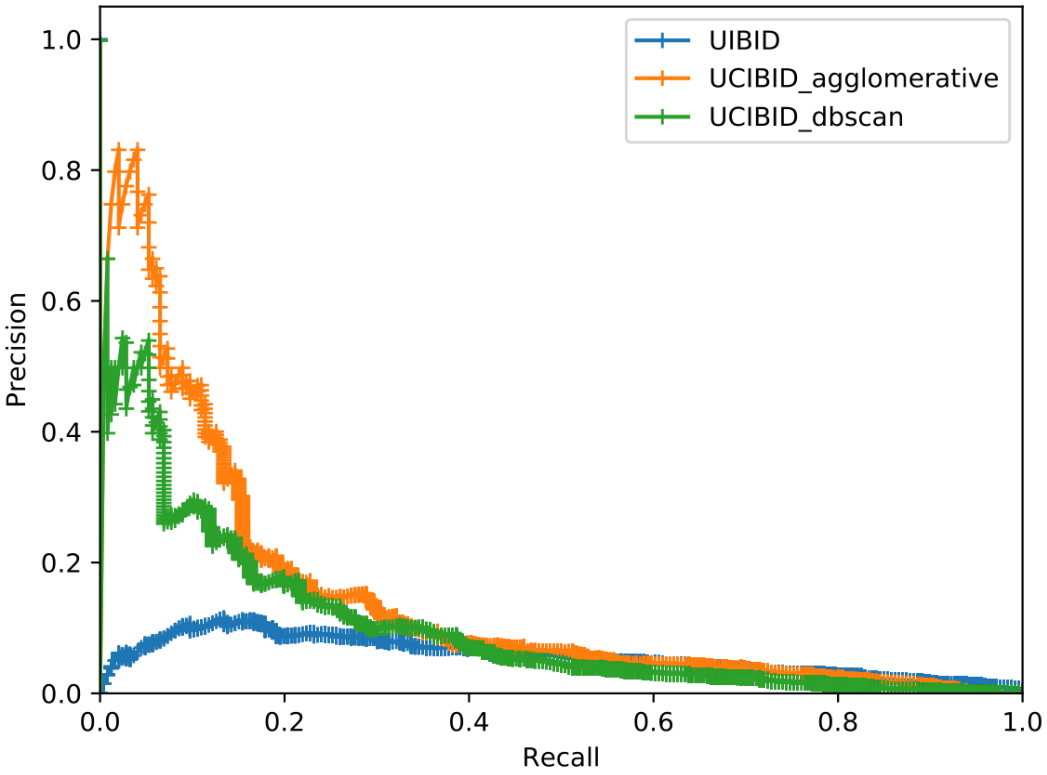


FIGURE 4.7: Precision-recall curves for masquerade detection in audit-sessions dataset for UIBID and UCIBID methods. UCIBID is able to reduce the false positive rate (higher precision at same recall level) compared to UIBID, particularly in the low recall domain (i.e. for most anomalous records).

	Intrusions by event type			
	device	email	http	logon
Test intr. rate	$2.10 * 10^{-3}$	$1.84 * 10^{-3}$	$5.08 * 10^{-3}$	$6.33 * 10^{-4}$
AUPRC				
SPAD	0.011	0.010	0.021	0.013
IndEnt	0.022	0.010	0.022	0.019
IsoForest	**0.033**	0.009	**0.030**	**0.124**
UIBID	0.012	0.003	0.009	0.025
UCIBID	-	**0.014**	0.019	0.029
	Intrusions by scenario			
	all	1	2	3
Test intr. rate	$6.85 * 10^{-3}$	$4.49 * 10^{-4}$	$6.10 * 10^{-3}$	$3.06 * 10^{-4}$
AUPRC				
SPAD	0.026	0.013	0.018	0.002
IndEnt	0.033	0.013	0.022	0.012
IsoForest	**0.035**	**0.118**	0.016	0.011
UIBID	0.012	0.034	0.008	0.001
UCIBID	0.024	0.039	**0.024**	**0.013**

TABLE 4.4: Insider threat detection performance (area under full precision-recall for all cross-validation folds) for all intrusions, intrusions split by type of malicious event and intrusions split by insider threat scenario.

the baseline models, the best performing configuration is "test" with one model per user for IndEnt (0.139 overall AUPRC) and isolation forest (0.222), whereas SPAD works best with the "train+test" setting (0.044 AUPRC). Average AUPRC per fold scores show a similar picture.

Insider threat detection

We report results for insider threat detection in table 4.4. For anomaly detection baselines only the results for the "train+test" configuration with one model per user are reported, as this strategy performs consistently better than others. Note that for UCIBID we report only the performance of the best model in each detection setting.

Overall, our new methods do not outperform the baselines on the insider threat detection task: while isolation forest gives 0.035 AUPRC, UIBID and UCIBID reach only 0.012 and 0.024 AUPRC respectively.

4.2.5 Discussion

It can seem surprising that UIBID is by far the best intrusion detection method in our benchmark on cert-sessions, while it is significantly worse than UCIBID for audit-sessions. An explanation to this phenomenon lies in the user identification accuracy, considerably higher for cert-sessions (0.876) than for audit-sessions (0.535). If users can be identified with a high accuracy, UIBID will perform better than UCIBID as it is more precise and sensitive by considering users individually. Indeed, attributing users A and B to the same cluster in UCIBID leads to masquerades of A among activity of B being undetected (i.e. false negatives), as long as they can be identified as A. The same is true for masquerades of B in activity of A.

On the contrary, UCIBID outperforms UIBID if user identification accuracy is average as it can reduce false positives due to systematically misidentified users, thus increasing precision for a same recall level (see precision-recall curve in figure 4.7). Moreover, the advantage of UCIBID over UIBID is significant in the low recall domain (i.e. left part of the curve, corresponding to most anomalous records) which is typically where IDS are mostly used. Overall, low AUPRC values must be considered in light of the difficulty of the problem, and in particular the very low intrusion rate.

Our results for insider threat detection are in line with the results of Emmott *et al.* [158] who have found isolation forest to be the best general anomaly detection method. The relatively low performance of our methods UIBID and UCIBID shows that they are rather adapted to masquerade detection than to insider threat detection. However scores of anomaly baselines are very low as well, suggesting a difficult problem.

Ground truth annotations in the cert-sessions dataset allow to characterize intrusions in a precise way. Indeed, for each intrusion session we know which type(s) of malicious events it contains and which insider threat scenario was carried out (see sections 3.2.1 and 3.2.2). Event type and intrusion scenario labels allow us to provide a more precise analysis of how well each type of anomaly is detected or not (see table 4.4). Note that detection performance for anomalous "file" events is not reported because of too few occurrences in the test set. For the "device" event type, performance of UCIBID is not reported as the best model defaults to UIBID (i.e. each user being assigned to a separate cluster).

Splitting the anomalies by event type shows that isolation forest has best performance to detect malicious "device", "http" and "logon" activity, whereas UCIBID outperforms other methods for anomalous "email" activity detection. This in turn explains the results split by intrusion scenario. For example scenario 1 involves the usage of removable drives after typical office hours, thus the best method for detecting such threats is isolation forest, which outperforms others in detecting anomalous "device" (characterizing removable drive usage) and "logon" (representing session time boundaries) events. Similarly, threat scenario 3 – involving massive email activity – is best detected with UCIBID. For scenario 2 the interpretation is less clear as it contains anomalous http, email and device events.

Overall, it is worth mentioning that *per user* anomaly detection models outperform global (*all users*) models, in both masquerade and insider threat detection settings. This means that these intrusions are best detected within a local context, i.e. by learning the behavior of an individual user – as opposed to finding global anomalies in the traces of all users.

4.3 Conclusion and perspectives

In this chapter, we have proposed to address the real-world intrusion detection use case in corporate audit logs from a user identification perspective. This approach has allowed us to gain first important insights about this problem and the corresponding audit-sessions dataset. Among others, major difficulties are the total absence of ground truth (intrusion labels) and lack of expert knowledge about the collected audit logs.

Through a supervised user identification approach permitting performance evaluation, we have determined that users can be distinguished based on their activity

traces, after aggregating these into web sessions. This partly answers RQ. 2; additionally synthetic masquerades can be used for evaluation. For user identification, we have found that IP addresses are strong predictors for the user identification task. However, this is not the case for other activity features. In particular, features representing application usage are not discriminative.

Subsequently, we have shown that user identification constitutes an interesting approach to intrusion detection as well. We have presented two methods to perform unsupervised intrusion detection in audit sessions through user identification. For each user the profile of normal behavior is represented implicitly within the user identification model. Our first method (UIBID) is very simple and turns user probability distributions into an anomaly score for user sessions. However UIBID raises many false alarms when users cannot be identified reliably from their audit sessions.

To address this issue we have proposed UCIBID, an extension of UIBID capable of clustering indistinguishable users in order to ignore false positives due to misidentifications among such users. User clusters are based on confusion matrix to indirectly model their similarity. One drawback is that users within a same cluster can no longer be distinguished, which however appears inevitable due to low feature discriminative power.

We have compared our methods to unsupervised anomaly detection baselines on the real-world audit-sessions dataset (thus answering RQ. 1) and the synthetic cert-sessions dataset. Results show that UIBID and UCIBID are particularly adapted to detect masquerades. UIBID works best when user identification rate is high, otherwise UCIBID is preferable. Overall, the user (cluster) identification perspective is suitable for masquerade detection, but insider threats require different methods.

A significant limitation of our user identification approach is that our current implementation only supports moderately large numbers of users (up to some hundreds). Thus an extension to support a larger number of classes will be required for the real-world intrusion detection case. In this regard, extreme classification methods [162], [163] could be adapted.

Work presented in this chapter only represents a first step toward understanding and addressing the real-world intrusion case. In this context, given our results we recommend building a simple rule-based classifier which would, for each user, classify known IP addresses as normal and the rest as anomalous. We expect this rule-based approach to constitute a strong baseline for this use case. However evaluating it is currently impossible due to lack of ground truth. Hence a substantial effort is required to gather domain knowledge and label the audit traces at hand, or at least some part of it.

Another important insight is that, except for IP addresses, the current log collection lacks informative features to characterize user behavior. In order to increase intrusion detection possibilities, more user activity traces should be logged. In this regard, focusing on the CERT insider threat use (for evaluation purposes), we address the search for additional useful features for intrusion detection in chapters 5 (RQ. 3) and 6 (RQ. 4).

Finally, methods presented in this chapter rely on the presence of sufficient audit data for each user, which might not be available in practice. Next chapter introduces a solution regarding this issue.

Chapter 5

Addressing the "cold start" problem with zero-shot learning

In chapter 4, we have addressed specific aspects of a real-world intrusion detection use case via user identification. This approach, like other anomaly-based intrusion detection methods, relies on user activity data to build profiles of normal behavior. However in some real-world situations, such activity data might be inherently unavailable (e.g. for users with no activity history). In this chapter, we present a solution based on zero-shot learning.

On a general level, zero-shot learning provides an extension of the supervised classification setting, allowing objects to be attributed to classes for which no examples were observed (i.e. unseen classes). To achieve this, an intermediary mapping is used to represent semantic properties of target classes. The classification of an object is classically performed in two steps: attribute learning and inference. First, in attribute learning, one has to determine which semantic properties are embodied by the object and to which degree; the result can be thought of as the position of said object in a multidimensional semantic space. Second, during inference, the best matching class is selected based on the position of the object in semantic space. Zero-shot learning is predominantly used in image classification, where text is classically used to extract semantic properties of classes (represented as word vectors called embeddings). We refer the interested reader to section 2.4 for a more formal introduction to zero-shot learning and its application to intrusion detection.

In the following, we use a zero-shot learning approach to improve an anomaly-based intrusion detection system. As such systems rely on the availability of audit log data to learn some representation of normal user behavior, they are subject to the "cold start" problem. Indeed, sufficient amounts of relevant activity data must be available to characterize anomalous behavior. The absence of activity traces can be quite problematic, as an organization cannot necessarily afford to wait until audit logs are collected before deploying protection. A second, slightly different variant of this problem, lies not in the absence but irrelevance of available data. Should normal user behavior change significantly at some point, then previous audit logs could no longer be used reliably to estimate future normal behavior and identify deviations thereof.

However these issues can be addressed through the zero-shot learning paradigm. The central idea is to rely on context data (i.e. independent from user activity) to estimate expected user behavior, when no previous activity logs are available (or if available logs are irrelevant). Hence in our problem, unseen classes correspond to users whose behavior we cannot estimate well using their activity logs. It is also worth mentioning that our system conceptually departs from the traditional zero-shot learning setting in two ways. First, it does not exactly perform classification: this would be similar to user identification as described in chapter 4, where the goal

is to predict which user has generated some activity trace. On the contrary, the intrusion detection system implemented hereafter can be seen as performing "soft" classification: given some user history and newly observed activity, it assigns anomaly scores to this last activity, which can be understood as the likelihood of the user (profile) given observed activity. Second, our system does not clearly separate the two attribute learning and inference stages; indeed it uses semantic class (user) descriptions in conjunction with observable examples (available activity logs).

On a more conceptual level, the zero-shot learning solution presented here complements the user identification approach presented in chapter 4, which relies on learning one behavioral profile per user. As we have seen, this can be challenging for a very large pool of users. However our zero-shot learning solution relies on contextual data to improve profile modeling, under the assumption that users with similar context (semantic description) will behave similarly. This relaxes the need to have a profile for each user and in this regard addresses very large user pools.

More concretely, we focus on the CERT insider threat use case [57], [58], which contains activity logs for users within a large organization and different labeled intrusion scenarios (see section 3.2 for further details). As described in section 3.2.1, the CERT datasets also contain information about the organization's hierarchy, user roles and their assignments to projects in the form of an LDAP repository. The content of this repository evolves over time as the organization's structure changes; the CERT datasets contain snapshot of the repository at beginning of each month for the whole period corresponding to activity logs. A closer look at the repository's evolution shows that two common types of changes occur: on one hand, some users leave the organization (though surprisingly, no new users are joining); on the other hand, users are successively assigned to different projects. In the following, we address two realistic scenarios in which anomaly-based intrusion detection systems suffer from unavailability of activity data:

- scenario 1: new users joining the organization;
- scenario 2: users changing their current project assignment.

Although there are no examples of new users joining in the CERT use case, this scenario is realistic and important, for it is observed in the life cycle of virtually any organization. Nevertheless it represents a significant challenge for anomaly-based intrusion detection due to unavailability of previous activity data. Within the CERT use case, it is possible that this scenario was omitted on purpose for simplification. The second scenario (project change) is already present in the CERT use case and corresponds to available but irrelevant logs of previous activities: our assumption is that change of project assignment leads to change of behavior.

Our primary goal is to propose and evaluate a solution for anomaly-based intrusion detection whenever activity logs are unavailable or irrelevant. Our approach to do so consists in using zero-shot learning principles to leverage contextual user attributes like project assignments and relations to organizational entities and users within the organization. Thus we address the general research question:

RQ. 3: How to address the "cold start" problem (i.e. absence of activity data) in anomaly-based intrusion detection?

Focusing on our concrete problem, our specific research questions can be stated as follows:

RQ a. How to obtain semantic class (i.e. user) representations useful to address the "cold start" problem in intrusion detection?

RQ b. How can the zero-shot learning paradigm be used in intrusion detection? Does it improve detection performance?

The rest of this chapter is structured as follows. In section 5.1, we propose and qualitatively evaluate different approaches to build a graph of user relationships to diverse organizational entities. Our goal is to model user similarity such that proximity in the graph will likely lead to similar behavior. Hence we partially address RQ. a, our emphasis being on which relations to select among available information. Then in section 5.2, we integrate these user relations into an existing intrusion detection system. We use graph node embedding methods to retain user proximity defined by the previously obtained graph. We benchmark our extended system against the corresponding baseline in the two scenarios presented earlier. Section 5.3 summarizes our findings and concludes the chapter with a broader view of perspectives opened by this work.

5.1 Construction of organizational graph

A requirement to integrate zero-shot learning capabilities into our intrusion detection use case lies in the availability of semantic descriptions of classes. In this section, we investigate how such representations can be adequately constructed in the CERT insider threat use case.

5.1.1 Problem setting

In our case, classes for which semantic representations are needed correspond to users or employees of the organization depicted in the CERT use case. As our goal is to use zero-shot learning to address scarcity of activity data, it is clear that these descriptions should be obtained from context data independent of user activity.

In this regard, the organization structure provided as LDAP repository in the CERT datasets constitutes a very useful resource. For each user, the CERT LDAP repository provides following relations:

- user's current role;
- user's current project;
- user's affectation to organization units (in order of increasing size): supervisor, team, department, functional unit and business unit.

Figure 5.1 gives an overview of these relations. In the following experiment, we set out to determine a way of constructing a graph of user relations in which user proximity correlates with (expected) similarity of working behavior observable in activity logs, based on the relations described in the LDAP repository. This graph will be later used to learn vector representations (embeddings) for users, corresponding to semantic class descriptions required for our intrusion detection system based on zero-shot learning.

In theory, determining the optimal graph construction can be seen as tuning yet another hyperparameter of the intrusion detection system; ultimately it should yield

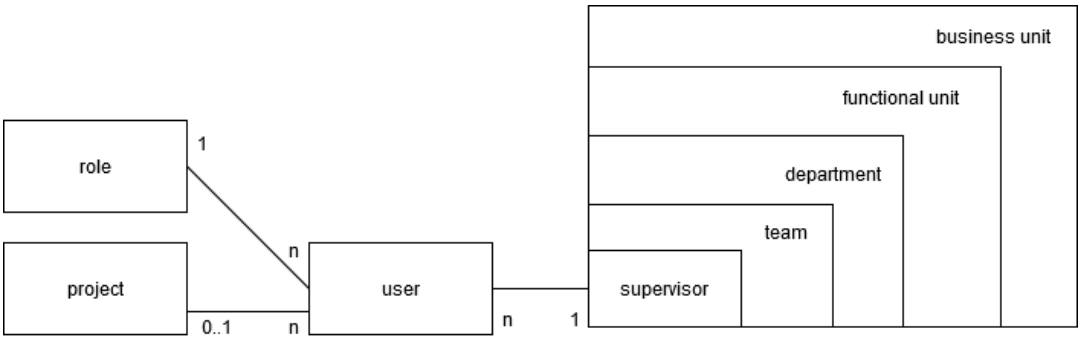


FIGURE 5.1: Entities and relations described in the LDAP repository of the CERT datasets.

Setting	Graph edges are all links from user to...
Project	role, supervisor, project, members of same project
Team	role, supervisor, team, members of same team
Department	role, supervisor, department, members of same department
Functional unit	role, supervisor, functional unit, members of same functional unit

TABLE 5.1: Settings for constructing the organizational graph of the CERT insider threat use case.

the best possible detection results. However in practice running and evaluating different graph constructions along with other hyperparameters is too costly in time. Moreover, using meaningful semantic class descriptions is key to our approach. In order to ensure this is the case, we carefully construct and qualitatively evaluate different graph constructions as described next.

5.1.2 Methods

We compare four different settings for constructing an organizational graph of users in the CERT dataset (version 6.2). We start by defining a baseline consisting of most specific relations: each user is connected to his closest collaborators (supervisor and members of the same project) and organization entities (role, assigned project). This baseline (named "project") represents this first graph construction setting.

We then define three other settings by progressively increasing the generality of relations to organization entities (i.e. by connecting a user to entities of increasing size). In the "team" setting project members are replaced by members of the same team, and a relation to corresponding team entity is added. "Department" and "functional unit" settings are defined similarly, by replacing relations of an organization unit with the parent unit (see figure 5.1). We omit the "business unit" as user assignment to these units are too general to be informative. Table 5.1 summarizes relations used in the four graph construction settings.

Note that all settings link users to their role and supervisor, which we expect to be highly relevant with regard to their working behavior. Nevertheless the four graph construction settings described above differ in which organization unit is used to define user relations, from most specific (project) to most general (functional unit).

We then apply existing graph embedding methods to obtain user vector representations preserving similarities from the graph, i.e. node neighborhood. The general goal of graph embedding is to map a graph (or its components, like nodes, edges

or substructures) to a space of low dimension while retaining the structure and/or relations defined by the graph. A large variety of graph embedding methods exist to address different requirements, depending on graph embedding input, output and learning strategy for the mapping [164]. In our experiment, we use node2vec [165], one of the most commonly used graph embedding method for nodes. Like the closely related method Deepwalk [166], it uses random walks to obtain sequences of adjacent nodes. Then the word2vec [111] algorithm is applied to obtain vector representations preserving node neighborhood, i.e. neighbor nodes are close in embedding space. According to its authors, node2vec outperforms Deepwalk by allowing more flexibility in random walks [165], which is why we prefer it over Deepwalk in this experiment.

Intuitively, applying graph embedding methods on a graph of very specific relations, we expect to get small user clusters in which behavior similarity should be high. However estimating the distance between two dissimilar users is problematic as there will be no path joining users from a different unit. On the contrary, using general relations we expect to link all users together to the detriment of loosing fine-grained similarities.

We evaluate the four proposed graph construction settings qualitatively from two perspectives, after applying a graph embedding learning algorithm to obtain vector representations for nodes. In this experiment we use node2vec with default parameters as provided by Grover and Leskovec [165]; however when evaluating the full intrusion detection system we compare different embedding methods. The first evaluation perspective consists in verifying that the nearest neighbors of some nodes (as defined by their embeddings) correspond to the expected LDAP relations. The second consists in projecting the embeddings obtained with node2vec using t-SNE [167]. By visualizing these projected vector representations we want to verify that the user similarities defined by the relations from LDAP repository are preserved.

5.1.3 Results and discussion

Table 5.2 summarizes the nature of nearest neighbors obtained from node2vec embeddings of different nodes. A quick look at neighbor nodes of units (middle column) shows that similar entities in embedding space are members of the corresponding unit, hence relations are preserved as expected. The same is true for role relations (last column), for which similarities in embedding space allow, given some role, to retrieve members with this role, related roles and members of units where this role is common.

However our main interest lies in neighbors of user nodes (first column), as they are the entities for which we require semantic vector representations. Using the "project" graph construction, neighbors of a user in embedding space are either members of the same project or have the same role. In the "team" setting, user neighbors are mostly members of the same team and working on the same project. Thus in both project and team graph, fine-grained work relations are preserved. Using links to larger units than team (department, functional unit), the obtained graph (and vector representations extracted from it) do reflect membership to the corresponding unit, but user neighbors are no longer on the same team or project. Thus organizational structures of smaller scale are lost, and fine-grained relations with them.

Visualization of learned embedding representations for nodes confirm these first impressions. t-SNE visualizations of node2vec embeddings obtained in the project,

Setting	Graph node used to inspect neighbors: User	Unit (= setting)	Role
Project	members of same project or same role	project members, users with same role	role members related roles
Team	members of same team and same project, or same department	team members	related roles, members of unit containing role
Department	members of same department and role, but different projects	department members	members of unit containing role
Functional unit	members of same functional unit, but different dpt/team/project	functional unit members	role members, related roles, members of unit containing role

TABLE 5.2: Neighbors retrieved from graph embeddings in the different graph construction settings.

team, department and functional unit settings are shown in figures 5.2, 5.3, 5.4 and 5.5 respectively. When user memberships to large organizational units are used to construct the graph (department, functional unit), we obtain clusters representing precisely these units. Unfortunately, this level of granularity is too high for our purpose, since we want to relate users to their closest collaborators. Thus we should use the "project" or "team" graph construction settings. Visual inspection of embeddings shows that small clusters are obtained in both settings. However, team relations seem to lead groups of more consistent size. Unlike team assignments which are rather constant, project assignments evolve over time. Some users can be temporarily free of any project assignment, which explains the "out of cluster" nodes visible in figure 5.2.

In conclusion, it appears that the position of users within the organization described in the CERT datasets can be characterized by a combination of their links to their role, supervisor, project colleagues and team colleagues. Using graph node embedding techniques like node2vec, we can obtain a vector representation preserving these links. Our expectation is that this representation characterizes user working behavior and thus can be used as semantic description of user classes for zero-shot learning.

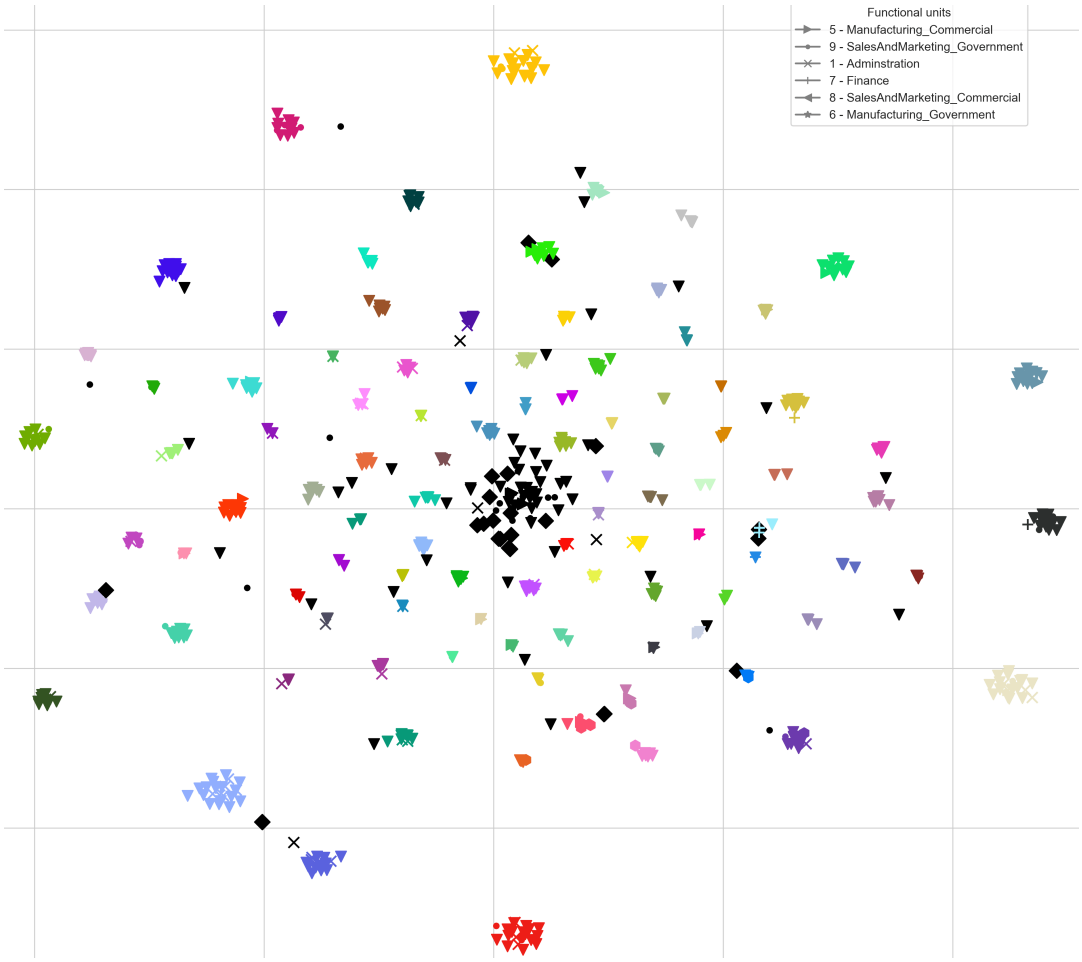


FIGURE 5.2: 2D t-SNE projections of node2vec embeddings obtained in the "project" graph construction setting. Colors represent projects (non-user nodes in black) while markers represent functional units. Note that the top right legend contains only a subset of all functional units for brevity. Figure taken from [168].

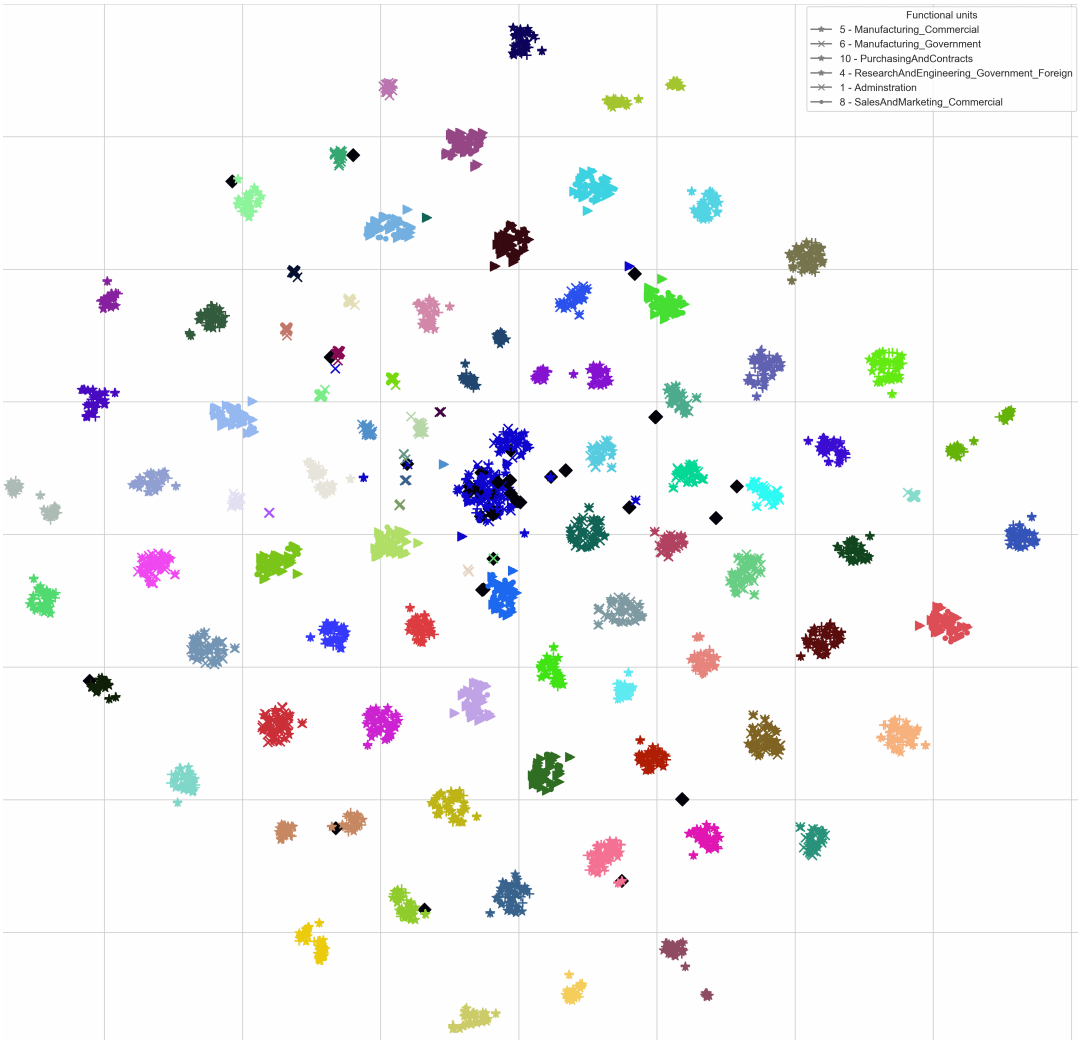


FIGURE 5.3: 2D t-SNE projections of node2vec embeddings obtained in the "team" graph construction setting. Colors represent teams (non-user nodes in black) while markers represent functional units. Note that the top right legend contains only a subset of all functional units for brevity. Figure taken from [168].

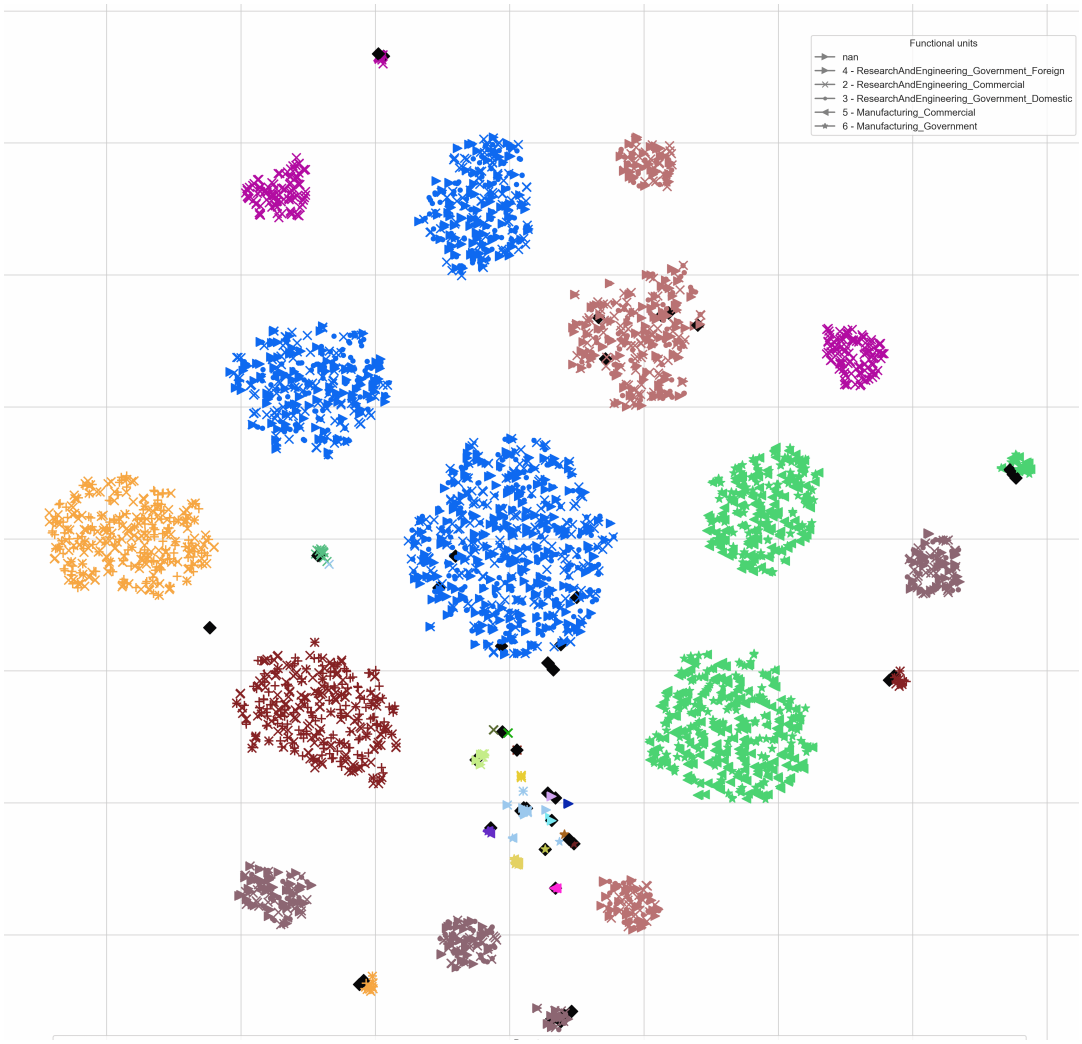


FIGURE 5.4: 2D t-SNE projections of node2vec embeddings obtained in the "department" graph construction setting. Colors represent departments (non-user nodes in black) while markers represent functional units. Note that the top right legend contains only a subset of all functional units for brevity. Figure taken from [168].

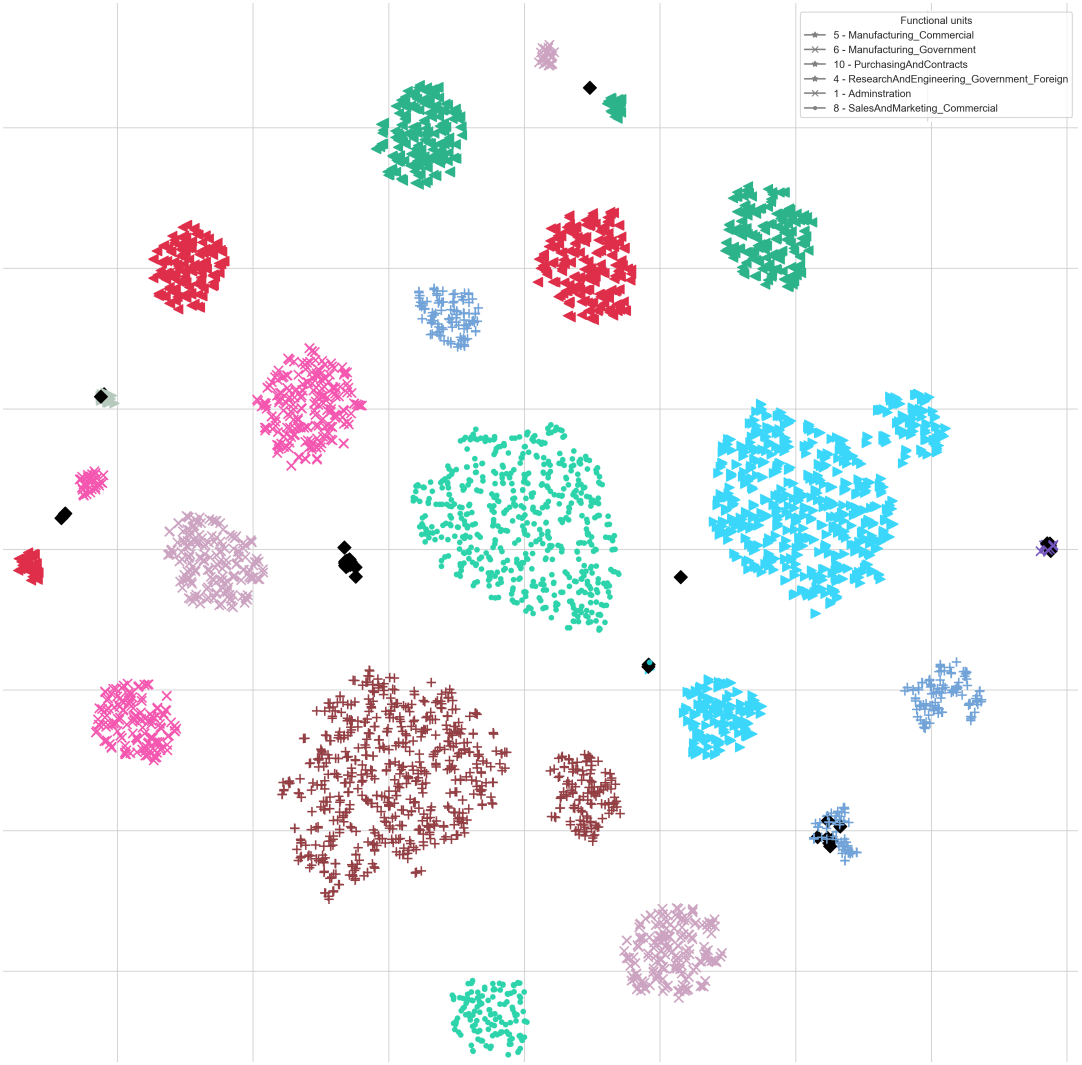


FIGURE 5.5: 2D t-SNE projections of node2vec embeddings obtained in the "functional unit" graph construction setting. Colors and markers represent functional units (non-user nodes in black). Note that the top right legend contains only a subset of all functional units for brevity. Figure taken from [168].

5.2 Improving an intrusion detection system with zero-shot learning

In the previous analysis, we have given an overview of contextual attributes representing the organization's structure in the CERT datasets and shown how they can be used to build semantic representations of users via graph embeddings.

In this section, we describe how we extend an existing intrusion detection system in zero-shot learning fashion using these semantic representations. We then evaluate our system in two scenarios where activity data is unavailable or irrelevant.

This section is structured as follows. First, we introduce our problem setting more precisely in section 5.2.1. We then describe the baseline intrusion detection system and present our zero-shot learning extension in section 5.2.2. After detailing our evaluation setting (section 5.2.3), we present experimental results (section 5.2.4) and discuss our findings (section 5.2.5).

5.2.1 Problem setting

In the following experiments, we address intrusion detection by focusing on the CERT insider threat use case. That is, we seek to detect threat scenarios described in section 3.2.2 using the diverse audit data sources provided in the CERT datasets (see section 3.2.1). However our method for extending intrusion detection systems with zero-shot learning described hereafter is not specific to this use case and could be used in other settings. In that regard, the work presented in the following consists of a possible implementation for the CERT insider threat use case, nevertheless its applicability is not limited to this specific context.

As in the rest of this thesis, we address intrusion detection through an anomaly ranking approach. More precisely, our goal is to compute anomaly scores for fixed time frames of user activity. In particular, the length of such time frames is chosen to correspond to one day, and we extract features describing the actions performed by each user during this period. Hence in the rest of this section, observations processed by the intrusion detection systems are referred to as "user-days".

Therefore detection level differs from the methods presented previously in this thesis, which use session-level detection (chapter 4) or event-level detection (chapter 6). The main reason is that our zero-shot learning intrusion detection system extends the framework introduced by Tuor *et al.* [76]. As their system constitutes our baseline, we keep their evaluation setting (including the user-day aggregation) schema to allow performance comparison. Moreover, we believe that detection techniques at different granularity levels should be seen as complementary rather than competing (this point is discussed more deeply in chapter 7).

5.2.2 Methods

In this section, we describe two intrusion detection systems used in our experiments. First, we present the intrusion detection system introduced by Tuor *et al.* [76], which we use as baseline. Second, we detail our extension of this system through zero-shot learning with the aim to improve user behavior prediction, and therefore intrusion detection.

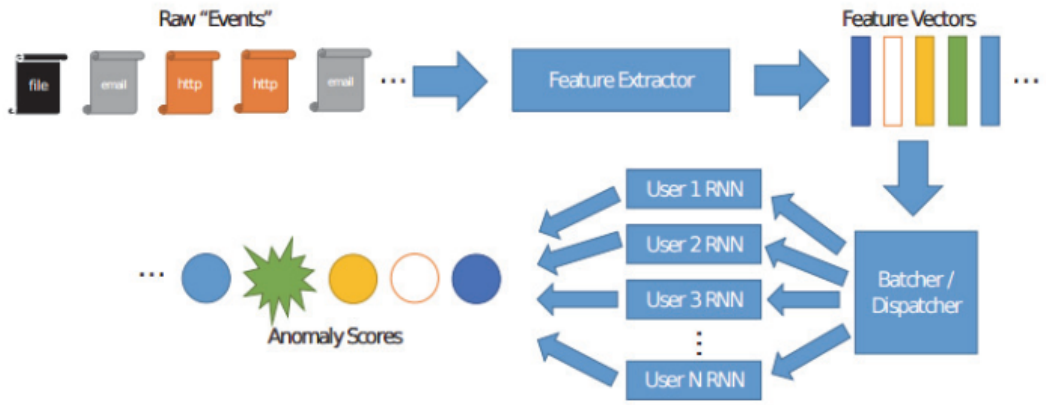


FIGURE 5.6: Overview of the baseline intrusion detection system introduced in [76]. The feature extractor aggregates audit events into user-days by computing features described in figure 5.7. Figure taken from the original paper.

Baseline

In the following experiments, we use a slightly modified version of intrusion detection system proposed by Tuor *et al.* [76] as baseline. Figure 5.6 shows an overview of this system.

First, raw audit events from the CERT insider threat dataset (see section 3.2.1) are aggregated into "user-day" feature vectors. Each vector represents the activity of one user during one day. The exact list of features describing one user-day can be found in figure 5.7. However the original paper does not describe feature extraction in a fully reproducible way: In particular, what corresponds to "uncommon" and "common" web pages, email correspondents and computers is not explicitly specified. Our re-implementation of the pre-processing step defines events as common if more than n occurrences are observed in the training period (n is set to 5 in the experiments presented here); other events are considered uncommon.

Another difference is that we also use following attributes available from the LDAP descriptions in the CERT datasets: role, project, functional unit, department, team and supervisor of each user. These features are simply encoded as categorical values. Finally, we concatenate categorical values (describing the user) and numeric values (describing his activity during each day) to obtain feature vectors as shown in figure 5.8.

Feature vectors are then fed into an auto-encoder neural network, i.e. hidden layers are of smaller dimensionality than the input and the network is trained to reconstruct the input at its output layer. Although figure 5.6 indicates that a recurrent neural network is used, the original implementation also provides a feed-forward network as auto-encoder. We use the feed-forward network as it is simpler and faster to train, and gives similar detection performance according to Tuor *et al.* [76]. This model takes as an input a series of T feature vectors $x_1^u, x_2^u, ..., x_T^u$ for a user u and outputs a series of T hidden state vectors $h_1^u, h_2^u, ..., h_T^u$ after passing through each hidden layer l with non-linear activation function g according to:

$$h_{l,t}^u = g(W_l h_{l-1,t}^u + b_l) \tag{5.1}$$

where W_l and b_l represent the parameters of layer l. The model is trained over all users.

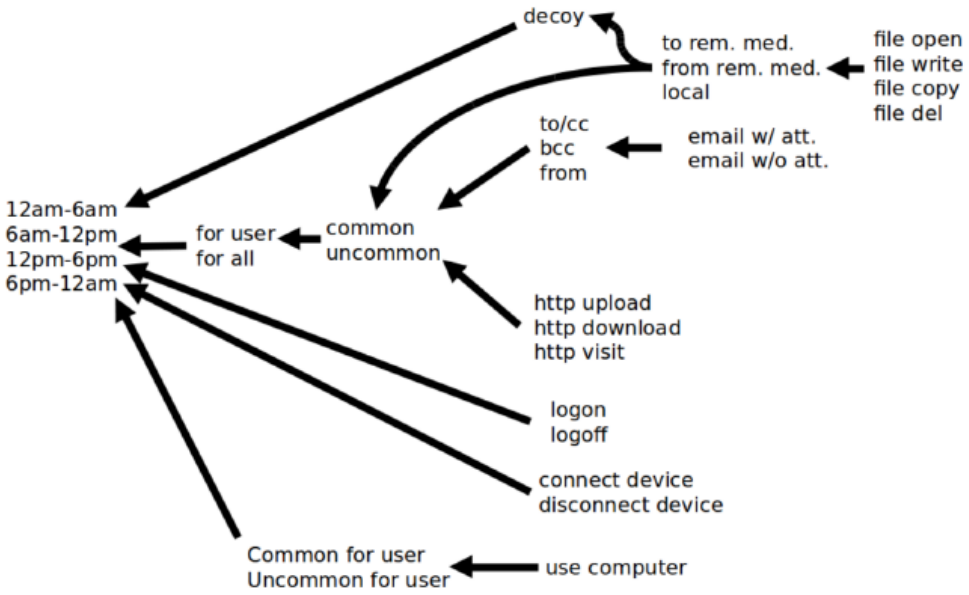


FIGURE 5.7: Hierarchy of features extracted to describe each user-day in the baseline system. For each path (bottom to top of the tree), the count of audit events matching the corresponding conditions is computed and added to the feature set. For example, the first feature (from the top) would be the number of file openings where a removable device is used, the opened file is a decoy and the action took place between 12am and 6am. Figure taken from [76].

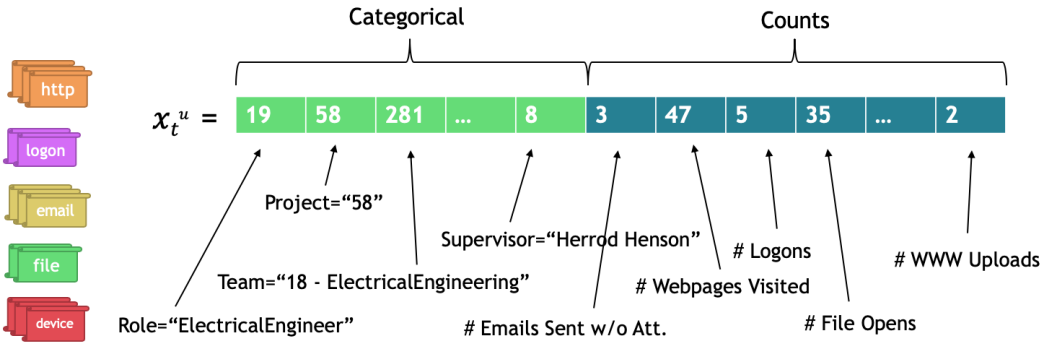


FIGURE 5.8: Example of a feature vector used as input in the baseline intrusion detection system. Figure taken from [168].

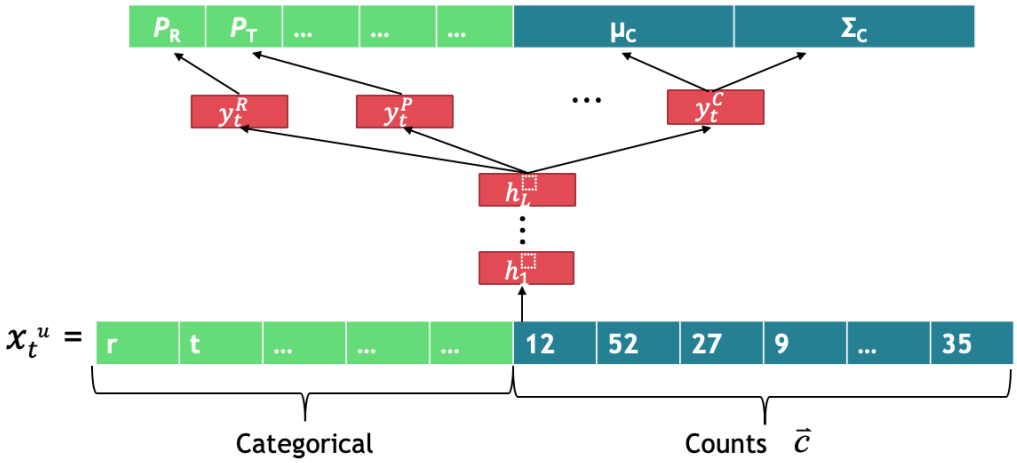


FIGURE 5.9: Predicting input probabilities in the baseline feed-forward neural network model. Figure taken from [168].

Finally, we follow the original method of Tuor *et al.* [76] to turn auto-encoder outputs into probabilistic anomaly scores. The anomaly score assigned to input x_t^u corresponds to the log-probability:

$$a_t^u = -\log P(x_t^u | h_{t-1}^u) \tag{5.2}$$

Assuming conditional independence for simplification, this probability is approximated by the joint probability of categorical and numeric (count) inputs:

$$P(x|h) \approx P(x_{counts}|h) \prod_{c \in C} P(x_c|h) \tag{5.3}$$

where x_{counts} and x_c respectively represent the continuous (blue in figure 5.9) and categorical (green in figure 5.9) components of feature vector x. At this point, an additional linear layer (parameters noted U, b and output y) is used to map predicted outputs to probabilities:

- for categorical components, the probability of observed value corresponds to a simple softmax: $P(x_c = i) = y_c^i = softmax(U^T h + b)^i$ where y_c^i denotes the component corresponding to value i;
- for the continuous count vector, the probability of observed value is computed via the multivariate normal density: $P(x_{counts}) = N(x_{counts}, \mu(y), \Sigma(y))$ where mean vector μ and covariance Σ (assumed to be diagonal) are obtained from the model.

Figure 5.9 summarizes this probability estimation step.

Zero-shot learning based intrusion detection system

Our zero-shot learning based intrusion detection system is an extension of the baseline system described above, as shown in figure 5.10. Feature extraction (step 2) and prediction of anomaly scores (step 3) are the same as for the baseline system.

However, an additional set of features is input to the model, namely graph embeddings used to characterize user context. These embedddings are obtained similarly to the process described in section 5.1. First, a graph containing users, their

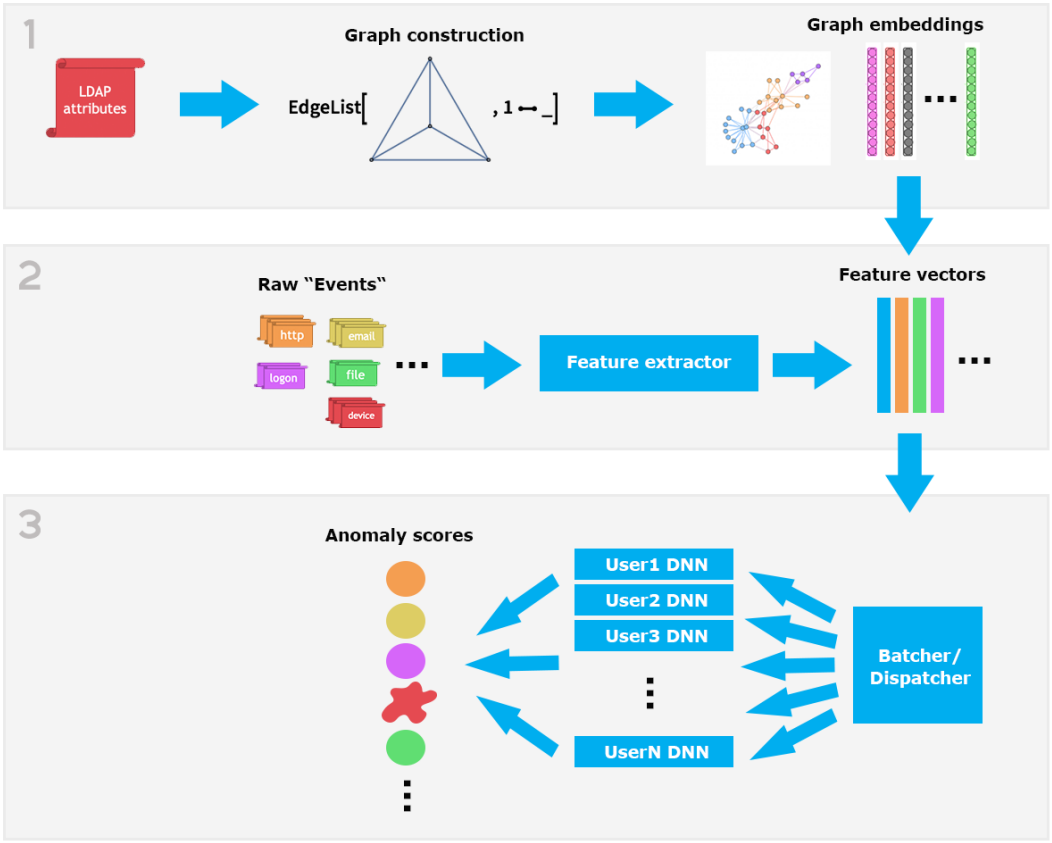


FIGURE 5.10: Overview of our intrusion detection system using zero-shot learning. We extend the system introduced in [76] (see figure 5.6) by concatenating activity features with graph embeddings constructed from the organizational graph. Figure taken from [168].

positions and relations in the organization's structure (as described in the LDAP repository provided in the CERT datasets) is built. In our preliminary analysis, we have found that assignments to projects and teams, as well as roles, allow to characterize user positions with adequate granularity. For this reason, we construct the graph of user relations as follows. Nodes are created for all user, team, project and role entities. In order to represent user roles within teams and projects, we also create nodes for existing combinations of roles and projects on the one hand, and roles and team on the other hand. For each user, edges are created to represent existing relations to following entities:

- project;
- team;
- role;
- role-project combination;
- supervisor;
- coworkers (users on the same team);
- roles of coworkers;
- role-project combinations of coworkers;
- role-team combinations of coworkers.

We then use some graph embedding technique to learn vector representations for entities present in this graph. The choice of the graph embedding method itself and of its configuration is regarded as hyper-parameters of our system.

In each of our two evaluation scenarios, following graph embeddings are used to characterize user context. For the first scenario (when a new user joins the organization), the embedding of the user node is used as additional input. For the second scenario (change of project assignment), we compare two approaches, named "role-project" and "role-project-team". In the role-project setting, we use the embedding representing the role-project entity (corresponding to the newly assigned project). However it is possible that such node does not exist; in this case the user embedding will be used as default. In the role-project-team setting, we use the embedding for the first existing entity in the following order: role-project, role-team, project. If none of these nodes exist, we use the user embedding as default.

Note that users can be idle between two project assignments. To address this issue we use a virtual "empty" project to which users are assigned when they have no real project ongoing.

5.2.3 Evaluation setting

In this section, we describe implementation details of our experiments for reproducibility. We first detail data selection and how our two evaluation scenarios (new users joining the organization and users project assignments) are implemented. Second we list hyper-parameters and how they were tuned in our study. A particular attention is given to the choice of graph embedding method used to extract vector representations for users, which we present separately. Finally, we describe the recall-based detection metric used for evaluation.

Scenarios

We compare our intrusion detection system to the baseline in two scenarios where no user activity data is available. The first of these scenarios represents new users joining the organization described in the CERT dataset. However, it turns out that the dataset contains only instances of existing users leaving and none of new users joining. Hence to implement our scenario, we select a random subset of users who will be added to the organization at a later time: at the end of the training period. For this we remove all their activity traces from the training set. The result is that joining users are present in the test set only. On their first work day, no previous activity history is available; however their LDAP attributes are known.

In the second scenario, we evaluate how our zero-shot learning approach improves detection performance when users are changing their project assignments. Unlike new users joining, project changes are already present in the CERT dataset hence no further modification of the original dataset is required in this case.

Data selection

We use version 6.2 of the CERT dataset, which contains approximately 135 million audit events and 5 threat cases. Following Tuor *et al.* [76], the first 418 days represent the training period while the last 98 days constitute the test set.

Hyper-parameters

Hyper-parameters of the feed-forward neural network used for prediction are set as follows. Following Tuor *et al.* [76], we use a batch size of 256 and set the learning rate to 0.01. We use hyperbolic tangent as activation function and the Adam optimizer [169] stochastic gradient descent. We tune the number of hidden layers (1 to 6), the hidden layer dimension (20 to 500) and the dimensionality of embeddings for categorical inputs (as ratio from the number of categories for each feature, between 0.25 and 1). In all cases we vary one parameter at a time and report results in the best performing configuration.

For the zero-shot learning system, another hyper-parameter is the graph embedding method used to obtain vector representations for users and other entities. We have experimented with following methods: node2vec [165], DeepWalk [166], LINE [170], SDNE [171] and struc2vec [172] with the parameters recommended by their authors. We have found that for our application best results were obtained with struc2vec.

Metrics

We use the recall curves and cumulative recall metrics introduced by Tuor *et al.* [76]. Cumulative recall values are normalized by their maximum attainable value in order to obtain a score between 0 and 1. These metrics are the same as the ones used in chapter 6 thus we refer the reader to their description in section 6.1.3. In the following results, we report cumulative recall values at budgets 1500 and 3500.

5.2.4 Results

Scenario 1: new users joining

Table 5.3 shows detection scores obtained for the baseline detection system and our zero-shot learning approach based on user embeddings in 3 configurations,

Model	CR-1500	CR-3500
Without user embeddings (baseline)	0.0	0.123
With user embeddings		
200 users joining	0.343	0.457
500 users joining	0.226	0.270
1000 users joining	0.136	0.263

TABLE 5.3: Normalized cumulative recall values at budgets 1500 and 3500 in the first evaluation scenario, for different numbers of new users joining the organization.

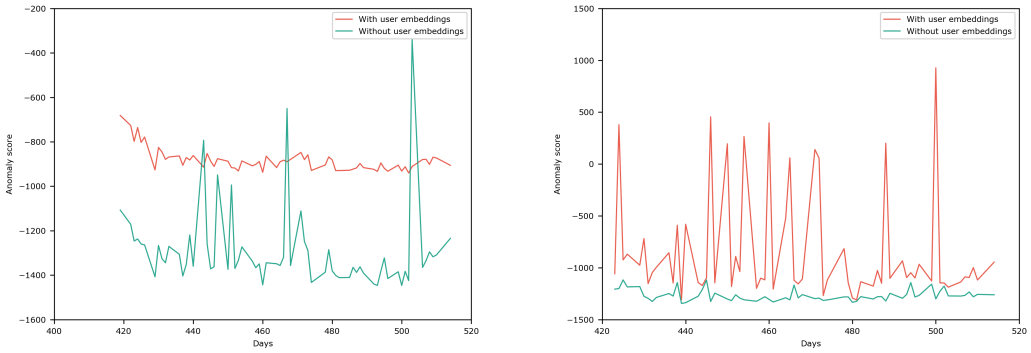
Model	CR-1500	CR-3500
Without user embeddings (baseline)	0.109	0.126
With user embeddings	0.471	0.630

TABLE 5.4: Normalized cumulative recall values at budgets 1500 and 3500 in the first evaluation scenario, for 200 new users joining, including all malicious insiders.

varying the number of new users considered. CR-1500 scores show that our approach outperforms the baseline in all configurations with scores in the range 0.136–0.343, while the baseline scores 0. Similarly, our detection system also outperforms the baseline on CR-3500 scores (0.236–0.457 against 0.123). At the two budgets considered, our approach integrating user embeddings has better intrusion detection performance. We also note that increasing the number of new users joining the CERT organization degrades detection performance. This result was expected because joining users are users whose activities are more difficult to predict than for other users. Hence by increasing the proportion of joining users among all users, the overall difficulty of predicting user activity increases as well.

In table 5.4, we compare the detection performance of our system and the baseline, in the case where all malicious insiders are among joining users (we set the number of joining users to 200). In this setting, integrating user embeddings is beneficial as well, leading to CR-1500 = 0.471 (baseline = 0.109) and CR-3500 = 0.630 (baseline = 0.126). Here, the detection performance boost surpasses the one obtained previously, where joining users were selected randomly. This can be explained by the fact that our approach specifically focuses on improving the activity prediction for some users (i.e. the ones who join the organization).

Finally, in figure 5.11, we show daily anomaly scores computed for two single users: one with only normal activities, and one malicious insider. Using recall at different budgets as described in section 6.1.3, anomaly scores are relative: at budget k, each day the k most anomalous users are scrutinized. Thus the absolute value of individual anomaly scores is not necessarily meaningful. However, the evolution of anomaly scores can be quite insightful. In figure 5.11a, we can see that our system yields more stable scores for a normal user, which should decrease false positives. On the contrary, malicious activities lead to higher fluctuations in the anomaly score (see figure 5.11b). Of course this only reflects the functioning of our approach on two specific users, however it provides an explanation to how our system is able to achieve better detection results than the baseline.



(A) Normal user (GSM0930) (B) Malicious user (CMP2946)

FIGURE 5.11: Representation of daily anomaly scores for two users (one without malicious behavior, left and one malicious insider, right), using the baseline detection system without embeddings (green curve) and our zero-shot learning system with user embeddings (red curve). Note that anomaly score values are meant to be interpreted relatively (absolute values are not necessarily meaningful). For the normal user, we expect the daily anomaly score value to fluctuate as little as possible, as high peaks could lead to false alarms. In this regard, our detection system with embeddings seems to compute much more stable scores than the baseline. For the malicious insider, we expect to observe the opposite: high peaks indicating anomalous activities, between days 400 and 460. Anomaly scores using our detection system match this expectation while the baseline system does not and might lead to false negatives in this case. Both figures taken from [168].

Scenario 2: project change

Table 5.5 shows detection scores obtained in the second scenario (project change) using the role-project and role-project-team approaches for user embeddings. With the role-project method, our system outperforms the baseline at both budgets 1500 (0.088 vs. 0.0) and 3500 (0.198 vs. 0.062). With the role-project-team approach, higher cumulative recall scores are attained and the baseline is outperformed by a larger margin (CR-1500 = 0.460 vs. 0.138 and CR-3500 = 0.658 vs. 0.140).

Figure 5.12 shows the evolution of the average anomaly score of two sets of users: one set contains users assigned to the same project; the other contains the same number of users but selected at random. When considering the set of random users, adding embeddings has little influence on anomaly scores, which are more or less constant in the represented time frame (keep in mind that score values are relative). However when considering the set of users which are simultaneously affected by project change (on day 486, vertical line), we see that without user embeddings (i.e. in the baseline setting), this change leads to higher anomaly scores. However project change is part of the normal life cycle of the organization and should not be considered malicious; hence these higher anomaly scores can possibly lead to false alarms. When user embeddings are integrated (i.e. in our system), the average anomaly score increase following project change is still observable, however the difference is less important.

Model	CR-1500	CR-3500
Role-project		
Without user embeddings (baseline)	0.0	0.062
With user embeddings	0.088	0.198
Role-project-team		
Without user embeddings (baseline)	0.138	0.140
With user embeddings	0.460	0.658

TABLE 5.5: Normalized cumulative recall values at budgets 1500 and 3500 in the second evaluation scenario (project change).

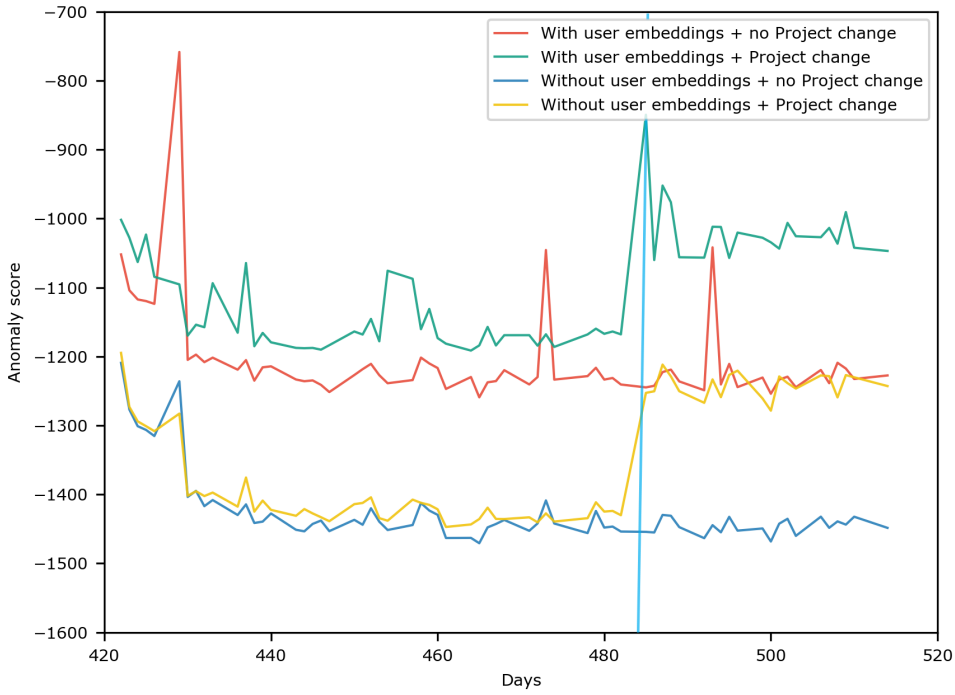


FIGURE 5.12: Daily average of anomaly scores for two sets of users in the second evaluation scenario (project change). Yellow and green curves respectively represent anomaly scores computed by the baseline (without embeddings) and our system (with embeddings) for a set of users assigned to the same project. The date of project change is indicated by the blue vertical line (day 486). Blue and red curves respectively represent anomaly scores obtained with the baseline and our system for a same size set of random users (i.e. not assigned to the same project). Figure taken from [168].

5.2.5 Discussion

Our experiments show that our proposed zero-shot learning approach allows to increase intrusion detection performance in two evaluation scenarios on the CERT insider threat dataset. By learning graph embeddings on user relations to organizational entities, like projects, teams and roles, we are able to extract vector representations which are meaningful to model user context.

In the first scenario, predicting user behavior – in the form of features summarizing user activity on the next day – is difficult for new users whose activity history is empty. However additional contextual information about users improves user behavior modeling, which translates into improved intrusion detection performance, through reduction of false positives and false negatives. Due to prediction being more difficult, detection performance of our system decreases as the number of new users increases; however the baseline system of Tuor *et al.* [76] is still outperformed for an unrealistically high proportion of new users (1000 out of 4000). Performance gain of our system compared to the baseline is even higher when malicious users are among the new users.

In the second scenario, which corresponds to users changing their project assignments, semantic embeddings also allow to improve intrusion detection performance. The choice of how to construct these embeddings of course has an influence on detection performance, but in the two settings tested, we find that our system outperforms the baseline. Moreover observing anomaly scores of individual users confirms our hypothesis that affectation to new projects is followed by a change of user behavior.

These findings are in line with intuitions and expectations which motivated the design of our system, as described in the introduction to this chapter. What is more surprising however is that our baseline system could not match performance levels reported in [76]; indeed our baseline scores far lower than expected. Note that our evaluation settings differ from the ones used in this previous study in several regards. First, this is due to reproducibility issues: Tuor *et al.* [76] do not provide exact descriptions for the extraction of user-day features, so we had to re-implement it (see section 5.2.2). Similarly, optimal hyper-parameter values are not stated. Second, unlike them we do include LDAP attributes as categorical values (in addition to user embeddings) and third we remove periods of user activity to simulate "new" users in the first evaluation scenario. Whether these differences can entirely explain the performance gap between our baseline and results reported by [76] should be further investigated.

5.3 Conclusion and perspectives

In this chapter, we have proposed to address the "cold start" problem of anomaly-based intrusion detection systems. These systems rely on past activity data to construct some profile of normal observations and flag deviations from this norm as anomalies. However, there are situations in which no relevant activity history is available: for example, for new users or when user change their behavior following new work conditions. We have addressed these two cases through the zero-shot learning paradigm, using semantic descriptions of users built from their relations to other entities within their work organization. A key point is that such relations are available independently of user activity, hence the "cold start" problem can be tackled.

Focusing on the CERT insider threat use case, we have shown how to obtain semantic representations of users and other organizational entities such as teams, projects and roles. The core of our approach consists in first constructing a graph representing users, entities and their relations within the organization, before applying graph embedding methods to extract vector representations of these graph nodes. With this process, we answer our research question RQ a.

Once such semantic representations of users (and entities) are obtained, they can be used as additional input feature for an intrusion detection system to address the "cold start" problem. Our experimental results show that this approach is able to outperform the baseline system where semantic descriptions are not included; this answers our research question RQ b.

Furthermore, our work shows that significant intrusion detection improvements can be attained, even with qualitative evaluation of different configurations for the construction of semantic representations (i.e. graph construction, embedding method and its parameters). This methodology has allowed us to demonstrate the feasibility of our zero-shot learning approach, while getting an intuitive comprehension of learned embeddings. Now that effectiveness of this approach is established, further improvement can be sought, in particular by integrating embedding learning and activity prediction into an end-to-end process.

The methodology employed here represents a specific embodiment of the zero-shot learning paradigm, designed to address the CERT insider threat case. However our approach is not applicable to this context only: contextual descriptions of users can be used to improve intrusion detection in other operational settings. With regard to research question RQ. 3, LDAP attributes representing user roles and their relations to organizational units constitute a particularly useful resource that can be leveraged via zero-shot learning. This calls for tighter coupling of intrusion detection with user identities, roles and privileges used in identity and access management solutions.

Chapter 6

Insider threat detection using heterogeneous data

In chapter 4, we have addressed a real-world intrusion detection case and found that user identification constitutes an effective approach to detect masquerades, but not insider threats. It also appears that currently collected audit data are not sufficient to characterize user behavior. As pointed out in section 2.3.2, heterogeneous features like graph and text data can provide informative context to detect insider threats. Thus we wish to extend our log data collection in this direction and ask following general research question:

RQ. 4: Which heterogeneous data features can be useful to detect intrusions, in particular insider threats, and how to leverage them?

We address RQ. 4 in the context of the CERT insider threat use case, which contains different sources of audit data and ground truth about intrusions. CERT datasets are described in more detail in section 3.2. Although generally helpful in insider threat detection, heterogeneous features have been widely ignored in the CERT case, as shown in section 2.3.3. Thus our first specific research question can be stated as follows:

RQ. a: How to leverage heterogeneous data in the CERT datasets?

Our main contribution consists in introducing ADSAGE (Anomaly Detection in Sequences of Graph Edges), a new method for anomaly-based intrusion detection supporting heterogeneous data. In particular, we leverage graph features by modeling user events (equivalent to log lines) as graph edges representing interactions between entities. For instance, an email being sent corresponds to an edge from the sender to the receiver. Edges can be augmented with attributes to provide context, such as the time the email was sent or its text content. Note that ADSAGE's applicability is not limited to insider threat detection; our method can be used for anomaly detection in sequences of attributed graph edges in general. To the best of our knowledge, no existing method for anomaly detection at edge level supports both edge sequences and attributed edges (see related work in section 2.3.4).

Besides better support of heterogeneous data, we address the issue of alert traceability. Existing insider threat detection systems heavily rely on data aggregation and feature engineering [50], [52], [76]. Indeed, this strategy can be effective, nevertheless at the cost of alert traceability. For instance in [76], users are assigned an anomaly score based on their daily activity, thus determining specifically which action(s) lead to an alert is not straightforward.

On the contrary, our method operates at event (i.e. log line) level with a unique data source. This allows flagging anomalies at a fine-grained level and reduces the need for feature engineering and data aggregation. However, it is important to note that direct performance comparison with existing systems like [76], which leverage multiple audit data sources, is unfair. In this study, we rather seek answers to following specific research questions:

RQ. b: Is detection at fine-grained level feasible with individual sources of audit data?
RQ. c: Which sources of audit data can be useful for insider threat detection and for which threat scenarios?

Finally, as pointed out in section 2.2.3, the CERT insider threat use case currently lacks a standard benchmark methodology. Existing works use different metrics and data subsets for evaluation, rendering performance comparison difficult. As an effort towards benchmark standardization we adopt the evaluation setting from [76], chosen for its business-realistic metrics.

The rest of this chapter is structured as follows. We first introduce our method ADSAGE and evaluate it on the CERT insider threat datasets in section 6.1. In section 6.2, we analyze properties of the graphs representing the different audit data sources in the CERT datasets, providing additional insight with regard to RQ. c. Section 6.3 concludes this chapter by summarizing findings, answering our research questions and suggesting possible extensions.

6.1 ADSAGE: Anomaly Detection in Sequence of Attributed Graph Edges

6.1.1 Problem setting: anomaly detection at event level

We address the CERT insider threat use case through an anomaly detection perspective, i.e. we aim at modeling normal user behavior to detect deviations from this norm. Such anomalies are then considered as insider threat alarms. While this perspective has been widely adopted for intrusion detection, unlike existing systems our approach is to perform detection at fine-grained event level. In the CERT insider threat use case, one event (i.e. log line) represents an elementary user action and usually contains features to describe its context. For instance, an event can represent a logon to a particular computer and features can be the event time or the device used. Our goal is to assign an anomaly score to each audit event.

The primary reason to perform detection at fine-grained event level is to enhance alert traceability. Intrusion detection systems are typically not used as standalone solution, but rather perform a first selection of suspicious activities to be further scrutinized by security analysts. In this context, flagging anomalies at fine-grained event level eases traceability, as analysts will be able to determine exactly which user action lead to an alert. On the contrary, using a system like [76], an anomaly score is assigned to a whole day of user activity, thus when an alarm is raised the question of which exact elements triggered it remains open. A second advantage is that data aggregation and feature engineering efforts are greatly reduced compared to systems like [50], [52], [76]. As we will show next, except for time features (which we transform only to model their periodical nature), our methods use audit event attributes without further preprocessing.

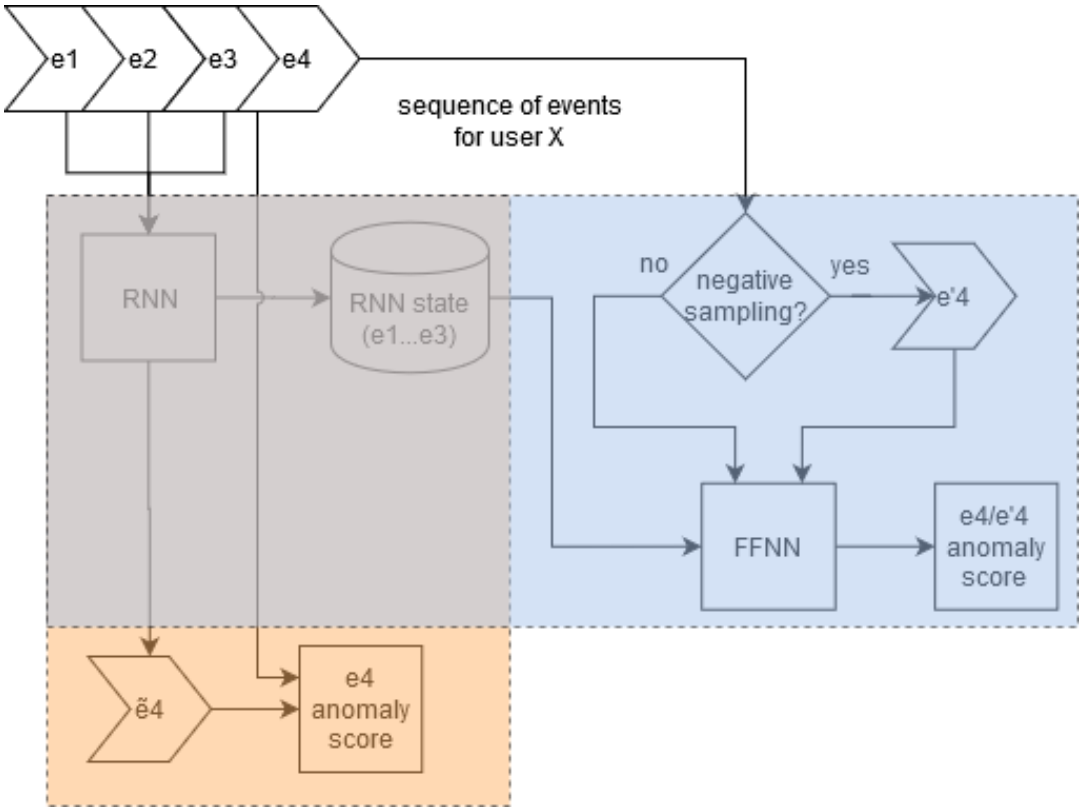


FIGURE 6.1: Joint training architecture for the recurrent (RNN) and feed-forward neural network (FFNN) used in ADSAGE. This example shows the anomaly score prediction for event $e4$ given previous events $e1...e3$. Note that ADSAGE components are delimited by the blue area, while the seq2one baseline corresponds to the orange area. The grey area corresponds to components shared by both models. Seq2one is trained on normal events only, predicts the entire next event $\tilde{e4}$ and compares it to the real $e4$ to derive an anomaly score. ADSAGE is trained on both normal and anomalous events being generated with negative sampling (e.g. $e'4$).

6.1.2 Methods

Seq2one baseline

To detect insider threats at event level, we adapt DeepLog [107], a log line anomaly detector for system traces. As we take into account one audit data source at a time, we only keep DeepLog's event features prediction module. It computes an anomaly score based on the error between predicted and observed value for the next event. We adapt the error function to support numeric and categorical attributes. For numeric features, mean squared error is used and for categorical attributes the error is $1 - p$, where p is the probability of the true category, obtained by applying the softmax function. Each error is then normalized by using its quantile (e.g. 0.99 if the error is greater than 99% of observed errors for this feature). Quantiles are finally averaged to obtain an event anomaly score. This method is referred to as "seq2one" and corresponds to the orange area in figure 6.1.

Unfortunately, our preliminary experiments on the CERT datasets have shown that seq2one gives poor threat recall. One plausible explanation is that user behavior in far less predictable than machine behavior, hence predicting the next event is

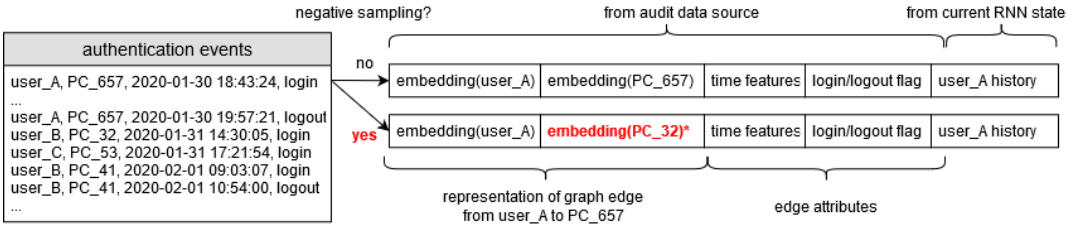


FIGURE 6.2: Representation of audit events as attributed graph edges in ADSAGE on the example of authentication events. An authentication event corresponds to a user-computer edge and is represented as the concatenation of user and computer embeddings. Edge attributes can be added as well (i.e. time features, flag indicating logon or logoff). The user history (RNN state) representing previous activities of current user A is also appended. Here, the first authentication event is encoded to be processed by ADSAGE's FFNN. On the right, the upper row represents the normal event. The lower row represents an anomalous version of the same event, obtained with negative sampling on the edge destination (indicated by the asterisk). The accessed computer in the anomalous event (PC_32) is selected at random among all computers never used by user A.

much more difficult with user activity traces than with system logs used in [107]. This motivates us to extend DeepLog to better learn the distinction between normal and anomalous behavior.

ADSAGE: Anomaly Detection in Sequence of Attributed Graph Edges

As predicting the exact features of the next event is difficult for user generated events, we propose ADSAGE, a method focusing on predicting the validity of graph edges. In the following, we detail how events can be represented as attributed graph edges and how ADSAGE is trained to predict the validity of such events, by relying on negative sampling.

Representing events as attributed graph edges In ADSAGE, events are represented as attributed graph edges. Figure 6.2 shows an example on authentication events similar to logs from CERT. An authentication event is an interaction between a user and a computer, corresponding to an edge in the graph of users and computers. In ADSAGE, this edge is represented as the concatenation of its source (user) and destination (computer) entities. As ADSAGE is based on neural network models, an embedding layer is used for each entity feature. Thus source and destination embeddings are optimized according to the prediction task.

In addition to its source and destination entity, an edge can also have features extracted from its event context (e.g. time features and logon/logoff attributes in figure 6.2). Different types of features are possible: numeric values, categorical attributes (as one-hot or embedding representation) or even text content (via pre-trained word embeddings). Therefore heterogeneous data sources are supported.

Note that ADSAGE can be easily extended to the case where an event has multiple sources and/or destinations. Events with a fixed and limited number of sources or destinations can be represented as concatenation of corresponding embeddings. For events with a high and/or varying number of sources or destinations, it is possible to use an embedding bag layer [173] (i.e. an embedding layer with pooling function such as average or max) to obtain a fixed-length representation.

Training FFNN and RNN jointly to learn edge validity To perform anomaly detection in sequences of attributed edges, we use a combination of sequence-to-one RNN (similar to seq2one) and feedforward neural network (FFNN), both trained jointly. Given a sequence of events for a given user, the RNN is trained to predict the next event and outputs an RNN state representing the event history up to this instant. The RNN's training loss is computed as the sum of mean squared error (for numeric features), cross-entropy loss (for one-hot encoded features) and cosine loss (for embeddings).

The RNN state encoding history of previous events is used as input for the FFNN, together with the next event. The FFNN is trained to predict a binary label (representing whether the next event is valid or not) using binary cross-entropy loss.

Figure 6.1 shows the full architecture with RNN and FFNN, and algorithm 1 details how both are trained simultaneously. Both our seq2one baseline and ADSAGE maintain a separate RNN state for each user. With this mechanism each user event sequence can be modeled individually while the model is trained on all users.

Note that ADSAGE is trained on both normal and anomalous events (generated by negative sampling, see next section) and outputs anomaly scores directly while seq2one is trained on observed events only and predicts entire events. As shown later in evaluation, these differences allow ADSAGE to better detect anomalous events.

Algorithm 1 Training ADSAGE

```
function train_ADSAGE (n_epochs, batches):
for ep in n_epochs do
  reset_rnn_states() // set all user RNN states to zero
  for ffnn_x, ffnn_y, rnn_x, rnn_y in batches do
    // ffnn_x: next events including negative samples
    // ffnn_y: event validity labels
    // rnn_x: sequences of true (positive) events
    // rnn_y_true: next true event for each sequence in rnn_x
    // FFNN step
    // retrieve current RNN states for FFNN inputs
    ffnn_states = get_user_rnn_states(ffnn_x)
    // concat input events and user histories
    ffnn_input = {ffnn_x, ffnn_states}
    // predict validity of events (anomaly scores)
    ffnn_y_pred = FFNN(ffnn_input)
    backprop(ffnn_y_pred, ffnn_y)
    // RNN step
    // retrieve current RNN states for RNN inputs
    rnn_states = get_user_rnn_states(rnn_x)
    // predict the next event for each sequence
    rnn_y_pred, rnn_states = RNN(rnn_x, rnn_states)
    backprop(rnn_y_pred, rnn_y)
    save_user_rnn_states(rnn_states)
  end for
end for
```

Generating anomalous edges through negative sampling In order to get negative examples for the event validity classification task (i.e. anomalous edges), we artificially replace the destination entity through negative sampling (see figure 6.2). In a negative event, the destination entity should be anomalous in the sense that interactions from the source entity are usually not observed. In practice we randomly draw a destination entity (e.g. computer for logons) from the set of destinations never accessed from the source entity (e.g. user) during the training period, while other edge attributes are left unchanged. We use a constant negative sampling rate of 0.5 (i.e. for one positive event, we generate a corresponding negative event), however this value could be tuned as desired.

6.1.3 Evaluation setting

Baselines

In each evaluation setting, we benchmark ADSAGE against 3 different types of baselines. The first is "seq2one" which uses an RNN model to predict the features of next event given previous events (see section 6.1.2).

Simple rule-based classifiers constitute the second type of baselines. These models are not expected to be competitive for insider threat detection in practice, but they should be outperformed by ADSAGE to ensure that detected anomalies are not trivial. For example, if each user is assigned a computer, a simple rule is to consider all authentication attempts to a different computer as anomalous. Similar rules can be used for other types of events, the general pattern being that an edge from a source to a destination entity is flagged as anomalous if it was not observed in the train set, and as normal otherwise.

The last baseline we use is SedanSpot [129], a general anomaly detection method applicable to sequences of graph edges. SedanSpot takes into account the timestamp of each edge, but does not support additional edge attributes. Note that SedanSpot and rule-based methods are deterministic and do not depend on initialization. This is why we report exact performance metrics for these methods, unlike for ADSAGE and seq2one.

Feature selection

ADSAGE and seq2one allow a flexible selection of features. However as our goal is to reduce feature engineering and preprocessing, we consistently use following approach for all events from CERT. First, we extract two time features from the date/time of each event: the minute of day and day of week. We represent both through their cosine and sine values in order to model their periodical nature. Second, we use all other (i.e. non time) available event attributes without further modification (except for categorical values, for which we use one-hot encoding). In each experiment, we list these additional features for completeness.

Tuning hyper-parameters

For each dataset we optimize ADSAGE's hyperparameters. Most of them are related to the underlying neural networks (RNN and FFNN). We tune following hyperparameters: number of timesteps, number of hidden units and layers in the RNN, batch size, dimension of embeddings used to represent graph features, learning rate and use different types of pre-trained word embeddings (for datasets containing text

features). To speed up training on large datasets, we reduce training set size by sampling users randomly. The user sample rate is another hyperparameter to tune, and one can also choose to sample only from users presenting no malicious behavior. Testing is always performed on all users. We tune one hyperparameter at a time to determine its optimal value, then combine all best parameter values as final configuration. Although this process does not take into account dependencies between hyperparameters, it is much faster than extensive grid search. For each evaluation setting we report the optimal configuration found.

Recall-based metrics

For our evaluation, we use recall-based metrics introduced in [76]: recall curves and cumulative recall at budget k (CR_k). These metrics are realistic from the perspective of an organization with a fixed budget to investigate alerts generated by an insider threat detection system. The organization's daily budget k represents the number of (most suspicious) users to be investigated each day. If a malicious user is investigated on a given day, all his malicious activities conducted that day are considered as detected. Recall (at budget k) R_k is computed as the recall of malicious users per day, averaged over all test days (days with no malicious activity are ignored).

R_k reflects detection performance at a fixed daily investigation budget. To assess performance across multiple budgets, R_k can be plotted against k up to a maximum budget k_{max} to obtain a recall curve. Such curve can be summarized with normalized cumulative recall computed as $CR_k = \sum_{i=0}^{i=k} R_i \ / n$, where n is the number of budget steps. CR_k can be seen as an approximation of the area under recall curve up to budget k.

We report cumulative recall (CR) at budgets 400 and 1000 and at maximum budget 4000 for completeness. We also report recall metrics based on detecting *all* threats present in the test set, i.e. malicious activity across *all event types*, including log data sources not seen by the detector. This is possible with daily budget-based recall metrics, by assuming that investigators review all user activity that day, even if the alert was generated by a single type of event. For example, an anomaly alert triggered by an unusual logon event might lead to an investigation which will uncover malicious email activity from the same user on the same day. This setting leads to metrics aligned with [76]. However keep in mind that the performance comparison is unfair since ADSAGE and other baselines see a single log data source.

Moreover, note that although ADSAGE detects threats at *event* level, recall at budget is computed at *user-day* level, i.e. in terms of number of anomalous users detected for a given day. The first reason to do so is to allow a comparison with [76]. The second is that when a user is investigated following an alert, the investigator will have to review the entire user activity (at least the whole user-day) to have sufficient context to come to a decision.

Data selection, audit log sources and threat scenarios

Using the CERT dataset version 6.2, we perform the same data split as [76] to compare our results to theirs (days 1 to 418 for training, days 419 to 516 for evaluation).

We also detail results for individual threat scenarios. This helps understanding which types of malicious behaviors are well detected by each method, and whether "blind spots" remain. Certain scenarios are virtually impossible to detect using some

log data sources. For example, scenario 2 does not involve any logon activity, meaning that logon event detectors only cannot possibly alert about threats of this type.

For these reasons, methods presented in the following experiments should not be viewed as standalone, "one-fits-all" detectors. They rather are complementary, and each one addresses the CERT insider threat detection problem from a different perspective, depending on its data source. Though we compare our results to those of [76] (who performs detection at user-day level and uses all data sources), our focus is on understanding which event types are relevant (in general and for each scenario) and finding out whether insider threat detection is feasible at fine-grained event level.

To complement our results on the synthetic CERT datasets, we evaluate our methods on real-world authentication logs from the LANL cyber-security datasets.

6.1.4 Results

Detecting threats in CERT logon events

In a first experiment, we apply ADSAGE and other baselines to detect insider threats in logon events from the CERT dataset. In addition to edge sources and destinations and time features, we include the binary attribute indicating whether the action performed was a login or a logoff.

We use two simple rule-based baseline detectors. "Own PC" flags all logon events occurring on user's own machine (defined as the most used computer for this user) as normal; all other events are considered anomalous. "Known PC" considers a logon event to be normal if the corresponding user-computer edge was observed in the training set; otherwise it will be flagged as anomalous. Both methods provide binary decisions.

We use following hyperparameters for seq2one and ADSAGE's RNN: 1 layer of 30 LSTM units, 15 timesteps, batch size = 100, learning rate = 0.001 with decay factor of 0.5 after 1 epoch without improvement and the dimensionality of computer embeddings is set to 20. For ADSAGE's FFNN we use 3 layers of respectively 50, 30 and 10 units with relu activation and dropout set to 0.2. We perform 5 runs with 10 epochs.

Detection results are shown in table 6.1. When it comes to detecting threats present in logon events only, ADSAGE outperforms all other methods, with cumulative recall at maximum budget of 0.981. Cumulative recalls at lower budgets show a similar picture, and full recall curves presented in figure 6.3 confirm that ADSAGE performs best at almost any budget. However, for the task of detecting all threats (i.e. including the ones not present in logon activity), ADSAGE is outperformed by the system of [76].

Detecting threats in CERT email events

In a second experiment, we use email events as log data source. Email events from the CERT dataset represent an email being sent or received/read. Considering the significant overlap between the two, we only use "send" events.

In addition to time features, we use following attributes from email events: email size (numeric), sender and receiver fields represented as embeddings ("from", "to", "cc", "bcc") and email content (text). Representing the sender is straightforward as it contains only one email address, so we use a simple embedding layer. However, receiver fields can contain several entities, so we combine them with an embedding bag layer [173] to obtain a fixed length representation. All three receiver fields are

Logon threats	CR-400	CR-1000	CR-4000
own pc	0.633	0.853	0.963
known pc	0.617	0.847	0.962
SedanSpot	0.219	0.513	0.874
seq2one	0.039 ± 0.061	0.171 ± 0.151	0.679 ± 0.084
ADSAGE	**0.813 ± 0.172**	**0.925 ± 0.069**	**0.981 ± 0.017**
All threats	CR-400	CR-1000	CR-4000
own pc	0.268	0.420	**0.772**
known pc	0.280	0.426	**0.772**
SedanSpot	0.119	0.338	**0.814**
seq2one	0.047 ± 0.088	0.155 ± 0.073	0.679 ± 0.084
[76]	**0.731**	**0.893**	not reported
ADSAGE	0.432 ± 0.037	0.605 ± 0.102	**0.842 ± 0.104**

TABLE 6.1: Detection results on logon events. For seq2one and ADSAGE we report 95% confidence intervals over 10 runs. Top table: detecting threats present in logon events only, bottom table: detecting all threats (including those not present in logon events).

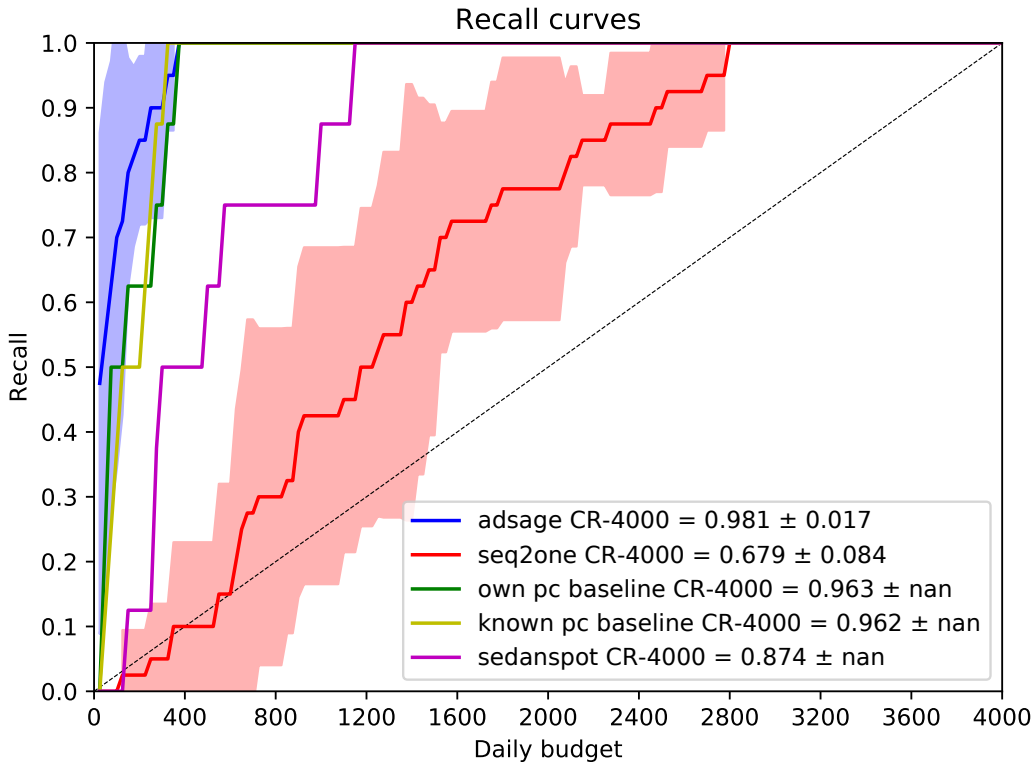


FIGURE 6.3: Recall curves with 95% confidence intervals over 5 runs for detecting threats present in logon events.

Email threats	CR-400	CR-1000	CR-4000
known receivers	**0.106**	0.340	0.782
known receiver set	**0.138**	0.278	0.725
SedanSpot	0.044	0.098	0.628
seq2one	**0.217** $\pm$ **0.124**	0.431 $\pm$ 0.109	0.830 $\pm$ 0.035
ADSAGE	**0.332** $\pm$ **0.226**	**0.646** $\pm$ **0.117**	**0.907** $\pm$ **0.026**
All threats	CR-400	CR-1000	CR-4000
known receivers	0.116	0.408	0.827
known receiver set	0.134	0.318	0.754
SedanSpot	0.280	0.415	0.784
seq2one	0.199 $\pm$ 0.093	0.426 $\pm$ 0.100	0.822 $\pm$ 0.036
[76]	**0.731**	**0.893**	not reported
ADSAGE	0.447 $\pm$ 0.118	0.728 $\pm$ 0.67	**0.930** $\pm$ **0.017**

TABLE 6.2: Detection results on email events. For seq2one and ADSAGE we report 95% confidence intervals over 5 runs. Top table: scores when detecting threats present in email events only, bottom table: scores when detecting all threats (including those not present in email events).

encoded as separate features; senders and receivers are embedded into a unique vector space. Text content of emails is represented through pre-trained word vectors, combined with a pooling scheme [115]. We have empirically determined that GloVe [113] vectors with average pooling work best for our problem.

We use two rule-based baselines for anomaly detection in email events. In the first, called "known receivers", each email event is assigned a score representing the proportion of unobserved receivers, i.e. receivers that were never contacted by the sender during the training period. The second is referred to as "known receiver set". It assigns a binary score depending on whether the exact set of receivers was observed in the training set for the corresponding sender (normal) or not (anomalous).

We use following hyperparameters for seq2one and ADSAGE's RNN: 1 layer of 100 LSTM units, 20 timesteps, batch size 1024, 5 epochs, learning rate of 0.01 with 0.5 decay factor after 1 epoch without improvement. Embeddings of email senders and receivers are of dimension 20. For ADSAGE's FFNN we use 3 layers of respectively 50, 30 and 10 units with relu activation and dropout = 0.2. We perform 5 runs with 5 epochs and use the same data split as for logon events, but we train only on a random sample of all users (10%). Due to the large number of email events, this speeds up the training process without significantly altering performance.

Detection results are shown in table 6.2. For threats present in email events, ADSAGE outperforms other methods and reaches a cumulative recall at maximum budget CR-4000 = 0.907. As shown in figure 6.4, a budget of around 800 allows to detect 90% of threats in email events. Applying ADSAGE to email events also allows to detect threats present in all events effectively (CR-4000 = 0.930), even though the system of [76] still performs best. Nevertheless it suggests that email events are a good marker for insider threats.

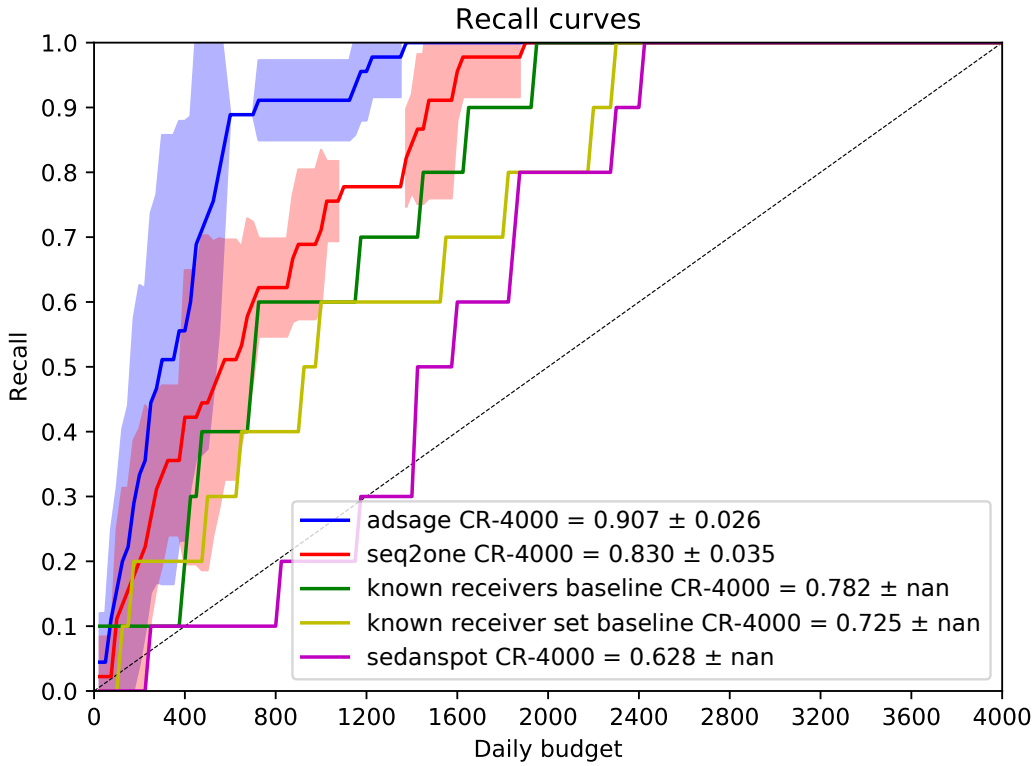


FIGURE 6.4: Recall curves with 95% confidence intervals over 5 runs for detecting threats present in email events.

Detecting threats in CERT web events

In a third experiment on the CERT dataset, we use web events as data source. Web events represent user browsing activities. In addition to edge sources and destinations and time features, we tried adding the content of web page as text feature but ended up discarding it as it did not improve detection performance significantly.

We use a rule-based baseline for anomaly detection which we call "known domain". It assigns binary anomaly scores based on whether the web domain of an event has been observed in the training period for the corresponding user. Thus all accesses to new, unobserved domains are considered anomalous; the rest is deemed normal.

We use following hyperparameters for seq2one and ADSAGE's RNN: 1 layer of 100 LSTM units, 20 timesteps, batch size 2048, 5 epochs and learning rate of 0.001 with 0.5 decay factor after 1 epoch without improvement. Embeddings of email senders and receivers are of dimension 50. For ADSAGE's FFNN we use 3 layers of respectively 50, 30 and 10 units with relu activation and dropout = 0.2. As the volume of web events is much larger than for other audit data sources, we train on a random sample of 5% of all users, discarding malicious ones and perform 5 runs with 5 epochs.

Table 6.3 shows detection results. SedanSpot outperforms other methods with CR-4000 = 0.928 when detecting threats present in web events only. As shown in figure 6.5, a budget of around 600 allows to detect all threats in web events. When considering recall of all threats, SedanSpot gives the best results (followed by seq2one, difference not statistically significant), but is not as effective as the system from [76], which uses all data sources. Overall, ADSAGE is not adapted to detect anomalies

Web threats	CR-400	CR-1000	CR-4000
known domain	0	0.150	0.598
SedanSpot	**0.313**	**0.711**	**0.928**
seq2one	0.175 ± 0.129	0.344 ± 0.054	0.745 ± 0.078
ADSAGE	0.054 ± 0.070	0.179 ± 0.127	0.696 ± 0.102
All threats	CR-400	CR-1000	CR-4000
known domain	0.042	0.148	0.588
SedanSpot	0.199	0.432	**0.736**
seq2one	0.132 ± 0.078	0.259 ± 0.087	$\mathbf{0.693 \pm 0.061}$
[76]	**0.731**	**0.893**	not reported
ADSAGE	0.035 ± 0.030	0.109 ± 0.026	0.608 ± 0.031

TABLE 6.3: Detection results on web events. For seq2one and ADSAGE we report 95% confidence intervals over 5 runs. Top table: scores when detecting threats present in web events only, bottom table: scores when detecting all threats (including those not present in web events).

in web events represented as user to web domain edges. One possible explanation is that the domain identifier is not informative enough to characterize browsing behavior.

Detecting different threat scenarios

In order to characterize which methods and data sources allow to detect each CERT insider threat scenario (see section 3.2.2), we evaluate logon, email and web detectors using a different data split. We use the period from January to July 2010 as train set and test on August 2010 to April 2011. This allows us to assess detection performance on all threat scenarios, whereas the test set used by [76] contains only scenarios 2 and 4. Hyperparameter values determined earlier are kept unchanged.

For completeness, we also add detectors for file and device events, which represent the two remaining audit data sources in the CERT datasets. However, representing these events as user relations to entities is not as natural as for logon, email and web events.

File events are represented as user to text content relations, i.e. for one log line, its destination node in ADSAGE is the embedding representing the content of the accessed file. Text embeddings are obtained through pre-trained word vectors combined with pooling [115] and kept fixed during training. ADSAGE is trained with negative sampling, where for each user, negative examples are randomly drawn from the set of never accessed files.

We also report results obtained with seq2one and a rule-based baseline called "known file activity". In this last model, a given event is flagged as normal if the current combination of activity type (file open or write) and file type (extension) has been observed in the train set for current user; otherwise the event is considered anomalous. SedanSpot is not applicable here as destination nodes consist of text data.

Device events represent the usage of removable devices by users. Within ADSAGE, a device log line is regarded as link between a user and the file structure present on the device. This file structure corresponds to the "file tree" attribute of the CERT dataset, i.e. the list of directories present on the device. We encode this

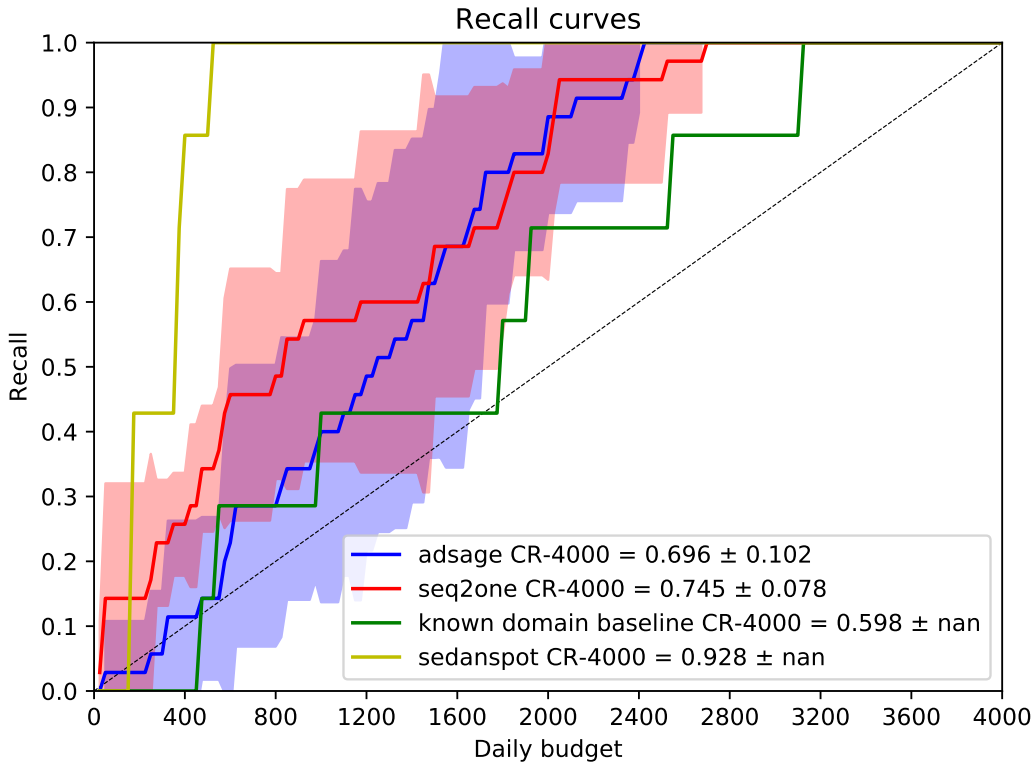


FIGURE 6.5: Recall curves with 95% confidence intervals over 5 runs for detecting threats present in web events.

destination node attribute as set of categorical values using an embedding bag [173]: each directory is encoded as separate embedding, and all embeddings are combined using pooling [115]. As for the other event types, ADSAGE is trained with negative sampling, for which negative examples consist of unobserved links between source (i.e. users) and destination nodes (i.e. file structures).

In addition to ADSAGE, we report results for the seq2one baseline and a binary rule-based classifier "known device activity". This last baseline flags events as abnormal if their combination of activity (connect/disconnect), pc and file tree was not observed in the train set for the corresponding user; and as normal otherwise.

Detection results (CR-4000 scores) for each scenario presented by detector and data source are shown in tables 6.4 (scenarios 1 to 3) and 6.4 (scenarios 4, 5 and all threats present in each audit data source).

It appears that monitoring logon events can be effective (CR-4000 ≥ 0.85) to detect scenarios 3, 4 and 5. Rule-based methods ("own PC", "known PC") give good performance for these scenarios, however ADSAGE and SedanSpot can be better for 3 and 5 respectively. Email traffic is a good audit data source to detect scenarios 2, 4 and 5. ADSAGE ranks among the best detectors for scenarios 2 to 5, while SedanSpot is particularly effective for scenario 2 and the "known receivers" baseline proves strong against scenarios 4 and 5. Web browsing logs can be used to uncover scenarios 2 (using SedanSpot or "known domain"), 4 and 5 (with "known domain"). File logs allow to detect all five scenarios using ADSAGE or the "known file activity" rule-based method. Finally, device detectors are effective to flag scenarios 1 to 3 (scenarios 4 and 5 do not imply usage of removable devices).

	Threat scenario		
	1	2	3
Logon			
own pc baseline	0.392	0.636	0.848
known pc baseline	**0.598**	0.646	**0.855**
SedanSpot	0.117	0.642	0.364
seq2one	**0.618 ± 0.099**	0.614 ± 0.019	0.695 ± 0.017
ADSAGE	**0.645 ± 0.109**	0.667 ± 0.086	0.825 ± 0.012
Email			
known receivers baseline	0.652	0.810	0.600
known receiver set baseline	0.617	0.714	0.646
SedanSpot	**0.823**	**0.928**	**0.664**
seq2one	**0.762 ± 0.124**	0.770 ± 0.036	**0.669 ± 0.054**
ADSAGE	0.668 ± 0.120	**0.853 ± 0.132**	**0.669 ± 0.043**
Web			
known domain baseline	**0.731**	0.853	**0.717**
SedanSpot	0.294	**0.944**	0.554
seq2one	**0.743 ± 0.089**	0.720 ± 0.021	0.619 ± 0.029
ADSAGE	0.468 ± 0.075	0.704 ± 0.112	**0.727 ± 0.029**
File			
known file activity baseline	**0.979**	**0.937**	0.905
seq2one	0.870 ± 0.050	0.917 ± 0.007	**0.915 ± 0.014**
ADSAGE	0.930 ± 0.012	0.900 ± 0.015	**0.931 ± 0.017**
Device			
known device activity baseline	**1.000**	**0.963**	**1.000**
seq2one	0.909 ± 0.059	**0.958 ± 0.006**	0.935 ± 0.022
ADSAGE	0.980 ± 0.010	**0.973 ± 0.022**	**0.968 ± 0.032**

TABLE 6.4: Detection performance (cumulative recall at maximum budget, CR-4000) for insider threat scenarios 1 to 3.

	Threat scenario		
	4	5	all
Logon			
own pc baseline	0.966	0.963	0.845
known pc baseline	0.963	0.963	**0.884**
SedanSpot	0.875	**0.969**	0.633
seq2one	0.690 ± 0.046	0.515 ± 0.185	0.687 ± 0.038
ADSAGE	**0.975 ± 0.007**	0.495 ± 0.135	**0.876 ± 0.033**
Email			
known receivers baseline	**0.885**	**0.900**	**0.702**
known receiver set baseline	0.783	0.894	0.695
SedanSpot	0.548	0.763	0.682
seq2one	**0.815 ± 0.032**	0.730 ± 0.210	**0.712 ± 0.032**
ADSAGE	**0.798 ± 0.138**	**0.942 ± 0.071**	**0.722 ± 0.039**
Web			
known domain baseline	**0.837**	**1.000**	**0.930**
SedanSpot	0.433	0.906	0.880
seq2one	0.638 ± 0.039	0.680 ± 0.257	0.721 ± 0.044
ADSAGE	0.446 ± 0.056	0.631 ± 0.287	0.690 ± 0.099
File			
known file activity baseline	**0.906**	**0.956**	**0.930**
seq2one	**0.908 ± 0.021**	0.879 ± 0.050	0.903 ± 0.011
ADSAGE	**0.911 ± 0.013**	**0.899 ± 0.071**	**0.924 ± 0.008**
Device			
known device activity baseline	-	-	**0.971**
seq2one	-	-	0.951 ± 0.014
ADSAGE	-	-	**0.980 ± 0.013**

TABLE 6.5: Detection performance (cumulative recall at maximum budget, CR-4000) for insider threat scenarios 4, 5 and all threats present in each respective audit data source.

Detecting red team events in LANL authentications	CR-1000	CR-4000	CR-12000
known dest pc	0.237	0.566	0.829
known source pc	**0.254**	**0.669**	**0.890**
SedanSpot (dest) pc	0.016	0.104	0.490
SedanSpot (source) pc	0.089	0.191	0.538
seq2one	0.167 ± 0.023	0.423 ± 0.031	0.752 ± 0.014
ADSAGE	$\mathbf{0.255 \pm 0.078}$	$\mathbf{0.653 \pm 0.042}$	$\mathbf{0.877 \pm 0.014}$

TABLE 6.6: Detection results for red team anomalies in LANL authentication events. We report normalized cumulative recalls at budgets 1000, 4000 and 12000 for 5.2. For seq2one and ADSAGE we report 95% confidence intervals over 5 runs.

Detecting anomalies in real authentications

To complement our results on the synthetic CERT datasets, we evaluate our methods on real-world authentication logs from the LANL's multi-source cybersecurity events, preprocessed as described in section 3.3.

For ADSAGE and seq2one, we use the same time features as for CERT data. Graph features are the source and destination computer of an authentication event, meaning that we have two attributed edges (user to source computer and user to destination computer). For this reason, we implement two rule-based baselines "known source PC" and "known destination PC", which are equivalent to "known PC" for logon events for each corresponding graph feature. We also run two instances of SedanSpot, one for edges from user to source computer and the other for user to destination computer.

We use following hyperparameters for seq2one and ADSAGE's RNN: 1 layer of 50 LSTM units, 10 timesteps, batch size 512, 15 epochs, learning rate of 0.001 with 0.5 decay factor after 1 epoch without improvement and no dropout. Embeddings of source and destination computers have a dimensionality of 20. For ADSAGE's FFNN we use 3 layers of respectively 50, 30 and 10 units with relu activation with dropout = 0.2. Due to the large volume of events, we train on a 10% random sample of all users.

Cumulative recall values at budgets 1000, 4000 and 12000 are presented in table 6.6. ADSAGE and the "known source pc" rule-based classifier outperform all other methods at all 3 budget values. At maximum budget, their cumulative recall reaches 0.88 and 0.89 respectively (though the difference is not statistically significant). Detection results from SedanSpot and our rule-based classifier also suggest that the source computer attribute in LANL authentication events is more informative than the destination computer. ADSAGE has the advantage to support both attributes simultaneously.

6.1.5 Discussion

Our experimental results on the CERT insider threat datasets show that our method ADSAGE outperforms concurrent approaches to detect intrusions in logon and email events, which are respectively modeled as user to computer and sender to receiver links. For web events, which we regard as user to web domain relations, ADSAGE's detection performance is not better than baselines, but SedanSpot can be used instead. Hence graph representations are useful to detect insider threats in

6.2 (Tuor dataset)	CR-400	CR-1000	CR-4000
[76]	**0.731**	**0.893**	-
max	0.475 ± 0.118	0.753 ± 0.023	$\mathbf{0.938 \pm 0.006}$
mean	0.432 ± 0.050	0.722 ± 0.034	$\mathbf{0.932 \pm 0.009}$
6.2 (all scenarios)	CR-400	CR-1000	CR-4000
max	$\mathbf{0.448 \pm 0.076}$	$\mathbf{0.619 \pm 0.054}$	$\mathbf{0.861 \pm 0.015}$
mean	0.363 ± 0.044	0.528 ± 0.035	$\mathbf{0.845 \pm 0.005}$

TABLE 6.7: Detection results over 5 runs for combining detectors. Two dataset splits are used: the one used by Tuor *et al.* [76] (top) and the one containing all threat scenarios (bottom).

these three types of audit data. However, other event types representing file activities and usage of removable devices are less straightforward to represent as graph edges, thus rule-based classifiers are preferable.

Note that results obtained on LANL authentications and CERT logon events are consistent. This suggests that the CERT datasets are quite realistic, even if they are synthetic.

Detection results split by threat scenarios suggest that anomalies flagged by distinct detectors overlap only partially, thus methods can be complementary in detecting insider threats. By combining anomaly scores obtained from several perspectives (i.e. computed by distinct methods using different audit data sources), we can expect detection performance improvement.

To test whether this is feasible, we combine anomaly scores from detectors using a very simple scheme. For each user, for each day of activity, we compute a daily anomaly score using some aggregation function (mean, max) on all scores collected for that user on that day. We try all possible combinations of detectors (using only one detector for each audit data source) and report best performance in table 6.7. Unfortunately, detection performance does not match results reported by Tuor *et al.* [76], with CR-1000 scores around 0.75 (versus 0.893). CR-4000 scores reach 0.93 (no value available for comparison from Tuor *et al.* [76]). On the dataset with all scenarios, CR-1000 scores are up to 0.62 and CR-4000 up to 0.86. Overall, mean and max aggregation schemes give similar results and improve slightly over using detectors on single audit data sources.

6.2 Properties of audit data graphs

We have previously introduced our method ADSAGE which treats audit events as graph edges with attributes. We have shown that ADSAGE can be straightforwardly applied to different audit data domains where user actions can be seen as interactions with some other entities. For example, authentications can be seen as interactions between users and computers, emails as sender to receivers interactions and web browsing as user to web domain interactions.

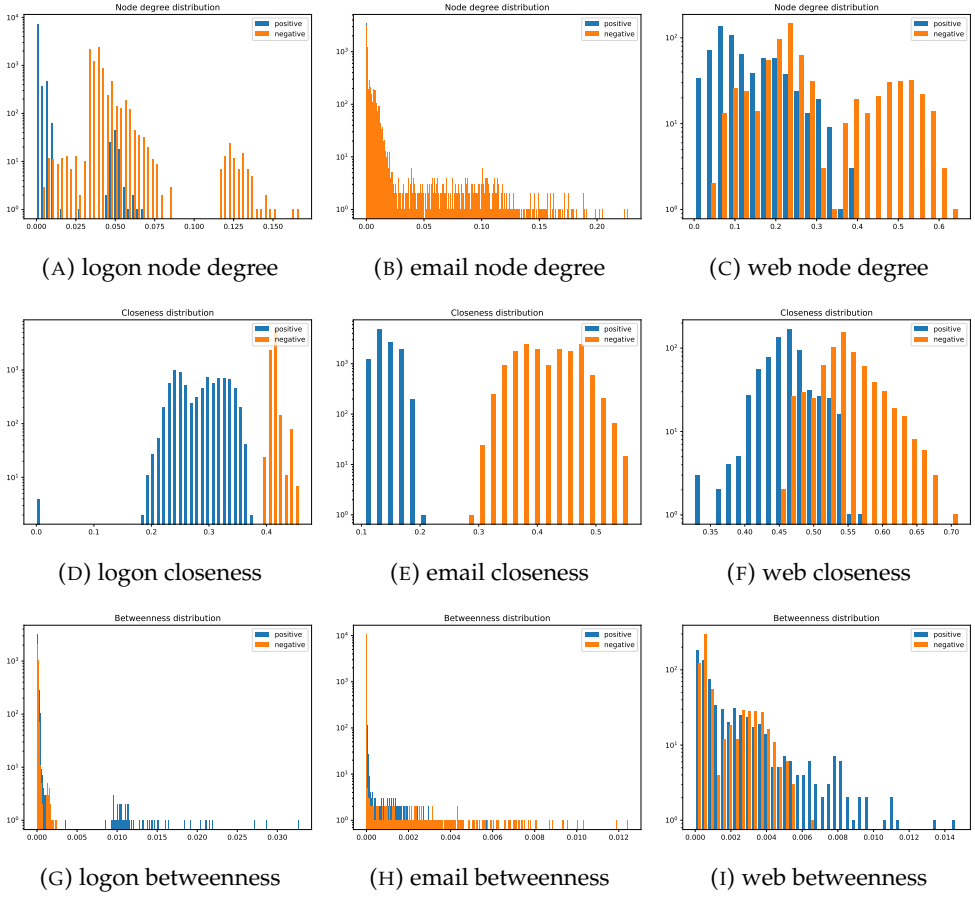
Our experimental results show that ADSAGE is effective to detect anomalies corresponding to insider threats in the authentication and email domain, but not in web browsing events. In the following, our goal is to better understand why this is the case, by inspecting and comparing different properties of the corresponding logon, email and web graphs.

6.2.1 Methodology

We compute and represent the distribution of following graph node centrality measures using NetworkX [174]: node degree, betweenness, and closeness centrality.

For each type of event (logon, email or web), we build two undirected graphs corresponding to normal and anomalous behavior respectively. The normal graph contains positive (i.e. observed) edges, while the anomalous one consists of negative edges, generated by ADSAGE's negative sampling mechanism (i.e. destination entity randomly sampled from the list of unobserved values). Graphs are built using events from January to July 2010, and user sampling is set to 0.1 for email and 0.05 for web (this corresponds to the same setting as in our experiment to characterize detection performance on all threat scenarios).

6.2.2 Results



(A) logon node degree (B) email node degree (C) web node degree

(D) logon closeness (E) email closeness (F) web closeness

(G) logon betweenness (H) email betweenness (I) web betweenness

FIGURE 6.6: Node degree, closeness and betweenness centrality distributions for logon, email and web graphs. Centrality distributions for positive graphs are shown in blue, orange is used for negative graphs.

Figure 6.6 shows the distributions of node degree, closeness and betweenness centrality for the three graph types (logon, email or web). The first notable observation is that logon and email graphs contain a larger amount of nodes than the web graph, which can be explained by the relatively small number of web domains,

compared to the number of computers or email addresses. Node degree distributions show that for all three graphs, positive observations have smaller node degrees than negative samples on average. Nevertheless, the logon graph also contains some normal nodes with high degree; and the web graph shows a significant overlap of normal and anomalous nodes in terms of node degree.

Closeness centrality gives a clearer picture. For the logon and email graphs, the closeness distributions of normal and anomalous nodes are clearly different and do not overlap. In both cases, the graph obtained with negative sampling contains nodes with higher closeness values than the graph representing normal observations. However this is not the case for web graphs, for which node closeness values for the positive and negative observations overlap.

On the contrary, betweenness centrality distributions show no major differences between logon, email and web graphs. In all cases, most positive and negatives nodes concentrate around low betweenness values. Hence betweenness cannot be used as relevant criterion to distinguish normal from anomalous events.

6.2.3 Discussion

In the end, we find that the distribution of closeness node centrality sheds some light on why ADSAGE is effective for anomaly detection in logon and email events, but not web browsing activities. In the web graph, the overlap of closeness values for positive and negative observations suggests that the negative sampling procedure used to train ADSAGE is not adapted. Overlapping closeness values can be interpreted as negative events (i.e. browsing on a random website) not being significantly more unusual than true browsing activities. Hence if negative samples, which are supposed to represent anomalous behavior, are similar to the norm of positive, observed samples then ADSAGE's task of learning the distinction between the two is made more difficult, if not impossible. We thus conclude that web browsing, seen as an interaction between a user and a web domain, involves more randomness than other activities logged in audit data sources of the CERT insider threat dataset.

In order to perform anomaly detection web browsing events, changes in the ADSAGE training procedure are required. In this regard, two complementary modification possibilities are: a. representing web browsing differently (in particular, using a different destination entity for attributed edges) and b. adapting the negative sampling procedure to generate clearer deviations from normal activities.

6.3 Conclusion and perspectives

Surprisingly, graph and text features have been widely ignored in the CERT insider threat use case. In this chapter, we have proposed methods to allow better integration of such heterogeneous data sources in intrusion detection models. Our main contribution is the introduction of ADSAGE, an anomaly detection method for sequences of graph edges with heterogeneous attributes. ADSAGE is the first method to support both edges sequences and attributes. Our method learns to predict the validity of graph edges through negative sampling: unobserved graph edges with true source but wrong destination entity are used as anomalous examples.

We have shown how ADSAGE can be applied for insider threat detection in the CERT case, thus answering research question RQ. a: we regard audit events (log lines) as graph edges and apply anomaly detection at this fine-grained event level, using a single audit data source at a time.

We have compared our method against several baselines, using the evaluation setting from Tuor *et al.* [76], chosen for its realistic recall-based metric. Our experimental results show that ADSAGE is able to detect insider threats in authentication and email data, which are respectively represented as user to computer and sender to receiver edges. For CERT web browsing logs, regarded as user to web domain relations, ADSAGE is not appropriate but other methods such as SedanSpot [129] or rule-based detectors can be used instead. Hence these findings answer RQ. b by showing that fine-grained detection is feasible.

Although we could not meet state-of-the-art performance of [76] when detecting threats over all audit source domains, a direct comparison is unfair as our method relies on a unique audit data source. Still, we believe the performance gap is encouraging given that preprocessing and feature engineering effort are reduced, while alert traceability is improved.

In order to answer RQ. c, we have presented detection results for different threat scenarios using different detectors. Our results showed that most detectors are complementary and combining them could lead to detection improvement. We tested a simple aggregation scheme of anomaly scores, which lead to slight improvements. We believe that further performance gain is possible with more complex aggregation schemes. For instance, a more sophisticated approach could be to run several synchronized instances of ADSAGE (one for each audit data source) sharing their RNN states. This could help provide context from other data sources while still performing fine-grained anomaly detection.

Beyond the choice of detection method and on a more general level, we have found that graph (in particular user to computer relations, email communications) and text features (email contents) from the CERT datasets can be informative to spot insider threats. This provides an answer to RQ. 4, at least in the context of the CERT insider threat use case. Concerning our real-world intrusion detection case, these insights can guide us in extending the current audit data collection to better characterize user behavior.

Chapter 7

Conclusion and perspectives

7.1 Main findings

The main goal of this thesis consists in addressing specific aspects of a real-world intrusion detection case in the identity and access management domain through anomaly detection. Salient characteristics of this problem are as follows. First, real-world audit data sources contain mixed (i.e. numeric and categorical) data, which are not supported by most anomaly detection methods. Second, a significant difficulty lies in the total absence of ground truth. Third, available audit logs are limited as they were not initially collected for intrusion detection purposes. Focusing on two types of intrusions (masquerades and insider threats), we have tackled these challenges through several complementary contributions.

In chapter 4, we have addressed the first two challenges. In order to support mixed attributes (i.e. numeric and categorical data), we have introduced new unsupervised anomaly detection methods based on entropy. Concerning the problem of learning normal versus anomalous user behavior in total absence of ground truth, we have shown that unsupervised intrusion detection can be turned into supervised user identification from audit data activity. We have proposed two methods based on this approach, one of which uses clustering to reduce false positives among similar users. Our empirical evaluation on the real-world dataset of audit sessions has demonstrated the effectiveness of this user identification approach to detect masquerades. However, we have also discovered that the current audit data collection is insufficient to characterize user behavior and that our methods are ineffective against insider threats.

This has lead us to investigate how audit data collection could be extended to better detect insider threats. In this regard, we have targeted the CERT insider threat use case, which contains diverse sources of annotated audit data, providing a sound evaluation framework. We have chosen to focus on heterogeneous data (i.e. categorical, graph and text features), which are often discarded due to incompatibility of many anomaly detection methods. As shown in our literature review, this represents a research gap in the CERT use case: graph and text attributes remain largely underutilized, although they have been found useful to detect insider threats in similar settings. Therefore, we have proposed methods to leverage such heterogeneous data from two complementary perspectives.

On one hand, in chapter 5 we have focused on addressing the "cold start" problem, i.e. the inherent lack of relevant activity history for certain users. Anomaly-based intrusion detection systems usually cannot cope with this issue, although it is highly relevant in practice. Our approach is inspired by zero-shot learning; the central idea consists in compensating the lack of observed user activity data with some semantic description of the corresponding user, which does not depend on activity. To this aim, we have shown how user assignments to organizational units

(like teams, projects, departments) from the CERT use case can be represented as graph features and integrated into an intrusion detection system. As shown by our evaluation, this additional data source allows to improve insider threat detection in cases where user activity data is scarce or irrelevant.

On the other hand, in chapter 6 we have introduced ADSAGE (Anomaly Detection in Sequences of Attributes Graph Edges). This method is designed to detect anomalous graph edges in sequences thereof. We have demonstrated how ADSAGE can be used to leverage heterogeneous data in the CERT use case: log events from several domains can be represented as interactions between a source and destination entity, on which ADSAGE can straightforwardly be applied. These interactions/edges can be augmented with different types of attributes to provide context; hence heterogeneous data is supported. Contrary to concurrent detection systems, ADSAGE is able to compute anomaly scores at fine-grained audit event level without relying on data aggregation and feature engineering. Our empirical evaluation has shown the effectiveness of our method for authentication and email logs; for web browsing logs other methods (such as rule-based classifiers) can be used. Overall, representing audit events as graph edges (i.e. interactions between two entities) with heterogeneous attributes can help detecting insider threats in activity logs.

These two complementary studies suggest that graph features are particularly useful for insider threat detection.

7.2 Perspectives

Throughout this thesis, we have gained valuable insight about the real-world intrusion detection use case proposed by Atos. This has opened up new perspectives on future developments. In the following, we outline some promising research directions to extend this thesis.

Our first extension suggestion is directly applicable to the current audit dataset (see section 3.1) and consists in developing rule-based anomaly detectors for the most relevant categorical features, which are the IP address at authentication time and its derived geolocation. Until more informative attributes are available to characterize user behavior, these features remain the most discriminant across users (as shown in chapter 4). Hence one could focus on detecting anomalies in these features specifically. Moreover, our experiments from chapter 6 have suggested that simple rule-based methods applied to targeted, highly relevant audit data sources can perform surprisingly well. Thus we expect such anomaly detection models to be effective in our real-world use case as well.

In parallel, the current audit data collection should be extended with additional attributes to better characterize user behavior and intentions. This is an important step towards detecting insider threats. Our work presented in chapters 5 and 6 has already explored two complementary approaches to do so: integration of user contextual data and anomaly detection at fine-grained level in heterogeneous data. However we have only addressed the CERT insider threat use case, as it usefully provides ground truth for evaluation. The next natural step is to adapt these approaches to the real-world intrusion detection case. On one hand, using ADSAGE would be straightforward for audit data sources which represent interactions between two entities. On the other hand, the integration of user contextual data in zero-shot learning fashion fits very well into identity and access management, which

relies on and handles user roles, permissions and work group assignments. In a real-world setting, one can expect user contextual data to be of far greater extent than in the CERT use case, which in our opinion is very promising.

Finally, in future work we would like to explore the combination of anomaly detectors based on different audit data sources, such as the ones described in chapter 6. Given the variety of insider threat attacks, useful audit data domains surpass the scope of identity and access management. Considering additional audit data sources can be helpful, but also poses the challenge of how to aggregate different anomaly perspectives effectively. In this regard, an additional dimension to consider is the level of detection. In this thesis, we have explored detection at event (chapter 6), session (chapter 4) and user-day level (chapter 5). We have demonstrated the effectiveness of each of them in different evaluation scenarios. A thorough analysis of which detection level(s) to prefer will be insightful when it comes to combining anomaly detectors from different domains. This is in line with the general tendency of broadening the scope of audit data collection for insider threats, as observed in user and entity behavior analytics.

Bibliography

[1] IBM Security, *2019 Cost of a Data Breach Report*. [Online]. Available: https://databreachcalculator.mybluemix.net/ (visited on 2020-01-17).

[2] Federal Trade Commission, *Equifax to Pay $575 Million as Part of Settlement with FTC, CFPB, and States Related to 2017 Data Breach*. [Online]. Available: https://www.ftc.gov/news-events/press-releases/2019/07/equifax-pay-575-million-part-settlement-ftc-cfpb-states-related (visited on 2020-01-20).

[3] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," National Institute of Standards and Technology, Tech. Rep., 2012.

[4] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Tech. Rep., 2000.

[5] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *arXiv preprint arXiv:1806.03517*, 2018.

[6] Gartner Glossary, *Security Information And Event Management (SIEM)*. [Online]. Available: https://www.gartner.com/en/information-technology/glossary/security-information-and-event-management-siem (visited on 2020-01-20).

[7] T. Bussa, A. Litan, and T. Phillips, *Gartner Market Guide for User and Entity Behavior Analytics*, 2016.

[8] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," Tech. Rep., 2018.

[9] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 305–316, ISBN: 978-1-4244-6894-2. DOI: 10.1109/SP.2010.25.

[10] F. Maggi, W. Robertson, C. Kruegel, and G. Vigna, "Protecting a moving target: Addressing web application concept drift," in *Recent Advances in Intrusion Detection*, E. Kirda, S. Jha, and D. Balzarotti, Eds., Springer Berlin Heidelberg, 2009, pp. 21–40.

[11] M. Garchery and M. Granitzer, "On the influence of categorical features in ranking anomalies using mixed data," *Procedia Computer Science*, vol. 126, pp. 77–86, 2018.

[12] ——, "Identifying and clustering users for unsupervised intrusion detection in corporate audit sessions," in *2019 IEEE International Conference on Cognitive Computing (ICCC)*, IEEE, 2019, pp. 19–27.

[13] S. Zerhoudi, M. Garchery, and M. Granitzer, "Improving intrusion detection systems using zero-shot recognition via graph embeddings," in *IEEE Annual Computer Software and Applications Conference, COMPSAC 2020*, to appear, IEEE, 2020.

[14] M. Garchery and M. Granitzer, "Adsage: Anomaly detection in sequences of attributed graph edges applied to insider threat detection at fine-grained level," *arXiv preprint arXiv:2007.06985*, 2020.

[15] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[16] T. F. Lunt, "A survey of intrusion detection techniques," *Computers & Security*, vol. 12, no. 4, pp. 405–418, 1993.

[17] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[18] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1, pp. 18 –28, 2009, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2008.08.003.

[19] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *expert systems with applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[20] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016, ISSN: 2373-745X. DOI: 10.1109/COMST.2015.2494502.

[21] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," *Statistical science*, pp. 58–74, 2001.

[22] R. A. Maxion and T. N. Townsend, "Masquerade detection using truncated command lines," in *Proceedings international conference on dependable systems and networks*, IEEE, 2002, pp. 219–228.

[23] S. Coull, J. Branch, B. Szymanski, and E. Breimer, "Intrusion detection: A bioinformatics approach," in *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, IEEE, 2003, pp. 24–33.

[24] J. Seo and S. Cha, "Masquerade detection based on svm and sequence-based user commands profile," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, 2007, pp. 398–400.

[25] D. Geng, T. Odaka, J. Kuroiwa, and H. Ogura, "An n-gram and stf-idf model for masquerade detection in a unix environment," *Journal in computer virology*, vol. 7, no. 2, pp. 133–142, 2011.

[26] R. A. Maxion, "Masquerade detection using enriched command lines," in *2003 International Conference on Dependable Systems and Networks, 2003. Proceedings.*, IEEE, 2003, pp. 5–14.

[27] L. Ling, S. Song, and C. Manikopoulos, "Windows nt user profiling for masquerader detection," in *2006 IEEE International Conference on Networking, Sensing and Control*, IEEE, 2006, pp. 386–391.

[28] M. B. Salem and S. J. Stolfo, "Modeling user search behavior for masquerade detection," in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2011, pp. 181–200.

[29] C. Strasburg, S. Krishnan, K. Dorman, S. Basu, and J. S. Wong, "Masquerade detection in network environments," in *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, IEEE, 2010, pp. 38–44.

[30] J. B. Camiña, J. Rodríguez, and R. Monroy, "Towards a masquerade detection system based on user's tasks," in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2014, pp. 447–465.

[31] J. Peng, K.-K. R. Choo, and H. Ashman, "User profiling in intrusion detection: A review," *Journal of Network and Computer Applications*, vol. 72, pp. 14–27, 2016.

[32] S. McKinney and D. S. Reeves, "User identification via process profiling," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, 2009, pp. 1–4.

[33] J. Voris, Y. Song, M. Ben Salem, and S. Stolfo, "You are what you use: An initial study of authenticating mobile users via application usage," in *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services*, ICST (Institute for Computer Sciences, Social-Informatics and ..., 2016, pp. 51–61.

[34] G. Pannell and H. Ashman, "Anomaly detection over user profiles for intrusion detection," *Australian Information Security Management Conference*, 2010-01.

[35] Y. C. Yang, "Web user behavioral profiling for user identification," *Decision Support Systems*, vol. 49, no. 3, pp. 261–271, 2010.

[36] H. H. Nguyen, N. Harbi, and J. Darmont, "An efficient local region and clustering-based ensemble system for intrusion detection," in *Proceedings of the 15th Symposium on International Database Engineering & Applications*, 2011, pp. 185–191.

[37] L. Portnoy, "Intrusion detection with unlabeled data using clustering," Ph.D. dissertation, Columbia University, 2000.

[38] S. H. Oh and W. S. Lee, "An anomaly intrusion detection method by clustering normal user behavior," *Computers & Security*, vol. 22, no. 7, pp. 596–612, 2003.

[39] N. H. Park, S. H. Oh, and W. S. Lee, "Anomaly intrusion detection by clustering transactional audit streams in a host computer," *Information Sciences*, vol. 180, no. 12, pp. 2375–2389, 2010.

[40] J. Hunker and C. W. Probst, "Insiders and insider threats-an overview of definitions and mitigation techniques," *JoWUA*, vol. 2, no. 1, pp. 4–27, 2011.

[41] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–40, 2019.

[42] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A survey of insider attack detection research," in *Insider Attack and Cyber Security*, Springer, 2008, pp. 69–90.

[43] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and preventing cyber insider threats: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1397–1417, 2018.

[44] I. A. Gheyas and A. E. Abdallah, "Detection and prediction of insider threats to cyber security: A systematic literature review and meta-analysis," *Big Data Analytics*, vol. 1, no. 1, p. 6, 2016.

[45] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka, "Multi-domain information fusion for insider threat detection," in *2013 IEEE Security and Privacy Workshops*, IEEE, 2013, pp. 45–51.

[46] M. Dahmane and S. Foucher, "Combating insider threats by user profiling from activity logging data," in *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, IEEE, 2018, pp. 194–199.

[47] P. Parveen and B. Thuraisingham, "Unsupervised incremental sequence learning for insider threat detection," in *2012 IEEE International Conference on Intelligence and Security Informatics*, IEEE, 2012, pp. 141–143.

[48] S. Mathew, M. Petropoulos, H. Q. Ngo, and S. Upadhyaya, "A data-centric approach to insider attack detection in database systems," in *International Workshop on Recent Advances in Intrusion Detection*, Springer, 2010, pp. 382–401.

[49] A. Gamachchi and S. Boztaş, "Web access patterns reveal insiders behavior," in *2015 Seventh International Workshop on Signal Design and its Applications in Communications (IWSDA)*, IEEE, 2015, pp. 70–74.

[50] B. Böse, B. Avasarala, S. Tirthapura, Y. Chung, and D. Steiner, "Detecting insider threats using RADISH: a system for real-Time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, vol. 11, no. 2, pp. 471–482, 2017. DOI: 10.1109/JSYST.2016.2558507.

[51] J. Myers, M. R. Grimaila, and R. F. Mills, "Towards insider threat detection using web server logs," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, 2009, pp. 1–4.

[52] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal, and R. Rolleston, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data.," *JoWUA*, vol. 6, no. 4, pp. 47–63, 2015.

[53] T. E. Senator, H. G. Goldberg, A. Memory, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, D. A. Bader, E. Chow, and et al., "Detecting insider threats in a real corporate database of computer usage activity," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, Chicago, Illinois, USA: Association for Computing Machinery, 2013, 1393–1401, ISBN: 9781450321747. DOI: 10.1145/2487575.2488213.

[54] A. Ambre and N. Shekokar, "Insider threat detection using log analysis and event correlation," *Procedia Computer Science*, vol. 45, pp. 436–445, 2015.

[55] M. Kandias, V. Stavrou, N. Bozovic, and D. Gritzalis, "Proactive insider threat detection through social media: The youtube case," in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, 2013, pp. 261–266.

[56] O. Brdiczka, J. Liu, B. Price, J. Shen, A. Patil, R. Chow, E. Bart, and N. Ducheneaut, "Proactive insider threat detection through graph learning and psychological context," in *2012 IEEE Symposium on Security and Privacy Workshops*, IEEE, 2012, pp. 142–149.

[57] Software Engineering Institute, Carnegie Mellon University, *Insider Threat Test Dataset*. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099 (visited on 2020-02-12).

[58] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," in *Proceedings of the 2013 IEEE Security and Privacy Workshops*, ser. SPW '13, USA: IEEE Computer Society, 2013, 98–104, ISBN: 9780769550176. DOI: 10.1109/SPW.2013.37.

[59] A. D. Kent, "Cybersecurity Data Sources for Dynamic Network Research," in *Dynamic Networks in Cybersecurity*, Imperial College Press, 2015-06.

[60] ——, *Comprehensive, Multi-Source Cyber-Security Events*, Los Alamos National Laboratory, 2015. DOI: 10.17021/1179829.

[61] UCI KDD archive, *KDD Cup 1999 Data*. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (visited on 2020-02-12).

[62] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *Recent Advances in Intrusion Detection*, G. Vigna, C. Kruegel, and E. Jonsson, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 220–237, ISBN: 978-3-540-45248-5.

[63] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, IEEE, 2009, pp. 1–6.

[64] A. Özgür and H. Erdem, "A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ Preprints*, vol. 4, e1954v1, 2016-04, ISSN: 2167-9843. DOI: 10.7287/peerj.preprints.1954v1.

[65] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, "Kdd cup 99 data sets: A perspective on the role of data sets in network intrusion detection research," *Computer*, vol. 52, no. 2, pp. 41–51, 2019, ISSN: 1558-0814. DOI: 10.1109/MC.2018.2888764.

[66] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015-11, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[67] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357 –374, 2012, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2011.12.012.

[68] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSP*, 2018, pp. 108–116.

[69] Matthias Schonlau, *Masquerading User Data*. [Online]. Available: http://www.schonlau.net/intrusion.html (visited on 2020-02-12).

[70] M. B. Salem and S. J. Stolfo, "Modeling user search behavior for masquerade detection," in *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 181–200, ISBN: 978-3-642-23644-0.

[71] J. B. Camiña, C. Hernández-Gracidas, R. Monroy, and L. Trejo, "The windows-users and -intruder simulations logs dataset (wuil): An experimental framework for masquerade detection mechanisms," *Expert Systems with Applications*, vol. 41, no. 3, pp. 919 –930, 2014, Methods and Applications of Artificial and Computational Intelligence, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.08.022.

[72] A. Harilal, F. Toffalini, J. Castellanos, J. Guarnizo, I. Homoliak, and M. Ochoa, "Twos: A dataset of malicious insider threat behavior based on a gamified competition," in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, ser. MIST '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, 45–56, ISBN: 9781450351775. DOI: 10.1145/3139923.3139929.

[73] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, 29–38, 2006-01, ISSN: 0146-4833. DOI: 10.1145/1111322.1111330.

[74] G. Creech, "Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks.," Ph.D. dissertation, University of New South Wales, Canberra, Australia, 2014.

[75] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: A provocative discussion," in *Proceedings of the 2006 Workshop on New Security Paradigms*, ser. NSPW '06, Germany: Association for Computing Machinery, 2006, 21–29, ISBN: 9781595939234. DOI: 10.1145/1278940.1278945.

[76] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[77] Q. Lv, Y. Wang, L. Wang, and D. Wang, "Towards a user and role-based behavior analysis method for insider threat detection," in *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, 2018, pp. 6–10. DOI: 10.1109/ICNIDC.2018.8525804.

[78] T. Rashid, I. Agrafiotis, and J. R. Nurse, "A new take on detecting insider threats: exploring the use of hidden Markov models," in *Proceedings of the 8th ACM CCS International Workshop on Managing Insider Security Threats*, ser. MIST '16, Vienna, Austria: ACM, 2016, pp. 47–56, ISBN: 978-1-4503-4571-2. DOI: 10.1145/2995959.2995964.

[79] F. Yuan, Y. Cao, Y. Shang, Y. Liu, J. Tan, and B. Fang, "Insider threat detection with deep neural network," in *Computational Science – ICCS 2018*, Y. Shi, H. Fu, Y. Tian, V. V. Krzhizhanovskaya, M. H. Lees, J. Dongarra, and P. M. A. Sloot, Eds., Cham: Springer International Publishing, 2018, pp. 43–54, ISBN: 978-3-319-93698-7.

[80] L. Lin, S. Zhong, C. Jia, and K. Chen, "Insider threat detection based on deep belief network feature representation," in *2017 International Conference on Green Informatics (ICGI)*, 2017, pp. 54–59. DOI: 10.1109/ICGI.2017.37.

[81] J. Lu and R. K. Wong, "Insider threat detection with long short-term memory," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW 2019, Sydney, NSW, Australia: ACM, 2019, 1:1–1:10, ISBN: 978-1-4503-6603-8. DOI: 10.1145/3290688.3290692.

[82] D. C. Le and A. N. Zincir-Heywood, "Evaluating insider threat detection workflow using supervised and unsupervised learning," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, pp. 270–275. DOI: 10.1109/SPW.2018.00043.

[83] D. C. Le, S. Khanchi, A. N. Zincir-Heywood, and M. I. Heywood, "Benchmarking evolutionary computation approaches to insider threat detection," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18, Kyoto, Japan: ACM, 2018, pp. 1286–1293, ISBN: 978-1-4503-5618-3. DOI: 10.1145/3205455.3205612.

[84] A. J. Hall, N. Pitropakis, W. J. Buchanan, and N. Moradpoor, "Predicting malicious insider threat scenarios using organizational data and a heterogeneous stack-classifier," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 5034–5039. DOI: 10.1109/BigData.2018.8621922.

[85] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC Plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 10, no. 3, pp. 1–21, 2015-03. DOI: 10.1371/journal.pone.0118432.

[86] A. Taha and A. S. Hadi, "Anomaly detection methods for categorical data: A review," *ACM Comput. Surv.*, vol. 52, no. 2, 2019-05, ISSN: 0360-0300. DOI: 10.1145/3312739. [Online]. Available: https://doi.org/10.1145/3312739.

[87] M. E. Otey, A. Ghoting, and S. Parthasarathy, "Fast distributed outlier detection in mixed-attribute data sets," *Data Mining and Knowledge Discovery*, vol. 12, no. 2-3, pp. 203–228, 2006.

[88] A. Koufakou, M. Georgiopoulos, and G. C. Anagnostopoulos, "Detecting Outliers in High-Dimensional Datasets with Mixed Attributes," *DMIN 2006: Proceedings of the 2008 International Conference on Data Mining*, no. November 2015, pp. 427–433, 2008.

[89] J. X. Yu, W. Qian, H. Lu, and A. Zhou, "Finding centric local outliers in categorical/numerical spaces," *Knowledge and Information Systems*, vol. 9, no. 3, pp. 309–338, 2006.

[90] S. Aryal, K. M. Ting, and G. Haffari, "Revisiting Attribute Independence Assumption in Probabilistic Unsupervised Anomaly Detection," in *PAISI*, vol. 9650, 2016, pp. 73–86.

[91] K. Noto, C. Brodley, and D. Slonim, "Frac: A feature-modeling approach for semi-supervised and unsupervised anomaly detection," *Data mining and knowledge discovery*, vol. 25, no. 1, pp. 109–133, 2012.

[92] L. Rashidi, S. Hashemi, and A. Hamzeh, "Anomaly detection in categorical datasets using bayesian networks," in *International Conference on Artificial Intelligence and Computational Intelligence*, Springer, 2011, pp. 610–619.

[93] Y. Chen, S. Nyemba, W. Zhang, and B. Malin, "Specializing network analysis to detect anomalous insider actions," *Security informatics*, vol. 1, no. 1, p. 5, 2012.

[94] P. Moriano, J. Pendleton, S. Rich, and L. J. Camp, "Insider threat event detection in user-system interactions," in *Proceedings of the 2017 International Workshop on Managing Insider Security Threats*, ser. MIST '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, 1–12, ISBN: 9781450351775. DOI: 10.1145/3139923.3139928.

[95] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *Journal of Applied Security Research*, vol. 6, no. 1, pp. 32–81, 2010. DOI: 10.1080/19361610.2011.529413.

[96] P. Parveen, J. Evans, B. Thuraisingham, K. W. Hamlen, and L. Khan, "Insider threat detection using stream mining and graph mining," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, IEEE, 2011, pp. 1102–1110.

[97] J. S. Okolica, G. L. Peterson, and R. F. Mills, "Using plsi-u to detect insider threats by datamining e-mail," *Int. J. Secur. Netw.*, vol. 3, no. 2, 114–121, 2008-02, ISSN: 1747-8405.

[98] A. Patil, J. Liu, J. Shen, O. Brdiczka, J. Gao, and J. Hanley, "Modeling attrition in organizations from email communication," in *2013 International Conference on Social Computing*, 2013, pp. 331–338. DOI: 10.1109/SocialCom.2013.52.

[99] K. Nance and R. Marty, "Identifying and visualizing the malicious insider threat using bipartite graphs," in *2011 44th Hawaii International Conference on System Sciences*, IEEE, 2011, pp. 1–9.

[100] Q. Althebyan and B. Panda, "A knowledge-base model for insider threat prediction," in *2007 IEEE SMC Information Assurance and Security Workshop*, IEEE, 2007, pp. 239–246.

[101] E. Nwafor, A. Campbell, and G. Bloom, "Anomaly-based intrusion detection of iot device sensor data using provenance graphs," in *1st International Workshop on Security and Privacy for the Internet-of-Things*, vol. 59, 2018.

[102] V. Mavroeidis, K. Vishi, and A. Jøsang, "A framework for data-driven physical security and insider threat detection," in *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2018, pp. 1108–1115. DOI: 10.1109/ASONAM.2018.8508599.

[103] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "Grids-a graph based intrusion detection system for large networks," in *Proceedings of the 19th national information systems security conference*, Baltimore, vol. 1, 1996, pp. 361–370.

[104] S. Haas and M. Fischer, "Gac: Graph-based alert correlation for the detection of distributed multi-step attacks," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18, Pau, France: Association for Computing Machinery, 2018, 979–988, ISBN: 9781450351911. DOI: 10.1145/3167132.3167239.

[105] J. Camacho-Collados and M. T. Pilehvar, "From word to sense embeddings: A survey on vector representations of meaning," *Journal of Artificial Intelligence Research*, vol. 63, pp. 743–788, 2018.

[106] Y. Liao and V. R. Vemuri, "Using text categorization techniques for intrusion detection," in *Proceedings of the 11th USENIX Security Symposium*, USA: USENIX Association, 2002, 51–59, ISBN: 1931971005.

[107] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17, Dallas, Texas, USA: Association for Computing Machinery, 2017, 1285–1298, ISBN: 9781450349468. DOI: 10.1145/3133956.3134015.

[108] A. R. Tuor, R. Baerwolf, N. Knowles, B. Hutchinson, N. Nichols, and R. Jasper, "Recurrent neural network language models for open vocabulary event-level cyber anomaly detection," in *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[109] J. Saxe and K. Berlin, "Expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys," *arXiv preprint arXiv:1702.08568*, 2017.

[110] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Communication Networks*, vol. 2018, 2018.

[111] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[112] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[113] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[114] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[115] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," *arXiv preprint arXiv:1511.08198*, 2015.

[116] X. Zhuo, J. Zhang, and S. W. Son, "Network intrusion detection using word embeddings," in *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, 2017, pp. 4686–4695.

[117] J. Cui, J. Long, E. Min, and Y. Mao, "Wedl-nids: Improving network intrusion detection using word embedding-based deep learning method," in *International Conference on Modeling Decisions for Artificial Intelligence*, Springer, 2018, pp. 283–295.

[118] C. R. Brown, A. Watkins, and F. L. Greitzer, "Predicting insider threat risks through linguistic analysis of electronic communication," in *2013 46th Hawaii International Conference on System Sciences*, IEEE, 2013, pp. 1849–1858.

[119] M. Mayhew, M. Atighetchi, A. Adler, and R. Greenstadt, "Use of machine learning in big data analytics for insider threat detection," in *MILCOM 2015-2015 IEEE Military Communications Conference*, IEEE, 2015, pp. 915–922.

[120] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, vol. 11, no. 2, pp. 503–512, 2017. DOI: 10.1109/JSYST.2015.2438442.

[121] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: LIWC and computerized text analysis methods," *Journal of language and social psychology*, vol. 29, no. 1, pp. 24–54, 2010.

[122] I. Agrafiotis, P. A. Legg, M. Goldsmith, and S. Creese, "Towards a user and role-based sequential behavioural analysis tool for insider threat detection.," *J. Internet Serv. Inf. Secur.*, vol. 4, no. 4, pp. 127–137, 2014.

[123] A. Gamachchi, L. Sun, and S. Boztas, "A graph based framework for malicious insider threat detection," *arXiv preprint arXiv:1809.00141*, 2018.

[124] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, 2015-05, ISSN: 1384-5810. DOI: 10.1007/s10618-014-0365-y.

[125] S. Ranshous, S. Shen, D. Koutra, S. Harenberg, C. Faloutsos, and N. F. Samatova, "Anomaly detection in dynamic networks: A survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 7, no. 3, pp. 223–247, 2015. DOI: 10.1002/wics.1347.

[126] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1035–1044.

[127] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "Spotlight: Detecting anomalies in streaming graphs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18, London, United Kingdom: ACM, 2018, pp. 1378–1386, ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3220040.

[128] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, D. Makhija, M. Kumar, and C. Faloutsos, "Edgecentric: Anomaly detection in edge-attributed networks," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, pp. 327–334. DOI: 10.1109/ICDMW.2016.0053.

[129] D. Eswaran and C. Faloutsos, "Sedanspot: Detecting anomalies in edge streams," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 953–958. DOI: 10.1109/ICDM.2018.00117.

[130] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18, London, United Kingdom: ACM, 2018, pp. 2672–2681, ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3220024.

[131] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, 2019-07, pp. 4419–4425. DOI: 10.24963/ijcai.2019/614.

[132] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos, "Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '19, Anchorage, AK, USA: ACM, 2019, pp. 647–657, ISBN: 978-1-4503-6201-6. DOI: 10.1145/3292500.3330946.

[133] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova, "A scalable approach for outlier detection in edge streams using sketch-based approximations," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 189–197. DOI: 10.1137/1.9781611974348.22.

[134] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019.

[135] D. Jayaraman and K. Grauman, "Zero-shot recognition with unreliable attributes," in *Advances in neural information processing systems*, 2014, pp. 3464–3472.

[136] Z. Fu, T. Xiang, E. Kodirov, and S. Gong, "Zero-shot object recognition by semantic manifold distance," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2635–2644.

[137] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Advances in neural information processing systems*, 2009, pp. 1410–1418.

[138] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 951–958.

[139] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4582–4591.

[140] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[141] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," in *Advances in neural information processing systems*, 2016, pp. 3630–3638.

[142] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[143] M. M. U. Chowdhury, F. Hammond, G. Konowicz, C. Xin, H. Wu, and J. Li, "A few-shot deep learning approach for improved intrusion detection," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, IEEE, 2017, pp. 456–462.

[144] Z. Li, Z. Qin, P. Shen, and L. Jiang, "Zero-shot learning for intrusion detection via attribute representation," in *International Conference on Neural Information Processing*, Springer, 2019, pp. 352–364.

[145] J. L. R. Pérez and B. Ribeiro, "Attribute learning for network intrusion detection," in *INNS Conference on Big Data*, Springer, 2016, pp. 39–49.

[146] J. Rivero, B. Ribeiro, N. Chen, and F. S. Leite, "A grassmannian approach to zero-shot learning for network intrusion detection," in *International Conference on Neural Information Processing*, Springer, 2017, pp. 565–575.

[147] Z. Zhuang, X. Kong, E. Rundensteiner, A. Arora, and J. Zouaoui, "One-shot learning on attributed sequences," in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, 2018, pp. 921–930.

[148] A. Puzanov and K. Cohen, "Deep reinforcement one-shot learning for artificially intelligent classification systems," *arXiv preprint arXiv:1808.01527*, 2018.

[149] P. Robyns, E. Marin, W. Lamotte, P. Quax, D. Singelée, and B. Preneel, "Physical-layer fingerprinting of lora devices using supervised and zero-shot learning," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2017, pp. 58–63.

[150] T. K. Tran, H. Sato, and M. Kubo, "One-shot learning approach for unknown malware classification," in *2018 5th Asian Conference on Defense Technology (ACDT)*, IEEE, 2018, pp. 8–13.

[151] Maxmind, *GeoLite2 Free Downloadable Databases*. [Online]. Available: https://dev.maxmind.com/geoip/geoip2/geolite2/ (visited on 2020-03-10).

[152] Internet Archive, *Wayback Machine*. [Online]. Available: https://web.archive.org/ (visited on 2020-03-10).

[153] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[154] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[155] H2O.ai, *Distributed Random Forest*. [Online]. Available: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html (visited on 2020-03-18).

[156] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC bioinformatics*, vol. 8, no. 1, p. 25, 2007.

[157] F. T. Liu and K. M. Ting, "Isolation Forest," *ICDM'08. Eighth IEEE International Conference on Data Mining*, 2008.

[158] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, "A Meta-Analysis of the Anomaly Detection Problem," 2015.

[159] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[160] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Kdd-96*, pp. 226–231, 1996, ISSN: 09758887. DOI: 10.1.1.71.1980. arXiv: 10.1.1.71.1980.

[161] L. Rokach and O. Maimon, "Clustering Methods," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds. Boston, MA: Springer US, 2005, pp. 321–352, ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_15.

[162] K. Bhatia, K. Dahiya, H. Jain, A. Mittal, Y. Prabhu, and M. Varma, *The extreme classification repository: Multi-label datasets and code*, 2016. [Online]. Available: http://manikvarma.org/downloads/XC/XMLRepository.html (visited on 2020-06-29).

[163] S. Bengio, K. Dembczynski, T. Joachims, M. Kloft, and M. Varma, "Extreme classification (dagstuhl seminar 18291)," in *Dagstuhl Reports*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 8, 2019.

[164] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[165] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[166] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[167] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[168] S. Zerhoudi, "Zero-shot recognition via graph embeddings applied to intrusion detection systems," M.S. thesis, University of Passsau, Germany, 2019.

[169] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[170] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[171] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 1225–1234.

[172] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "Struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 385–394.

[173] Pytorch, *Embedding bag*. [Online]. Available: https://pytorch.org/docs/stable/nn.html?highlight=embeddingbag#torch.nn.EmbeddingBag (visited on 2019-10-22).

[174] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.