FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Jari Jääskelä

Anomaly-Based Insider Threat Detection with Expert Feedback and Descriptions

Master's Thesis
Degree Programme in Computer Science and Engineering
March 2020

Jääskelä J. (2020) Anomaly-Based Insider Threat Detection with Expert Feedback and Descriptions . University of Oulu, Degree Programme in Computer Science and Engineering, 56 p.

ABSTRACT

Insider threat is one of the most significant security risks for organizations, hence insider threat detection is an important task. Anomaly detection is a one approach to insider threat detection. Anomaly detection techniques can be categorized into three categories with respect to how much labelled data is needed: unsupervised, semi-supervised and supervised.

Obtaining accurate labels of all kinds of incidents for supervised learning is often expensive and impractical. Unsupervised methods do not require labelled data, but they have a high false positive rate because they operate on the assumption that anomalies are rarer than nominals. This can be mitigated by introducing feedback, known as expert-feedback or active learning. This allows the analyst to label a subset of the data. Another problem is the fact that models often are not interpretable, thus it is unclear why the model decided that a data instance is an anomaly.

This thesis presents a literature review of insider threat detection, unsupervised and semi-supervised anomaly detection. The performance of various unsupervised anomaly detectors are evaluated. Knowledge is introduced into the system by using state-of-the-art feedback technique for ensembles, known as active anomaly detection, which is incorporated into the anomaly detector, known as isolation forest. Additionally, to improve interpretability techniques of creating rule-based descriptions for the isolation forest are evaluated.

Experiments were performed on CMU-CERT dataset, which is the only publicly available insider threat dataset with logon, removable device and HTTP log data. Models use usage count and session-based features that are computed for users on every day. The results show that active anomaly detection helps in ranking true positives higher on the list, lowering the amount of data analysts have to analyse. Results also show that both compact description and Bayesian rulesets have the potential to be used in generating decision-rules that aid in analysing incidents; however, these rules are not correct in every instance.

Keywords: anomaly detection, unsupervised, semi-supervised, isolation forest, IForest, IF, one-class support vector machine, OC-SVM, local outlier factor, LOF, ensemble, active learning, active anomaly detection, AAD, compact descriptions, Bayesian rulesets

Jääskelä J. (2020) Poikkeamapohjainen sisäpiiriuhkien havainta palautteen ja kuvauksien avulla. Oulun yliopisto, Tietotekniikan tutkinto-ohjelma, 56 s.

TIIVISTELMÄ

Sisäpiirinuhat ovat yksi vakavimmista riskeistä organisaatioille. Tästä syystä sisäpiiriuhkien havaitseminen on tärkeää. Sisäpiiriuhkia voidaan havaita poikkeamien havaitsemismenetelmillä. Nämä menetelmät voidaan luokitella kolmeen oppimisluokkaan saatavilla olevan tietomäärän perusteella: ohjaamaton, puoli-ohjatta ja ohjattu.

Täysin oikein merkatun tiedon saaminen ohjattua oppimista varten voi olla hyvin kallista ja epäkäytännöllistä. Ohjaamattomat oppimismenetelmät eivät vaadi merkattua tietoa, mutta väärien positiivisten osuus on suurempi, koska nämä menetelmät perustuvat oletukseen että poikkeamat ovat harvinaisempia kuin normaalit tapaukset. Väärien positiivisten osuutta voidaan pienentää ottamalla käyttöön palaute, jolloin analyytikko voi merkata osan datasta.

Tässä opinnäytetyössä tutustutaan ensin sisäpiiriuhkien havaitsemiseen, mitä tutkimuksia on tehty ja ohjaamattomaan ja puoli-ohjattuun poikkeamien havaitsemiseen. Muutamien lupaavien ohjaamattomien poikkeamatunnistimien toimintakyky arvioidaan. Järjestelmään lisätään tietoisuutta havaitsemisongelmasta käyttämällä urauurtavaa aktiivista poikkeamahavainta (engl. active anomaly detection) -palautemetelmää, joka on tehty havaitsinjoukoille (engl. ensembles). Tätä arvioidaan Isolation Forest -havaitsimen kanssa. Lisäksi, jotta analytiikko pystyisi paremmin käsittelemään havainnot, tässä työssä myös arvioidaan sääntöpohjaisten kuvausten luontimenetelmä Isolation Forest -havaitsimelle. Kokeilut suoritettiin käyttäen julkista CMU-CERT:in aineistoa, joka on ainoa julkinen aineisto, missä on muun muuassa kirjautumis-, USB-laite- ja HTTP-tapahtumia.

Mallit käyttävät käyttöluku- ja istuntopohjaisia piirteitä, jotka luodaan jokaista käyttäjää ja päivää kohti. Tuloksien perusteella aktiivinen poikkeamahavainta auttaa epäilyttävämpien tapahtumien sijoittamisessa listan kärkeen vähentäen tiedon määrä, jonka analyytikon tarvitsee tutkia. Kompaktikuvakset (engl. compact descriptions)- ja Bayesian sääntöjoukko -menetelmät pystyvät luomaan sääntöjä, jotka kuvaavat minkä takia tapahtuma on epäilyttävä, mutta nämä säännöt eivät aina ole oikein.

Avainsanat: ohjaamaton, puoli-ohjattu, eristysmetsä, yksiluokkainen tukivektorikone, havaitsinjoukko, aktiivinen oppiminen, aktiivinen poikkeamahavainta, kompaktikuvaukset, Bayesian sääntöjoukko

Contents

FOREWORD

This thesis was done in collaboration with Netox Ltd under the Security and Software Engineering Center (S^2ERC) while I worked at the Oulu University Secure Programming Group (OUSPG).

I would like to thank M.Sc. Teemu Tokola (technical supervisor), Prof. Juha Röning (principal supervisor), D.Sc. Ulrico Celentano (second examiner) and staff at Netox for their advice and support.

Oulu, Finland 10th March, 2020

Jari Jääskelä

LIST OF ABBREVIATIONS AND SYMBOLS

$\mathbf{w}$	weight vector
θ	fraction of anomalies
AAD	active anomaly detection
BR	Bayesian rule sets
CD	compact description
FN	false negative
FP	false positive
FPR	false positive rate
HIDS	host-based IDS
IDS	intrusion detection system
IForest	isolation forest
LOF	local outlier factor
LSTM	long short-term memory neural network
NIDS	network-based IDS
OC-SVM	one-class SVM
ROC	receiver operation charasteristic
ROC-AUC	area under the ROC curve
recall@k	recall in the top-k instances
SIEM	security information and event management
SVM	support vector machine
TN	true negative
TNR	True Negative Rate
TP	true positive
TPR	True Positive Rate

1. INTRODUCTION

Insider threat is a security risk that originates within the target organization. Insiders have knowledge about the organization's computers, network and policies, and they have authorization. For example, insiders may be current or former employees, or contractors. Insider threats manifest primarily as intellectual property theft, information technology sabotage, fraud and espionage. [1]

The Global Economic Crime and Fraud Survey [2] conducted by PwC in 2018 found that 53 % of US companies had been affected by fraud during the previous 24 months. The share of frauds committed by insiders rose from 29 % in 2016 to 43 % in 2018. In the global survey conducted by Vormetric, Inc. in 2015, only 11 % of organizations reported that they are safe from insider threats [3].

The 2019 Data Breach Investigation Report [4] conducted by Verizon found that 34 % of data breaches involve internal actors. This share was the largest in the healthcare industry, where most (59 %) breaches are associated with internal actors.

Threats can be detected using signature-based and anomaly-based tools. Signature-based tools require the development of signatures for specific threats; hence, they cannot adapt to unknown threats. For the detection of such threats, an anomaly-based approach is required.

Anomaly detection can be performed using supervised, semi-supervised, or unsupervised techniques. Since supervised techniques require labelling large amounts of data, which is often infeasible in the real-world environment the focus here is on semi-supervised and unsupervised techniques. Unsupervised techniques have high false positive rate because they tend to detect all abnormal events, and this may not completely align with what the security analyst is interested in. To achieve lower false positive rate, the security analyst should be able to improve the anomaly detector by providing feedback. This is known as active learning, and it can be achieved using active anomaly detection (AAD). The objective of this thesis is to evaluate this technique.

Additionally, it is crucial for the system decisions to be interpretable by the analyst, so that incidents can be investigated more effectively. The system should be able to display the most prominent evidence to the analyst so that they can respond accordingly. For this reason, another objective of this thesis is to evaluate recently developed description methods.

In section 2, insider threat detection, related systems and research are introduced. Anomaly detection and unsupervised anomaly detection techniques are then introduced. Common unsupervised anomaly detection methods and isolation forest (IForest) with AAD are evaluated using the public insider threat dataset by CMU-CERT. Finally, compact description (CD) and Bayesian rule sets (BR), which are rule-based description methods, are evaluated for IForest. These description methods may allow an analyst to analyse potential incidents more effectively.

2. INSIDER THREAT DETECTION

The literature offers several similar definitions of insider threat. Security information and event management (SIEM) software and intrusion detection system (IDS) are not specifically intended for insider threat detection, but they can be used for this purpose. Several studies of insider threat detection have been conducted in the literature, and they are introduced in the following subsections.

2.1. Definition of insider threat

Brackney and Anderson [5] define an insider as "an already trusted person with access to sensitive information and information systems." Pfleeger et al. [6] define an insider as "a person with legitimate access to an organization's computers and networks." According to Spitzner [7], an insider has access to critical information, knows the structure of the organization, and is most likely seeking information, rather than information technology systems. The access to the information may be authorized; it is the use of the information that comprises the threat. Greitzer and Frincke [8] define an insider as "an individual currently or at one time authorized to access an organization's information system, data, or network."

The CMU-CERT Insider Threat Center [1] define malicious insider and unintentionl insider threat separately. They define malicious insider as "a current or former employee, contractor, or business partner" who "has or had authorized access to an organization's network, system, or data" and "has intentionally exceeded or intentionally used that access in a manner that negatively affected the confidentiality, integrity, availability, or physical well-being of the organization's information or information systems or workforce."

In [9], the CMU-CERT define an unintentional insider threat as a "a current or former employee, contractor, or business partner" who "through action or inaction without malicious intent causes harm or substantially increases the probability of future serious harm to the confidentiality, integrity, or availability of the organization's information or information system."

Greitzer and Frincke [8] define insider threat as "harmful acts that trusted insiders might carry out; for example, something that causes harm to an organization, or an unauthorized act that benefits the individual." Pfleeger et al. [6] define insider threat as "an insider's action that puts at risk an organization's data, processes, or resources in a disruptive or unwelcome way." Hunker and Probst [10] define insider threat as "an individual with privileges who misuses them or whose access results in misuse."

In this work, the distinction between malicious and unintentional insider threat is unnecessary. Insider threat is considered to be a harmful action by a person that has or had authorized access to the organization's network, systems, or data.

Table 1. Potential data sources for insider threat detection

Data source	Description
Logon logs	Analyse when employees log on and to which systems.
Physical access logs	Identify unusual work hours or correlate with other sources, such as logon activity.
File access logs	Identify unusual file access or access to confidential data.
Email logs	Identify relevant communications, such as communication with competitors or phishing.
HTTP/DNS logs	Analyse what websites employees browse.
Network monitoring logs	Identify unusual traffic patterns, such as insider exfiltrating data.

2.2. Security information and event management

Event correlation is the process of examining events to find relationships between them. In general, this can help in determining the underlying cause of a problem. Event correlation is therefore an important technique in insider threat detection and can be achieved using SIEM solutions.

SIEMs generally have following capabilities [11, 12]:

1. **Data aggregation**: Aggregates data across many sources.
2. **Correlation**: Integrates different data sources through various correlation techniques, to find useful information.
3. **Alerting**: Provides automatic analysis of correlated events and produces alerts.
4. **Presentation**: Produces informational charts using event information and assists in the detection of patterns and the identification of activity that deviates from standard patterns.
5. **Compliance**: Applications can be used to automate the gathering of compliance data, producing reports that can be adapted to existing security, governance and auditing processes.
6. **Retention**: Employs long-term storage of historical data for correlation over time, and for forensics and compliance reasons.

Additionally, recent SIEMs also have to some extent, the capability of analysing user and entity behaviour using machine learning [13, 14]. Insider threat monitoring can incorporate information from multiple data sources, such as logon logs, email logs, file access logs, physical access logs, and web browsing logs (Table 1). For a more comprehensive list of possible data sources, see [1] by CMU-CERT.

2.3. Intrusion detection system

An IDS is either a standalone system or a component of an SIEM system. It detects unauthorized behaviour in a system or network. Intrusion detection

systems are mainly designed to detect external actors, but because many technical actions overlap with those of internal actors, an IDS can be useful in insider threat detection.

Satria et al. [15] define an IDS as a collection of tools, methods and resources that assist in the detection, assessment and reporting of unauthorized access. According to Kemmerer and Vigna [16], an IDS cannot detect intrusions itself, but can detect evidence of them as they are ongoing or after they have occurred.

These systems can collect information by monitoring network traffic or by analysing events from information systems; for example, by examining log files or by looking for policy violations or rare events [17].

Intrusion detection systems are commonly classified based on the data source [17]:

1. **Network-based IDS (NIDS)**: These IDSes detect intrusions in the network data. Intrusion usually manifest as irregular patterns. NIDS analyses and models traffic to identify regular traffic and suspicious activity. These IDSes are capable of gathering and analysing entire packets, including payload, internet protocol addresses and ports. However, a NIDS may have some difficulties processing all network traffic during periods of high volume traffic, potentially leading to the failure to recognize an attack. Another disadvantage is that in the case of encrypted traffic, a NIDS may not be able to analyse the data.

2. **Host-based IDS (HIDS)**: These IDSes operate on a single host. They focus on identifying local suspicious behaviour, such as attacks against the host and attacks perpetrated by the user on the host. They have been used for various tasks, such as monitoring system calls, code analysis, detection of buffer overflows, privilege misuse and analysis of system logs. They are classified as agent-based systems, because they require installation of software on the host. Host-based IDSes are vital, because they can detect attacks that NIDS cannot, such as those using encrypted traffic.

By the detection technique, IDSes are classified as anomaly-based, signature-based (rule-based) or hybrid. Anomaly-based techniques are also referred to as profile-based [17]. In 1987, Denning [18] suggested that security violations could be detected by looking for anomalies in audit records.

In anomaly-based techniques, the operator does not have to configure the system to detect threats; it automatically learns to detect them. In this technique, a profile is created that describes the nominal behaviour for a target. The target may be a system, a user, or some other resource or group of resources. The profile is used as a baseline, to which the observed behaviour is compared. The profile is created based on data gathered over a defined period; for example, a day, a week or a month. This period is called the training period. If the profile is not updated automatically it is referred to as static; if the profile is updated when new events are observed, it is referred to as dynamic. The disadvantage of a static profile is that it cannot adapt to changing behaviour that is considered nominal in the environment. [19]

Dynamic profiles also have disadvantages [17, 20]. These profiles may adapt to consider malicious behaviour as nominal if the behaviour changes gradually, staying below the threshold of detection. Such a response is called a false negative. As the system does not have any preconceptions of how threats manifest, it can detect both known and unknown threats. For this reason, anomaly-based detection is a popular technique in IDS systems.

Another benefit of this technique is that it can detect internal threats. For example, a threat is detected if a malicious actor steals an account and exhibits behaviour that is outside the nominal behaviour of the legitimate account holder. As profiles are different for different targets, it is not easy for an intruder to know what kind of behaviour is outside the norm of the target. [21]

The disadvantage of this technique is that it detects all anomalies, but not all anomalies are unauthorized or malicious. Building accurate profiles can be difficult because the system and user behaviour can be complex. For example, maintenance or backup operations may not be included in the data used in the training phase, and so they would trigger an alert. For this reason, anomaly-based techniques can result in a high false positive rate (FPR), especially when the behaviour of users and systems varies significantly. [19]

Signature-based detectors use well-defined signatures and rules. When the system detects a threat that matches a signature defined in its database, it triggers an alert. This technique is highly accurate and has a low FPR. The major disadvantage is that, if a threat does not have a signature in the database the system cannot detect it. Hybrid systems implement combination of anomaly-based and signature-based techniques. [17]

2.4. Related research

This section reviews studies of anomaly-based insider threat detection.

Table 2. Related research on anomaly-based insider threat detection

Paper	Detection techniques	Granularity	Feedback	Interpretability
[22]	Clustering, Markov Model, and rarest change model	Day		
[23]	Hidden Markov Model, Graph Analysis, GMM, many others	Day		✓
[24]	IForest, Random Forest	Day		✓
[25]	PCA	Day	✓	
[26]	IForest	-		
[27]	CDF, mean and variance, IForest	Day		✓
[28]	Ensembles of individual detectors, IForest, GMM, many others	Day		
[29]	LSTM	Day		✓
[30]	CNN, LSTM	Day		
[31]	PCA, IForest	Day		
[32]	IForest, Graph Analysis	-		
[33]	Deep Autoencoder	Day		
[34]	Deep Autoencoder	Day		✓
[35]	Logistic Regression, Random Forest, Artificial Neural Network	Day, week, session	✓	
[36]	LSTM	Day		
[37]	LSTM	Session		

For each study, Table 2 shows which detection techniques and granularities were evaluated and whether feedback or interpretability was addressed. A lot of insider threat detection research [22–24, 28, 38] has been supported by DARPA

through the Anomaly Detection at Multiple Scales programme. This programme aims to detect anomalies in massive datasets to detect insider threats.

Goldberg et al. [27, 28] and Senator et al. [23] used detection system called PRODIGAL. They used detectors such as IForest, Gaussian mixture model and hidden Markov model.

Gavai et al. [24] detected anomalous behaviour using IForest and features extracted from social data including email communication patterns and online activity, such as web browsing. Brdiczka et al. [38] combined structural anomaly detection in social networks and psychological profiling of individuals. Their structural anomaly detection consisted of graph analysis, dynamic tracking and machine learning. Goldberg et al. [27] proposed methods that explain and aggregate user-days. User-days are one day time periods that include monitored activity for a particular user on that day. These methods involve the temporal aggregation of user-days to find users with multiple high-ranking days. Qiujian et al. [31] partitioned the day into four periods and used principal component analysis to select significant features as a pre-processing step before the application of IForest.

Sun et al. [26] used IForest and applied it successfully to an enterprise environment. In their approach, if the score was above a specific threshold it was considered anomalous, and the user was flagged. Gamachchi et al. [32] introduced a framework for insider threat detection based on graphical analysis and various anomaly detection approaches. In the framework, they used IForest because it has linear time complexity and low memory requirements.

Recently, many approaches based on deep learning have been proposed for insider threat detection. For example, Tuor et al. [29] presented unsupervised learning approach to detect anomalies from logs in real time. They developed two variants of their system: one uses a deep neural network and the another uses long short-term memory neural network (LSTM).

The deep learning approach proposed by Yuan et al. [30] uses LSTM to create abstracted temporal features for user behaviour from user action sequences. While the approach of Tuor et al. [29] can miss anomalous behaviour that occurs within one day, the model proposed by Yuan et al. is trained using user-action sequences and can therefore, detect anomalous behaviour that occurs within one day. Similarly, Lu et al. [36] proposed an LSTM-based approach, in which the parameters were tuned for best performance. The system has higher recall than principal component analysis or support-vector machine (SVM) methods.

Liu et al. [34] proposed a deep autoencoder-based system. Each autoencoder was trained on a subset of the data from one data source, which represents user's nominal behaviour. They created four detectors, one for each data source (logon/logoff logs, removable device logs, HTTP logs, file logs). Using the reconstruction error, top-N recommendation algorithm was applied to each detector. The result was four dimensional normalized top anomalous vectors. These feature vectors are standardized with respect to the user and the averaged reconstruction loss is used as maliciousness score. The top-N algorithm was used again to report user-days with the largest maliciousness scores. Autoencoders were optimized for audit data empirically. Both Yuan et al. [30] and Liu et

al. [34] chose approaches based on deep learning to mitigate the amount of feature engineering required.

Chattopadhyay et al. [33] proposed a supervised deep autoencoder-based approach. They computed statistical measures from the time series of each feature to preserve the variation of user behaviour. For this, they used a sliding time window. These statistical feature vectors were concatenated into a single feature vector.

Legg et al. [25] proposed an unsupervised system, which allows the analyst to provide feedback. Le et al. [35] proposed supervised machine learning methods and that incorporates feedback. For detecting cyber-attacks, Siddiqui et al. [39] developed a system that uses IForest for attack detection. It provides explanations of detected anomalies and improves detection using feedback.

Most approaches report insiders using a granularity of one day [23, 24, 27, 29, 30, 34–36, 38]. This generally means that one feature vector is constructed for each entity (user or role) per day. Only a few approaches have used week [35] or session [35, 37] granularities or partitioned the day into time periods [36, 37], such as morning, midday, evening and night. Such a partitioning is based on the notion that users behave differently during these periods. Liu et al. [34] chose to partition day into hours.

Most approaches use frequency features (specifically the number of actions over a period, such as number of logons, number of computers used and number of emails sent) and statistical features (e.g., time of the first and last logon and mean and standard deviation of transmitted data size) [22–25, 27, 28, 31, 33, 33, 35].

Additionally, [23, 27] use a ratio of features computed from other features. [25, 33, 35] also used session features, such as session duration.

Some papers addressed interpretability in the context of insider threat detection by using a modified IForest algorithm [24], visualization tools [25], single feature anomaly scores [23, 27, 29, 34] or drop-out explanations [27].

3. ANOMALY DETECTION

Anomalies are considered to be patterns in data that deviate from the expected. Depending on the context, anomalies are also known as outliers, discordant observations, exceptions, aberrations, surprises, peculiarities and contaminants [40].

Hawkins [41] defined an anomaly (outlier) as follows: "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism." Anomaly detection techniques have been widely used in various domains, such as intrusion detection [42], fraud detection [43] and fault detection [44].

There are several challenges in anomaly detection [40]. As mentioned previously, in the context of intrusion detection, in which the detected anomalies are caused by malicious behaviour, the adversaries tend to adapt their behaviour to make it appear nominal. The notion of nominal behaviour may evolve in the environment and the current notion of nominal behaviour may not be representative of nominal behaviour in the future. This is known as concept drift [45].

Moreover, the lack of availability of labelled data for the training and validation of models is a major issue. Furthermore, there is no such thing as the best the anomaly detector across all domains. The performance of the detector depends on how well the detector's definition of an anomaly matches the analyst's conception of an "interesting anomaly" [46].

Because of these challenges anomaly detection is difficult in general form. Most existing techniques focus on solving a specific formulation of the problem. Usually, the factors are defined by the domain in which anomalies are detected. [40]

Additionally, computational power, memory and real-time constraints should be considered when determining which techniques can be used in the real-world. Anomaly detection can be performed offline (also known as batch mode) if all data instances are available. Most anomaly detection techniques operate in the offline mode. In the online mode, data instances often arrive in real-time sequentially in data streams. [45, 47]

3.1. Types of anomaly detection techniques

Anomaly detection techniques use a supervised, semi-supervised, or unsupervised approach. Supervised approaches have higher detection rates and lower FPRs than unsupervised approaches; however, unsupervised approaches can detect unknown behaviour, but supervised approaches cannot. Obtaining accurate and representative data of all types of behaviours is often expensive. Labelling is usually performed by an expert. Obtaining data of anomalous behaviour is more difficult than obtaining data of nominal behaviour. Often, the data are dynamic in nature; for instance, new anomalies may arise and old anomalies may no longer be considered anomalous. [48]

In a supervised approach, the dataset is assumed to be fully and correctly labelled. In a semi-supervised approach, the training data consists only of only nominal data. [49]

Unsupervised approaches do not require training data; therefore, such approaches are the most widely applicable. They assume that nominal data instances are more common than anomalies [40]. If this assumption is not true unsupervised approaches suffer from high FPRs. Because unsupervised approaches do not require any labels, they can detect unknown anomalies. Usually, these approaches use clustering, distance or density-based methods to identify outliers within a dataset. [48, 49]

3.2. Types of anomalies

Anomalies can be classified into three types: point anomalies, contextual anomalies and collective anomalies. A point anomaly is an instance of data which has been found to be anomalous with respect to the rest of the data. This is the simplest type of anomaly. [40]

A data instance is a contextual anomaly only in a specific context. The notion of context is derived from the structure of the data and it is specified as part of the problem. Each data instance is specified using contextual and behavioural attributes. Contextual attributes are used to determine the context for the data instance and behavioural attributes define non-contextual characteristics. [40]

In a collective anomaly, related data instances are anomalous with respect to the rest of the dataset. Individual data instances may not be anomalies themselves, but when they occur together they are anomalous; that is, a specific sequence is considered anomalous, not the individual data instances. [40]

Anomalies can be further classifies as: local or global. For example, when there are multiple clusters and points that are nearby and far away from clusters, points that are far away from clusters are considered global anomalies. When neglecting these points and focusing only on the clusters and points nearby, the points that are near to clusters can be seen as anomalies. These are called local anomalies. [49]

3.3. Anomaly detection output

Two techniques exist for reporting anomalies [40, 49]: scoring and labelling. Scoring techniques assign an anomaly score for each data instance in the dataset. Domain-specific knowledge may be used to choose a threshold; or anomalies may be ranked by score, with the analyst focusing on a few of the top anomalies on the list. [40]

Labelling techniques label data as either nominal or anomalous; they do not allow an analyst to directly choose a threshold, but the selection can be made indirectly by tuning a parameter. In a supervised approach, labels are usually used because they are commonly supported in the available classification

algorithms. In unsupervised and semi-supervised approaches scores are more common, for the practical reasons mentioned previously. [49]

3.4. Evaluation metrics

The area under the ROC curve (ROC-AUC) is a commonly used evaluation method in unsupervised anomaly detection to measure the overall performance of the detector across all possible threshold values [50–52].

The receiver operating charasteristic (ROC) curve (Fig. 1) is obtained by plotting true positive rate (TPR) and true negative rate (TNR) with various threshold values. TPR is on the vertical axis and TNR is on the horizontal axis. TPR and FPR are defined as seen in Equation (1).

$$TPR = \frac{TP}{TP + FN}, FPR = \frac{FP}{FP + TN}, \tag{1}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives. A perfect classifier has TPR of 100 % and FPR of 0 %.

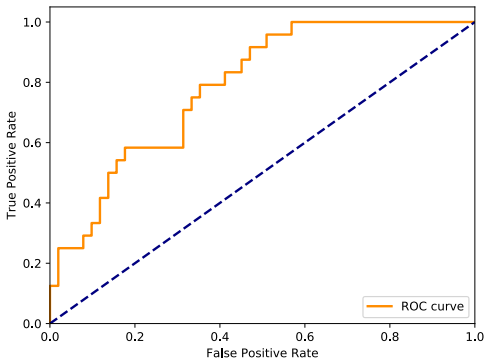


Figure 1. ROC curve

However, an ROC curve is not the best metric when the anomalous and nominal classes are imbalanced [53–55]. In such situation, a precision-recall curve (Fig. 2) is more informative. In such a curve, recall is plotted against precision with various threshold values. Precision is on the vertical axis and recall is on the horizontal axis. Recall is the same as TPR and precision is defined as seen in Equation (2).

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

In the precision-recall space, a perfect classifier has recall and precision of 100 %. [53, 54]

The recall@k metric is a metric commonly used in recommender systems [56]. It is a measure of relevant instances in the top-k with respect to all instances.

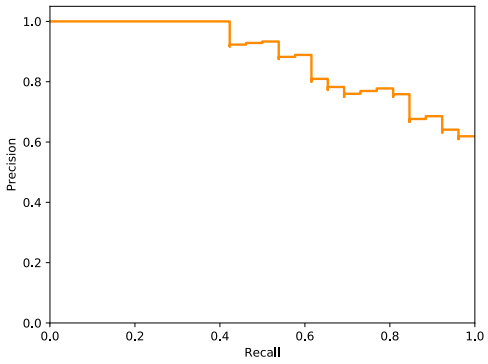


Figure 2. Precision-Recall (RR) curve

For example, if recall@k is 100 %, then all relevant instances are in the top-k. In the context of this thesis, k is the number of instances an analyst is capable of analysing (i.e., the budget) and relevant instances are anomalies that the analyst is interested in detecting.

3.5. Methods

In this section several of the most popular methods for unsupervised point anomaly detection are presented.

3.5.1. Local outlier factor

The local outlier factor (LOF) introduced in [57] is the most well-known density-based local-outlier detection algorithm and was the first to introduce the concept of local outliers [49]. This algorithm assigns an degree of being an outlier to each data instance. The algorithm is local in the sense that it only considers a restricted neighbourhood of a data instance. Local outliers are outliers relative to the densities of their neighbourhood. The local densities are estimated using the k-nearest neighbours for each data instance.

In Fig. 3, one can see the local densities for A, B, C and D. The lines from the point A illustrate the instances used in A's local density estimation. A is clearly an local outlier due to having a lower density than its neighbours.

The LOF requires the construction of a neighbourhood around every data point, involving calculation of pairwise distance between each data point, which is a process of $O(n^2)$ time complexity. Subsampling can be used to reduce complexity of LOF. The Minnesota Intrusion Detection System [42] subsamples the dataset and compares all data points to this smaller set, which reduces time complexity to $O(n * m)$, where n is the size of the data and m is the size of the subsample.

The advantage of the LOF is that it can be used to detect all kinds of outliers, including those that cannot be detected using distance-based algorithms [42].

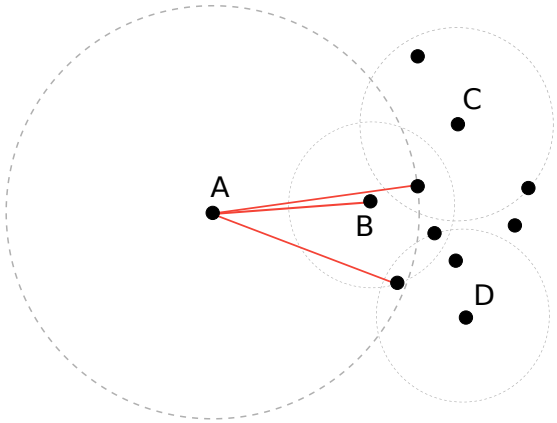


Figure 3. LOF local densities ($k = 3$)

According to the evaluation carried out in [49], LOF-based methods perform poorly with datasets containing global anomalies, generating many false positives; hence these methods should be avoided if the goal is to detect global anomalies.

3.5.2. Clustering

Clustering is a popular technique used to detect anomalies. Clustering-based anomaly detection is usually conducted in unsupervised or semi-supervised way. In the former approach, the model is trained using both nominal and anomalous data; in the latter, it is trained using only nominal data and the trained model is then used as a profile describing nominal behaviour. If the assumption holds that anomalous data instances are a minority and do not form large clusters themselves then the unsupervised approach can be used. [58]

There are many clustering techniques and variations thereof. The most common distance metric used in clustering is Euclidean distance. In this metric, each feature contributes equally to the calculation of the distance. This may not be desirable in many applications; for example, when features have different variabilities or are correlated. This would cause features with higher variability to dominate those with lower variability. An alternative is the distance metric known as Mahalanobis distance. [59]

K-means is one of the most well-known clustering algorithms. It is commonly used because it has a linear time complexity [49]. One such anomaly-based IDS is ADMIT [60], which monitors a user's terminal usage, creates a baseline profile corresponding to nominal usage for the user and verifies future data instances against the profile. K-means partitions the data into k clusters. Instead of k-means, ADMIT used a dynamic clustering approach because they did not want to preset k.

K-means clustering has also been used for anomaly detection in network traffic. Munz et al. [61] used k-mean clustering to separate traffic into nominal and anomalous clusters. They then computed cluster centroids for use in scalable

real-time anomaly detection. They chose $k = 2$ because they assumed that nominal and anomalous traffic form two clusters.

3.5.3. Support-vector machines

The standard SVM was introduced by Boser et al. [62] in 1992. The original SVM is a two-class-based supervised classification technique. One-class SVM (OC-SVM) is a variation of SVM widely used approach in anomaly detection. It was introduced by Schölkopf et al. [63], and the idea of using OC-SVM for anomaly detection was proposed by Schölkopf et al. [64] in 2001.

The OC-SVM is usually used in semi-supervised way. The training data are presumed to belong only to one-class [65]. The Schölkopf OC-SVM creates a hyperplane (Fig. 4a [66]) with a maximum margin from the origin in feature space separating all data points from the origin [63]. The OC-SVM can be used for anomaly detection because it can overcome the over-fitting problem by creating a soft margin using slack variables [67].

When the dataset cannot be separated linearly, kernel functions can be used to map it into higher dimensional space, where it can be separated linearly. This is called the "kernel trick" [68].

The OC-SVM approach by Tax and Duin [69] uses a hypersphere (Fig. 4b [66]) instead of a hyperplane. It aims to minimize the volume of the hypersphere. It uses slack variables to create soft margin similarly to the Schölkopf approach.

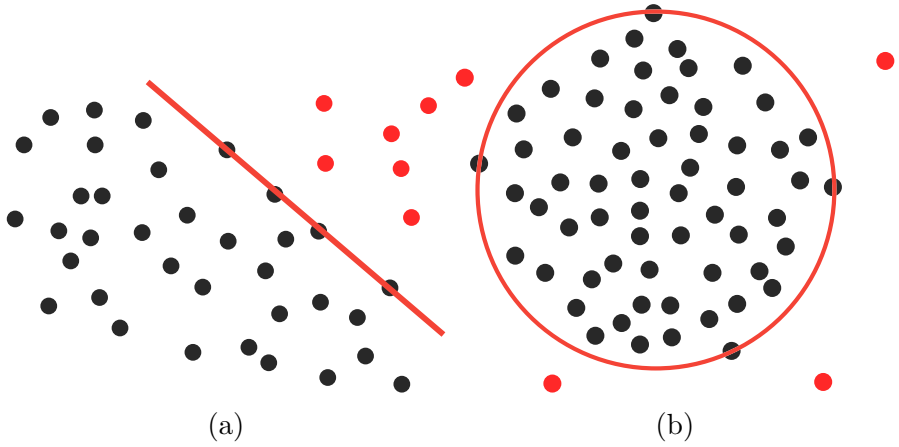


Figure 4. Hyperplane (a) and hypersphere (b) with border support-vectors. Red instances are anomalies. Adopted from Erfani et al. [66]

3.5.4. Ensemble-based methods

The main idea behind ensemble-based methods is to combine multiple detectors to create a detector that outperforms each individual detector in the ensemble [70]. In 2008, Liu et al. [51] proposed IForest, which is an ensemble-based decision tree algorithm for unsupervised anomaly detection. It was later improved in

2012 [71]. IForest is a state-of-the-art unsupervised anomaly detection method. Most model-based approaches to anomaly detection first build a profile of nominal instances and then identify as anomalies any instances that do not conform to this profile. For this reason, these approaches are optimized for profiling nominal instances instead of detecting anomalies. This can introduce false positives (i.e., the detection of nominal instances as anomalous).

IForest isolates anomalies instead of profiling nominal instances (Fig. 5). This method assumes that anomalies are in a minority and that their values are very different from nominal instances. The method builds an ensemble of isolation trees (Fig. 6) for a given dataset and then considers as anomalous those data instances that have short average path lengths. The idea behind this is that outliers are easier to separate from the rest of the data than inliers. An anomaly score is derived from the average path length. [71]

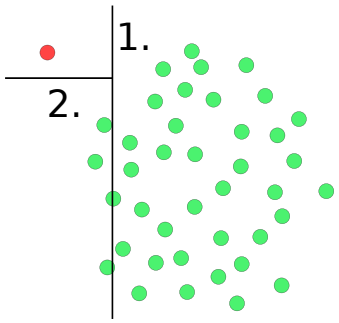


Figure 5. An anomaly isolated with two partitions

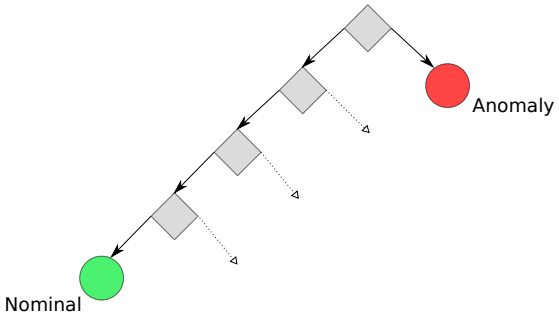


Figure 6. An isolation tree. Shallower leaf nodes have a higher anomaly score than deeper leaf nodes.

IForest builds a model with multiple subsamples to reduce the effects of masking and swamping. Masking occurs when there are too many anomalies, when anomaly clusters become large and dense. Swamping, on the other hand, occurs when nominal data instances are identified as anomalous. Anomaly detection with IForest proceeds in two stages. The initial training is carried out by building isolation trees using subsamples of the training set. In the isolation trees, at each node, a random feature is selected and a random split point for that feature. In

the testing stage, anomaly scores for test instances are computed by passing them through isolation trees. [71]

According to empirical evaluation [71] IForest performs better than LOF in terms of ROC-AUC and has a linear time complexity. IForest performs well for high-dimensional problems and in situations where the training data does not have any anomalies. IForest also has a low constant and a low memory requirement. Because of these factors, it scales well to large datasets with high-dimensional problems and a large number of irrelevant attributes.

Many IForest variations exist. Zhiguo and Minrui [72] extended IForest to work on streaming data using the sliding window technique. Marteau et al. [73] proposed a hybrid IForest that solves limitation in IForest known as "blind spots" and extends it to semi-supervised and supervised learning. The supervised extension enables the incorporation of knowledge about known anomalies. "Blind spots" exists in IForest because the algorithm assumes that anomalies have shorter path lengths thanl nominal data. This is true in normally distributed data; however, it is not true in general. For example, it is not true for data distributed in a concave set, like a torus, as demonstrated by Marteau et al. [49]. In the case of torus, the "blind spot" is inside it and the IForest performs no better in the "blind spot" than a random classifier. The hybrid IForest uses distances to neighbouring nominal and anomalous data as additional sources of information to overcome the "blind spot".

In addition to testing the hybrid IForest on synthetic data, it was also tested on intrusion detection datasets. It was found that it compares favourably to standard IForest and one- and two-class SVM in terms of ROC-AUC.

Das et al. [74] also extended IForest to semi-supervised anomaly detection by incorporating feedback from the analyst. Compared to the standard IForest, their technique performed similarly or better depending on the dataset.

A similar technique to IForest is the robust random cut IForest proposed by Guha et al. [55]. In standard IForest, the dimensions to be partitioned are chosen uniformly at random. The advantage of this is that dimensions are treated independently and it is unaffected by different scaling of dimensions. The disadvantage is that because cuts are chosen uniformly across dimensions, when there are many irrelevant dimensions in the dataset most of the partitions are in the irrelevant dimensions and this causes the algorithm to perform poorly. Robust random cut IForest is also designed to work on streams and it can be dynamically updated by inserting and deleting data points.

3.5.5. Artificial neural network-based methods

Autoencoders are artificial neural network based methods that can be used in semi-supervised and unsupervised anomaly detection. An artificial neural network consists of many computational units called neurons. A neuron takes inputs $x_1, x_2, ,,,, x_N$ and an interception term (bias). The output of the neuron is computed using the formula $f(\sum_{n=1}^{n=N} W_i x_i + b)$, where N is the number of input connections, W_i is the corresponding weights, b is the intercept term and f is the activation function (e.g., sigmoid or tanh). [75]

Layers between the input and output layers are called hidden layers, and units in those layers are called hidden units. An autoencoder with multiple hidden layers is called a deep autoencoder. [75, 76]

An autoencoder applies back-propagation to set the values of the output layer to those of the input layer. By limiting the number of hidden units, the network is forced to learn a compressed representation of the input. If there are many hidden layers units, a sparsity is used on the hidden units instead. This algorithm often results in similar lower-dimensional representation to the principal component analysis. The network of a simple autoencoder with one hidden layer can be seen in Fig. 7. [75]

Autoencoders are mostly used for feature learning and dimensional reduction. The key insight that allows the use of autoencoders for anomaly detection is that anomalies are incompressible and therefore cannot be effectively projected into the lower dimensional representation [77]. In the use of autoencoders for anomaly detection, the anomaly score can be derived from the reconstruction error. When a deep autoencoder trained on nominal data is used to reconstruct an anomalous data instance the reconstruction error is larger than on nominal data [78].

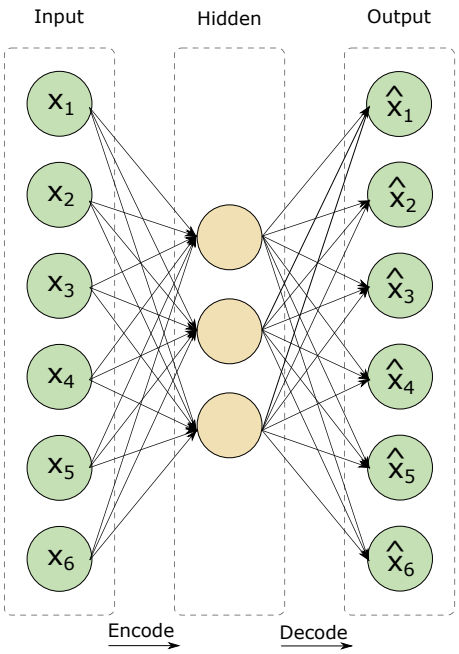


Figure 7. An autoencoder with one hidden layer

Madani and Vlajic [78] used an autoencoder in the context of network intrusion detection under adversarial contamination. The autoencoder was trained on nominal traffic data that had contaminating data (malicious traffic). When the contamination rate was lower than 2 %, contamination did not degrade detection rate much.

Malhotra et al. [79] proposed an LSTM-based autoencoder for anomaly detection on time-series data. The autoencoder was trained on only nominal sequences. Given anomalous sequence, it would not be able to reconstruct it well; hence, the reconstruction error is larger. Their approach worked well on predictable and unpredictable time-series.

3.6. Incorporating expert feedback

Anomaly detection algorithms usually provide scores that are used to rank anomalies, and the most anomalous instances are shown to the analyst. From these, the analyst identifies the most interesting anomalies. Hence, it is preferable that the anomaly system can be configurable by the analyst, so that it can be guided by the analyst's conception of an interesting anomaly, reducing the number of false positives. The analyst investigates these highly scored cases and then labels them as anomaly or nominal.

Using feedback to guide detection is known as active learning. In active learning, the learning algorithm queries an analyst for labels using a chosen query strategy. Labels are used to improve the prediction accuracy. The overall goal is to minimize the number of queries required to reach the target performance. [80]

Several studies have been conducted into how to incorporate feedback into anomaly detection systems [74, 80–83]. Techniques introduced in [80] and [82] exhibit similar performance and are considered to be state-of-the-art. Both of them build upon the same tree-based model proposed by Das et al. [74].

Both of them also use the same "greedy" query-selection strategy. This strategy is efficient and works well in practice [80]. In this context, greedy selection refers to the selection of the most anomalous instances based on the score.

Active anomaly detection [80] is a human-in-the-loop learning framework for discovering anomalies via ensembles of anomaly detectors. The feedback provided by the analyst is used to change the scoring mechanism to rank true anomalies higher. The scoring function combines the scores of the anomaly detectors in the ensemble using a weight vector $\mathbf{w}$. The weight of an ensemble member can be considered to be its relevance to the score.

Weights are adjusted based on the feedback for the goal of maximizing the number of true anomalies shown to the analyst. Optimized weights $\mathbf{w}$ are learned using a gradient descent algorithm. Active anomaly detection makes the assumption that θ fraction of instances are anomalous. The AAD IForest uses leaf node region scores of isolation trees.

3.7. Model interpretability

Interpretable models are required to support the analyst in distinguishing between malicious and non-malicious activity. Studies have confirmed the benefit of explanations for the analyst [84].

Providing descriptions is critical for feedback, because the anomaly detector's future performance depends on the analyst's ability to classify instances correctly.

Doshi-Velez and Kim [85] defined interpretability as the "ability to explain or to present in understandable terms to a human." Das et al. [80] defined explanations as "the most important reason(s) that influenced the algorithm's predicted output" and interpretability as "the representation of predictions in a concise and easy-to-understand manner"

Not all systems have to be interpretable. Explanations are not required if there are no significant consequences or the problem has been well-studied. The need for interpretability arises from an incompleteness in the formalization of the problem. [85]

The more interpretability a model is, the easier it is to comprehend why it made a certain prediction. Hence, there is a need for interpretability when there is value in providing a rationale for a prediction. Moreover, models can only be debugged and audited if they can be interpreted. When something in the model goes wrong the explanation for the erroneous prediction can be used to understand the cause of the error. Additionally, it is easier for users to trust a system that provides good explanations. [80, 86]

There are model-agnostic and model-specific interpretation methods. The commonly used solution to the explainability problem is to use model-specific methods, so called "interpretable" models, such as decision trees and decision rules [87]. These approaches use models in which explanations can be created by inspecting the components of the model, such as a single rule or a path in a decision tree. These work well as long as the model is accurate and the internal components of the model are reasonable restricted [88].

Model-agnostic methods are flexible. The model is treated as a black-box, and the explanations are seperated from the model. Hence, the explanations work with any model. Model-agnostic methods use approaches such as learning an interpretable model from the predictions of the black-box model, by changing inputs and observing how the model reacts or both. [88] A recently introduced model-agnostic method is local interpretable model-agnostic explanations [89].

If the model is very complex obtaining a global understanding of it can be very difficult. Local explanations may be inconsistent with one another because the model may use a feature differently depending on other features. [88]

Model-agnostic methods face some challenges. Rudin [90] argued that model-agnostic methods should not be used in high-stakes decision-making. According to Rudin [90], there is a widespread belief that more complex black box models are more accurate; however, this is often not true, especially when the data are represented in terms of naturally meaningful features. Explanations from model-agnostic methods may be more inaccurate than those from model-specific methods. Inherently interpretable models should be designed instead when applicable. [90]

One way to interpret a model's predictions is to use ruleset-based descriptions. Ruleset-based approaches are popular because they are easy for users to understand. A ruleset model consists of rules, and each rule is a conjunction of conditions. Alternative names for ruleset models include disjunctive normal form and classification rules. [91]

Das et al. [80] introduced a compact descriptions (CD) approach for finding compact subspaces. It can be used to describe discovered anomalies. The

algorithm starts at as many candidate subspaces as there are leaves in the forest. Feedback from the analyst is used in choosing the most relevant candidate subspaces. The algorithm penalizes complex rules; it prefers subspaces that are simple to define and thus easier for the analyst to interpret.

An alternative approach to CD is supervised Bayesian rulesets (BR) [91]. This approach utilizes Bayesian priors to reduce the search space in the search for a globally optimal "maximum a posterior probability" solution for rulesets. The BR used with AAD takes as the training set the queried data instances and a set of randomly labelled instances that are assumed to be nominals [80].

Das et al. [80] discovered that BR has greater precision than CD, whereas CD has higher recall. They also observed that CD usually generates simpler rules.

3.8. Score normalization

When ranking scores from multiple models (or users), scores should generally be on the same scale. One common method to achieve this is standardization (Fig. 8). Standardization is achieved by rescaling the samples to have a mean of zero and unit variance by substracting the mean and dividing by the standard deviation (Eq. 3):

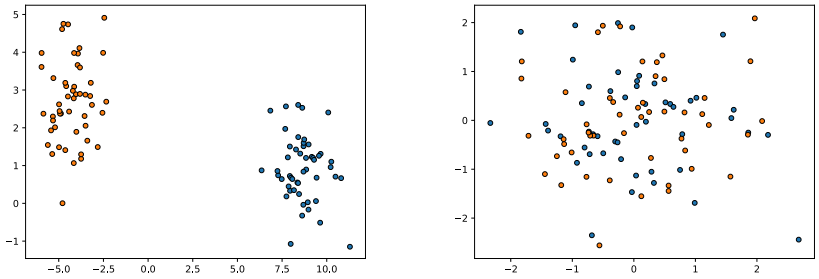
$$z = \frac{x - \mu}{\sigma}, \tag{3}$$

where x is the score to be standardized, μ is the mean, σ is the standard deviation and z is the standardized score (also called z-score).

If the mean and the standard deviation of the distribution is not known, they are estimated from the available data. This method assumes normal distribution; hence, it is not optimal for non-normally distributed scores. This method is not robust because both the mean and standard deviation are sensitive to outliers.

Another method is to use robust normalization [92]. This scales the data using the quantile range (Eq. 4):

$$y = \frac{x - q_1(x)}{q_2(x) - q_1(x)}, \tag{4}$$



(a) Original (b) Standardized

Figure 8. Effect of standardization

where x is the score to be standardized, q_1 and q_2 are quantiles, and y is the normalized score This method requires determining the quantiles q_1 and q_2.

4. DESIGN & IMPLEMENTATION

This section provides a high-level description of the system, the dataset and the used software in the evaluation.

4.1. System description

The proposed system aims to create ranked lists that reduce the workload of the security analyst by ranking suspicious events higher on the list. The chosen feature engineering approach is based on the related research. User-day granularity was chosen because it is the most common granularity in the literature, as seen in the related research section. Most approaches use frequency, statistic and session features, or a variation thereof. Similar features are used in this approach.

Log files from multiple sources are taken as input by the system (Fig. 9). The system then parses and aggregates the log entries into features for each user-day. Feature vectors describing user-days are then processed by the anomaly detector, which outputs an anomaly score for each instance. Optionally, the anomaly scores are normalized. The instances are ranked by their anomaly scores.

In the unsupervised experiment, this list is created for each day and an item in the list represents the activity of the user on that day. In the AAD experiment, this list is created for each user; therefore, on the top of the list are the most anomalous days for that user.

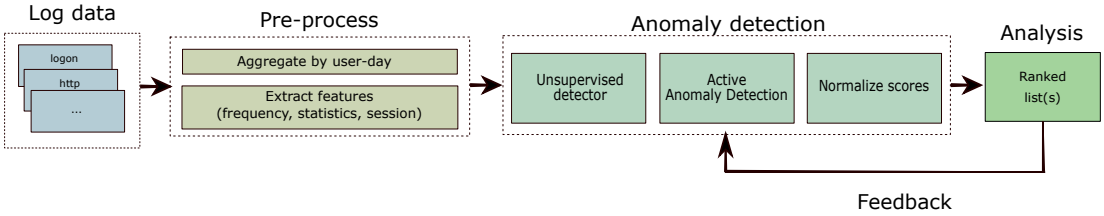


Figure 9. System diagram

4.2. Dataset

The CMU-CERT insider threat dataset [93], that contains logon, HTTP, email, file and removable device log data is used in this work. This dataset has been used in many studies [30, 33, 34, 36].

The dataset consists of synthetic data from a simulated organization (see [94] for how the dataset is generated). There are multiple revisions of the dataset. Version *r4.2* was chosen, because it is a "dense needle" dataset; that is, it contains multiple incidents of each scenario. Newer revisions include more scenarios. From Table 3, one can see the number of records and users in the log files of each data source.

Table 3. No. of records and users in each data source

Source	No. of records	No. of users
Logon	854859	1000
Device	405380	265
Email	2629979	1000
HTTP	28434423	1000
File	445581	264

The system is evaluated by detecting malicious insiders that participate in the following scenarios [93]:

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.*

The total number of user-days in the dataset is 330,452. From Table 4, one can see the number of insiders in each scenario, and how many user-days have insider threat activity. The dataset contains records from the 2^{nd} of January, 2010, to the 17^{th} of May, 2011.

Table 4. No. of malicious insiders and no. of user-days with malicious activity in each scenario.

Scenario	No. of insiders	No. of user-days
1	30	85
2	30	861
3	12	40

The data sources used are the log files from logon, removable device, HTTP, email and file activity. The features constructed from these data sources are shown in Table 5. In Table 6, is the log file format of each data source. "UUID" is a unique event identifier; the "KEYWORDS" field contains keywords such as "keylogger", "malware", "budget" and "shutdown". Keywords are not used in the evaluation.

Table 5. Features constructed for each user-day

Feature	Description
n_dev_conn	Count of removable device events
n_dev_pc	Count of computers with removable device events
min_dev_session	Minimum removable device connection duration
max_dev_session	Maximum removable device connection duration
min_dev	Time of the first removable device event
max_dev	Time of the last removable device event
n_logon	Count of logon events
n_logon_pc	Count of computers with logon events
min_logon_session	Minimum logon session duration
min_logon	Time of the first logon event
max_logon	Time of the last logon event
n_email	Count of sent emails
n_email_pc	Count of computers used to sent emails
n_max_email_recipient	Maximum count of email recipients
n_max_email_attachment	Maximum count of email attachments
n_http_req	Count of HTTP requests
n_http_hostname	Count of host names
n_http_pc	Count of computers used to sent HTTP requests
n_file	Count of transferred files
n_file_pc	Count of computers used to transfer files
min_file_time	Time of the first file event in hours
max_file_time	Time of the last file event in hours

Table 6. Data source log file formats

File	Format
logon.csv	UUID,DATETIME,USER,PC,{Logon, Logoff}
device.csv	UUID,DATETIME,USER,PC,{Connect, Disconnect}
email.csv	UUID,DATETIME,USER,PC,TO,CC,BCC,FROM,SIZE,ATTACHMENTS,KEYWORDS
http.csv	UUID,DATETIME,USER,PC,URL,KEYWORDS
file.csv	UUID,DATETIME,USER,PC,FILENAME,KEYWORDS

4.3. Implementation

Data exploration and initial model development was carried out using Python and Jupyter Notebooks. Python was chosen because it has a developed ecosystem for data science. Jupyter Notebooks allow interactive development and data visualization.

PyOD [95] was used in evaluation of unsupervised models. It is a Python toolkit that provides a uniform application programming interface for various unsupervised anomaly detection techniques. This allows rapid evaluation of different techniques. Scikit-learn [96] was used to compute evaluation metrics. The technique introduced in [80], known as AAD was chosen over [82], because it has implementation available in Python [97]. The BR and CD evaluations used the implementation by Das et al. [97]. The BR was based on the implementation by Wang et al. [91].

5. EVALUATION

In the first set of experiments, the performances of IForest, OC-SVM and LOF were compared in different scenarios and for different data sources. All detectors use default hyperparameters. IForest had 100 trees in the ensemble and 256 samples were used to train each tree. LOF used the 20 nearest neighbours; and OC-SVM used the radial basis function kernel, with a kernel coefficient of $1/n_{features}$. Scores were computed by running 10 iterations and taking the average. Quantile range for the robust score normalization was determined empirically ($q_1 = 0.95$, $q_2 = 0.05$).

By adjusting hyperparameters, it is possible to a obtain higher performance. However, this can be difficult in the real world due to the lack of labelled data, and because of added complexity tuning was not performed in this study. When evaluating scenario-based performance other scenarios were filtered out from the data.

5.0.1. Unsupervised detector performance

In this experiment, the objective was to evaluate the system by determining how much the detectors decrease the number of users required to investigate per day to catch malicious insiders. This was implemented by assigning a daily budget for the analysis; that is, the number of users that can be investigated per day. This is encapsulated by the recall@k metric, where k is the daily budget.

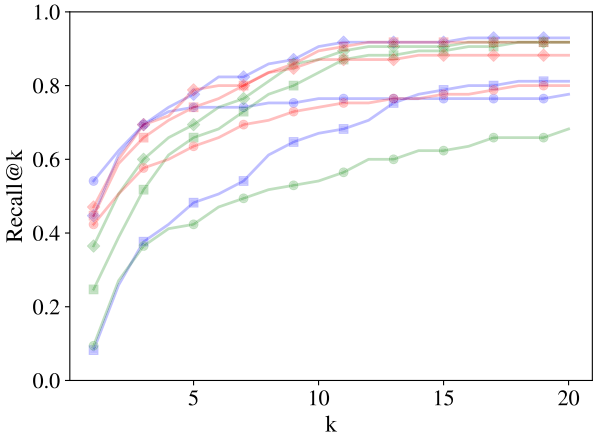
In scenario 1, insider activity occurred after hours, and so the logon-only model (Fig. 10a) resulted in high recall. However, the highest recall was obtained by combining the logon and device features (Fig. 10b). Adding irrelevant data sources had a negative impact on recall (Fig. 10c). In most cases, robust score normalization resulted in higher recall than standardization.

The insiders in scenarios 2 and 3 were more difficult to detect, and so in Figs. 11 and 12 daily budget is shown up to 800 users.

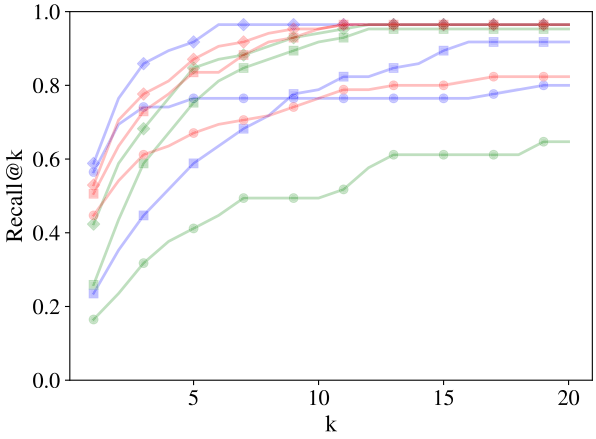
In scenario 2, the IForest model that used all data sources obtained the highest recall (Fig. 11c). When using only logon and device data sources (Fig. 11a), the recall of the unnormalized IForest was poor; IForest with standardization was significantly better. Furthermore, LOF resulted in the highest recall at budget of 200 users; however, it does not improve at larger daily budget.

Similarly, in scenario 3, IForest obtained the highest recall overall. The model that used logon and device sources resulted in the highest recall at smaller daily budget (Fig. 12b). In the model with all data sources (Fig. 12c) at lower budget the recall is worse, but it obtains higher recall at higher budget than in the Fig. 12b. In scenario 2 (Fig. 11c) and 3 (Fig. 12c), the highest recall was obtained with the models that used data sources.

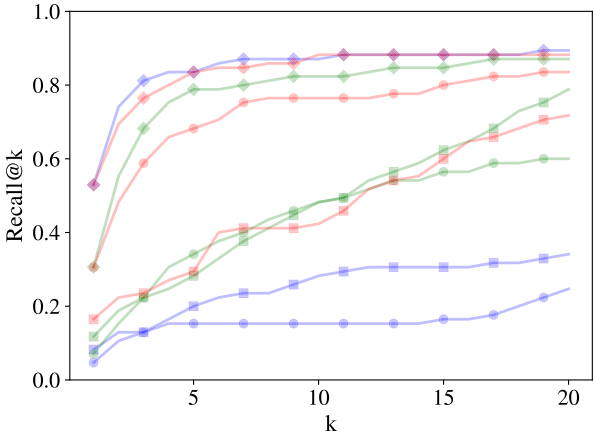
Robust score normalization tended to perform better at smaller daily budget, whereas z-score normalization performed better at larger budget (Figs. 10, 11 and 12). Normalization does not have a positive impact on IForest overall, although there are some exceptions, such as that shown in Fig. 11a, where z-score IForest



(a) With logon features



(b) With logon and device features



(c) With logon, device, email, http and file features

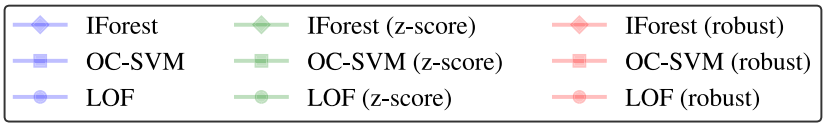
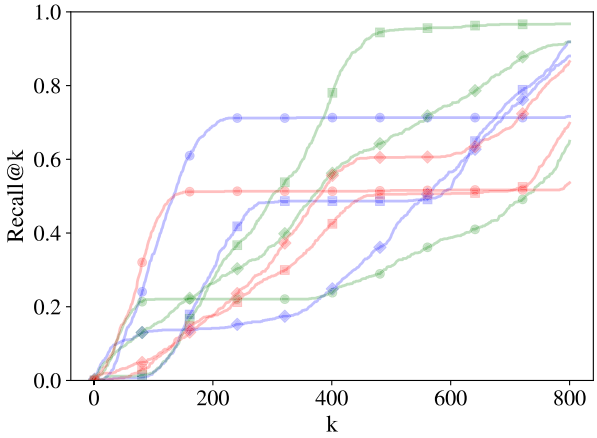


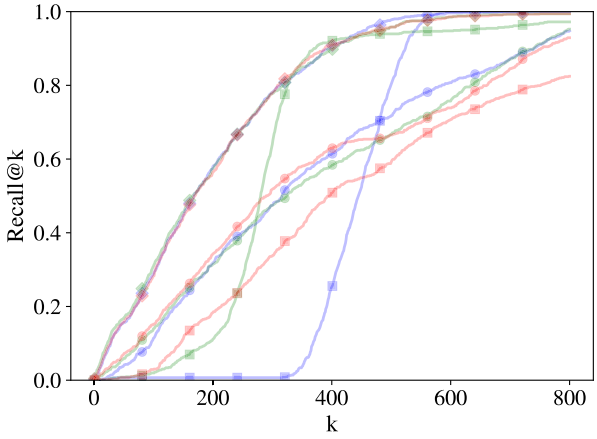
Figure 10. Recall@k with respect to k. Shows that in scenario 1, the unnormalized IForest detector resulted in the highest recall, and the highest recall was obtained using logon and device features (b).

had the highest recall of IForest detectors. In the case of LOF and OC-SVM, normalization had a significant impact on recall (Figs. 10, 11 and 12).

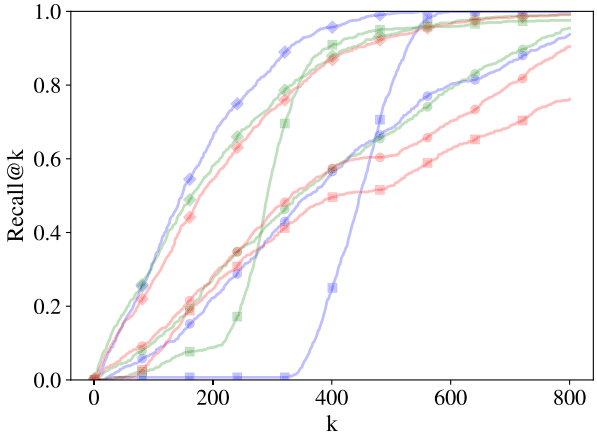
From the Table 7, one can see the recall of detectors with different data sources combinations more precisely. From the Table 7a, one can conclude that the highest recall in scenario 1 at daily budget of 10 users was 96 %. In other words, the detector with the highest recall missed 3 user-days. In scenario 2 (Table 7b), the highest recall at daily budget of 400 users was 96 %; that is to say, the detector missed 37 user-days. In scenario 3 (Table 7c) when using all data sources at daily budget of 400 users, the IForest detected all user-days.



(a) With logon and device features



(b) With logon, device, email and http features



(c) With logon, device, email, http and file features

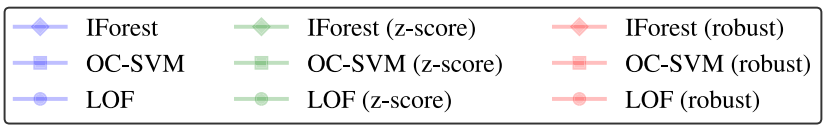
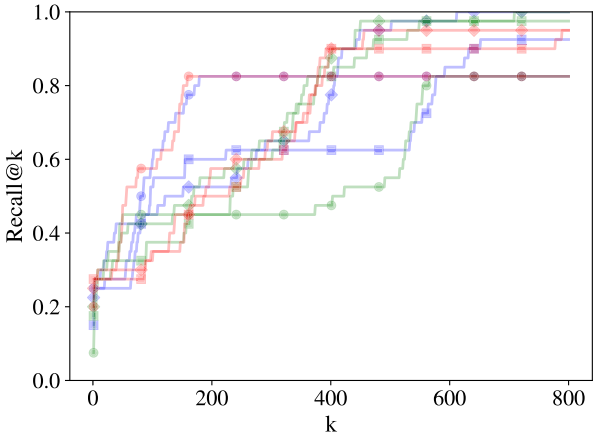
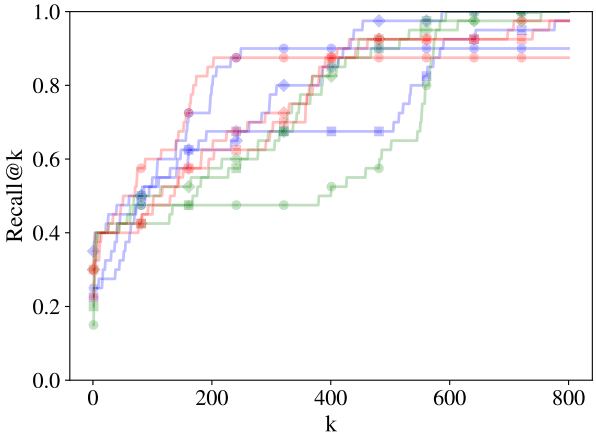


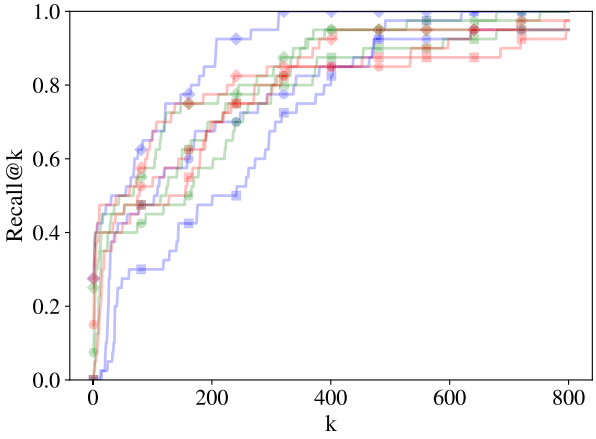
Figure 11. Recall@k with respect to k. Shows that in scenario 2, the unnormalized IForest detector resulted in the highest recall, and the highest recall was obtained using all features (c).



(a) With logon features



(b) With logon and device features



(c) With logon, device, email, http and file features

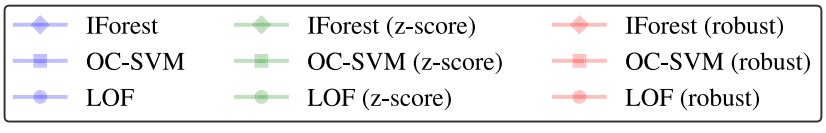


Figure 12. Recall@k with respect to k. Shows that in scenario 3, the unnormalized IForest detector resulted in the highest recall, and the highest recall was obtained using all features (c).

Table 7. Recall at daily budget and number of detected user-days in scenario 1 (a), 2 (b) and 3 (c). The highest performance detector for each data source combination is shown in bold.

(a) Recall@10 in scenario 1. The total number of scenario user-days is 85.

Data sources	Detector	None	z-score	Robust
Logon	IForest	**0.91 (77)**	0.87 (74)	0.87 (74)
	OC-SVM	0.67 (57)	0.84 (71)	0.89 (76)
	LOF	0.76 (65)	0.54 (46)	0.74 (63)
Logon and device	IForest	**0.96 (82)**	0.94 (80)	0.95 (81)
	OC-SVM	0.79 (67)	0.92 (78)	0.95 (81)
	LOF	0.76 (65)	0.49 (42)	0.76 (65)
All	IForest	0.87 (74)	0.82 (70)	**0.88 (75)**
	OC-SVM	0.28 (24)	0.48 (41)	0.42 (36)
	LOF	0.15 (13)	0.48 (41)	0.76 (65)

(b) Recall@400 in scenario 2. The total number of scenario user-days is 861.

Data sources	Detector	None	z-score	Robust
Logon and device	IForest	0.25 (213)	0.56 (484)	0.55 (475)
	OC-SVM	0.49 (419)	**0.77 (667)**	0.43 (366)
	LOF	0.71 (614)	0.24 (204)	0.51 (442)
Logon, device, email and HTTP	IForest	0.91 (781)	0.90 (773)	0.91 (783)
	OC-SVM	0.25 (219)	**0.92 (792)**	0.51 (437)
	LOF	0.61 (528)	0.58 (502)	0.63 (542)
All	IForest	**0.96 (824)**	0.88 (757)	0.87 (746)
	OC-SVM	0.24 (209)	0.91 (781)	0.50 (427)
	LOF	0.56 (485)	0.57 (492)	0.57 (493)

(c) Recall@400 in scenario 3. The total number of scenario user-days is 40.

Data sources	Detector	None	z-score	Robust
Logon	IForest	0.78 (31)	0.88 (35)	**0.90 (36)**
	OC-SVM	0.62 (25)	0.82 (33)	**0.90 (36)**
	LOF	0.82 (33)	0.47 (19)	0.82 (33)
Logon and device	IForest	0.85 (34)	0.82 (33)	0.88 (35)
	OC-SVM	0.68 (27)	0.85 (34)	0.88 (35)
	LOF	**0.90 (36)**	0.50 (20)	0.88 (35)
All	IForest	**1.00 (40)**	0.95 (38)	0.93 (37)
	OC-SVM	0.80 (32)	0.88 (35)	0.85 (34)
	LOF	0.85 (34)	0.95 (38)	0.85 (34)

5.0.2. Active anomaly detection

Active anomaly detection was evaluated against the unsupervised IForest. In the experiments, "region-based" refers to the IForest AAD implementation by Das et al. [80], which uses the leaf regions of isolation trees. The "tree-based" implementation used isolation tree scores. Unsupervised models are the starting conditions for the AAD (i.e., zero labelled instances). The "unsupervised tree-based" model has the same detection performance as the standard IForest.

This implementation was evaluated because, in some conditions, the region-based detector has lower recall than the standard IForest. Plots are shown with 95 % confidence intervals.

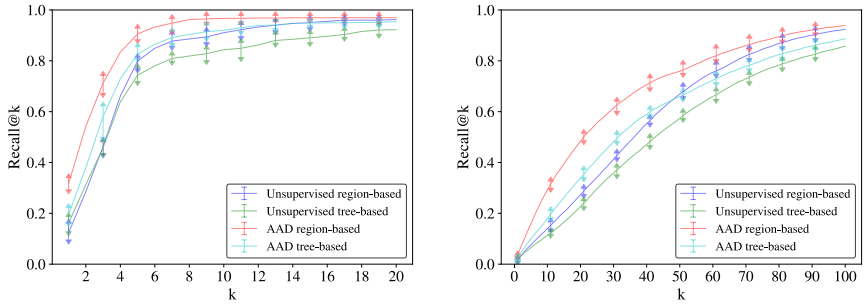
Ranked lists were generated for each user and k is the top-k (the budget) days in the ranked list of each user. They were constructed this way because the focus of this experiment was solely on evaluating AAD. However, because of this choice, comparing the recall with those of previous experiments is misleading.

From Fig. 13, one can see that AAD improved the recall in every scenario. The most significant improvement was in scenario 3 (Fig. 13c), where the improvement was approximately from 60 % recall@20 to 80 % recall@20 on average. Without feedback similar recall is obtained at budget of 50 days.

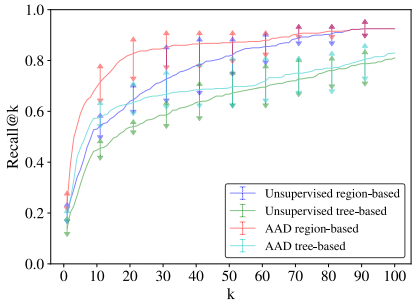
Unexpectedly, even the "unsupervised region-based" detector obtained higher recall generally than the standard IForest. This improvement was probably due to the difference in how the anomaly scores were constructed from the IForest structure. In some cases, the tree-based AAD detector resulted in significantly higher recall at larger budget (Fig. 14).

From Fig. 15, one can see that the number of labelled data instances required to obtain the highest recall was different in all scenarios. This experiment was performed at the budget of 10 days. The most significant improvement was in scenario 3 (Fig. 15c). From this experiment, one can conclude that the region-based detector uses labels more effectively because it has steeper recall increase with respect to the number of labelled instances (Fig. 15).

As can be seen from the Table 8, in scenario 1 AAD detected 7 user-days more than the respective unsupervised detectors at budget of 5 days, and in scenario 3 the region-based AAD detected 8 user-days more at budget of 20 days. The highest improvement in recall is in scenario 2, where the region-based AAD detected 431 user-days, which is 184 user-days more than the unsupervised detector detected at budget of 20 days.



(a) Scenario 1 (b) Scenario 2



(c) Scenario 3

Figure 13. Recall@k with respect to k in every scenario using all features. Region-based AAD had the highest recall in every scenario.

Table 8. Shows recall and number of detected user-days at budget of 5 days in scenario 1, and in scenarios 2 and 3 at budget of 20 days. The budget is the top-k most anomalous days for each user. The highest result in terms of recall is shown in bold for each scenario.

Detector	Scenario 1	Scenario 2	Scenario 3
Unsupervised region-based	0.85 (72)	0.29 (247)	0.65 (25)
Unsupervised tree-based	0.78 (66)	0.24 (206)	0.54 (21)
AAD region-based	**0.93 (79)**	**0.50 (431)**	**0.83 (33)**
AAD tree-based	0.86 (73)	0.35 (304)	0.64 (25)

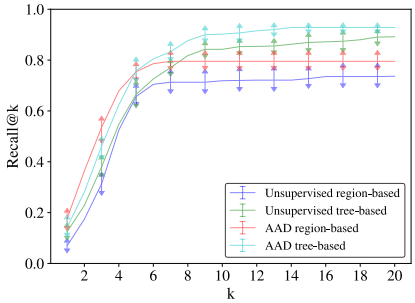
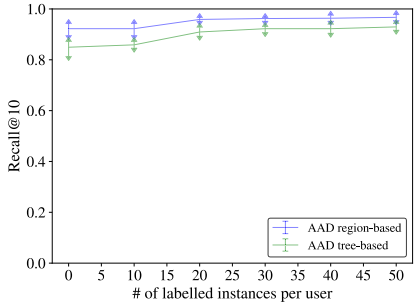
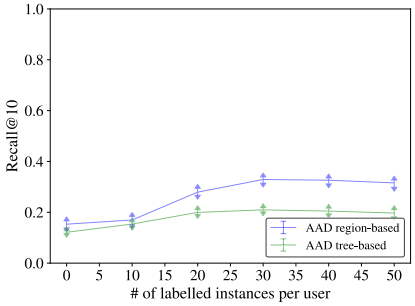


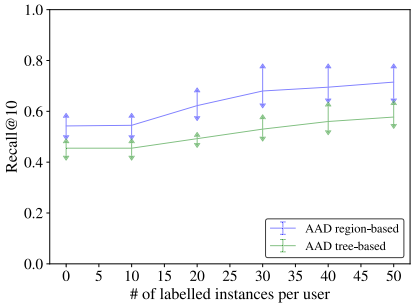
Figure 14. Tree-based detectors obtained higher recall than region-based at larger budgets when using only logon and file features in scenario 1.



(a) Scenario 1



(b) Scenario 2



(c) Scenario 3

Figure 15. Recall@10 with respect to the number of labelled anomalies in every scenario and up to 50 labelled instances per user. From the subfigures, one can conclude that the region-based AAD benefits more from the feedback.

5.0.3. Descriptions

The objective of this experiment was to evaluate how well CD and BR can generate rules that distinguish anomalies from nominals; that is, how well the rules describe anomalous activity. The rules were computed for known anomalies.

Precision and recall metrics were used. These metrics were computed for rules by using the rules to classify instances. If one of the rules encompasses an instance, it was classified as an anomaly. Weights $\mathbf{w}$ computed by the AAD are used to score regions in the computation of descriptions; therefore, experiments were evaluated with respect to the number of labelled instances. The reported metrics are averages of 10 runs with 95 % confidence intervals.

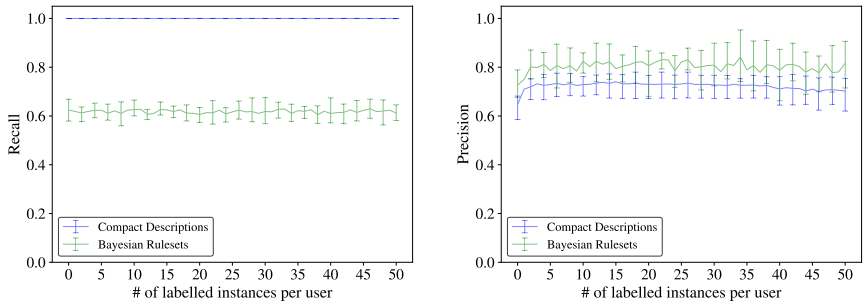
Table 9 shows the computed rules for the user *AJR0932*. In this case, the day 10^{th} of September was deemed an anomaly because of the user logged in on after-hours at 11 PM; the 17^{th} and 18^{th} were deemed anomalous because the duration of the removable device connection was over 18 minutes. See Appendix 1 for more comprehensive list of what kind of rules were generated and for their explanations.

Fig. 16, shows the results of the evaluation of the rules with respect to the number of labelled anomalies using AAD. Occasionally description methods may not be able to generate a description for an instance. The results show that the recall of CD was 100 %; the recall of BR was lower. From this, one can conclude that CD generated rule encompass nearly always the instance that the rule was generated for. This is not the case for BR; however, BR has higher precision and so the number of false positives is lower. From Fig. 17, one can see that on average rules have one to three conditions, depending on the scenario. Compact descriptions generates shorter rules in all scenarios. This may also explain partly why BR has higher precision.

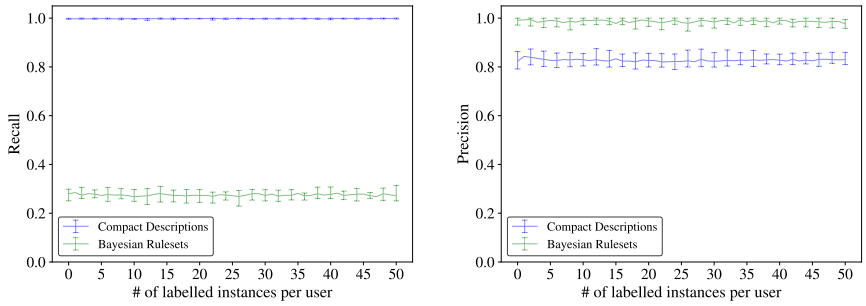
The number of labels used by the AAD did not have a significant impact on generation of descriptions (Fig. 17). The shortest descriptions were generated for scenario 1 (Fig. 17a) because the actors can be detected using the *last_logon* and *max_dev_session* features, as seen in the Table 9. The scenario 3 (Fig. 17c) is the most difficult scenario to generate descriptions for. This explains the large confidence interval.

Table 9. Example decision rules for the user *AJR0932* (scenario 1). Salient features are shown in bold. Generated using CD.

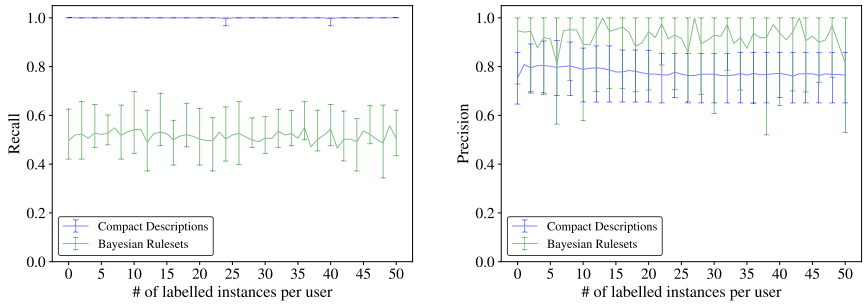
Satisfied rule	**last_logon > 22.45**	**max_dev_session > 18**	
Feature	2010-09-10	2010-09-17	2010-09-18
n_logon	2	3	0
n_logon_pc	1	1	1
min_logon_session	9	1	2
max_logon_session	9	8	2
first_logon	7	4	0
last_logon	**23**	18	0
n_dev_conn	0	9	1
n_dev_pc	0	1	1
min_dev_session	0	0	36
max_dev_session	0	**21**	**36**
first_dev	0	4	0
last_dev	0	20	0



(a) Scenario 1

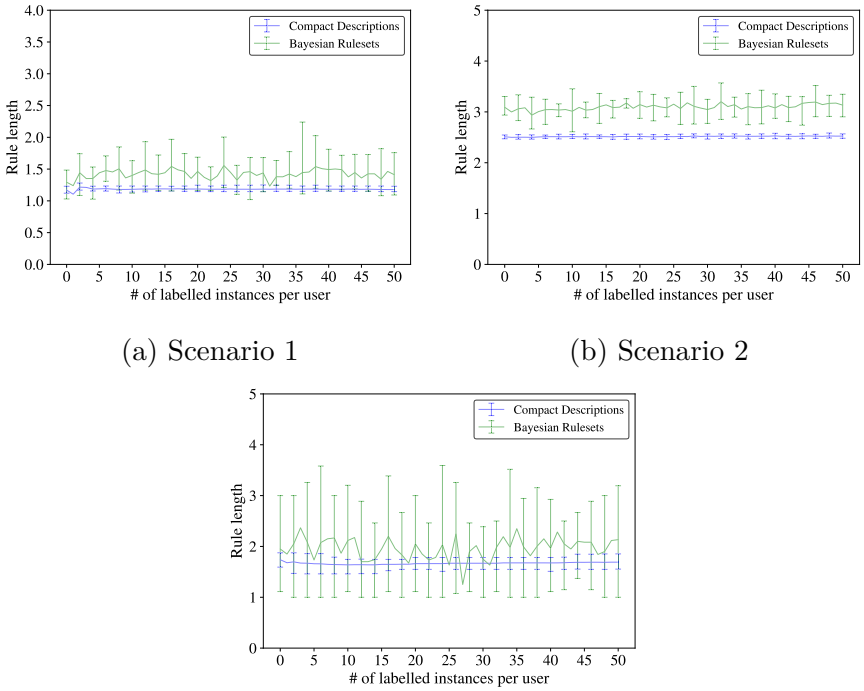


(b) Scenario 2



(c) Scenario 3

Figure 16. Recall and precision of rules with respect to the number of labelled anomalies using all features. From the subfigures, one can see that CD had higher recall and BR had higher precision.



(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

Figure 17. Rule lengths of CD and BR. CD had shorter rule lengths on average. Shorter rules have fewer conditions, and are therefore simpler.

6. DISCUSSION

From the unsupervised anomaly detector experiment, one can conclude that, generally IForest achieved the highest performance in terms of recall@k, where k is the analysis budget (i.e., the number of instances that are investigated), and it did not require normalization. LOF performed well with some data sources and poorly with others. This may affect its applicability in practice. This likely happens because LOF is sensitive to the choice of the nearest neighbours parameter. Combining multiple detectors using ensemble combination methods may mitigate this. Similarly, the performance of OC-SVM varied markedly across different source combinations.

IForest had the most consistent performance across different feature (or data source) combinations; it generally achieved the best performance. This implies that IForest is the most applicable unsupervised detector in practice of the detectors evaluated in this work.

Active anomaly detection was used with the IForest detector. It was found to improve recall@k, especially at smaller budget sizes; thus, it reduces the workload of the security analyst because they can discover more threats while investigating fewer data. Region-based AAD is more applicable than tree-based because the region-based had generally higher performance and improved more from the feedback.

The rules created using CD and BR largely filtered out well irrelevant features; however, the rules occasionally encompassed nominal behaviour or perhaps overfitted the data. This reduces their usability in practice because the analyst cannot completely trust the descriptions. Usually, CD had higher precision and lower recall than BR. This finding is consistent with the results of the experiments Das et al. conducted [80]. Generally, CD created simpler rules with fewer conditions.

Objective comparison to related studies is difficult for reasons such as the use of different datasets (or dataset versions) and evaluation metrics, and differences in how features are constructed and how the dataset is used in the evaluation. From the detection point of view, deep learning methods such as the unsupervised one [34] or the supervised one used in [33] that also utilized temporal aggregation and keywords likely achieve higher detection performance. Liu et al. [34] compared their unsupervised deep learning approaches to standard IForest, and the deep learning approaches performed favourably at lower daily budget.

Interpreting complex models, such as deep neural networks is challenging [88, 98]. Although deep neural networks can reduce the amount of feature engineering required [30, 34].

The system described in this work used only usage count, session and time-based features; it did not utilize all the information available in the dataset. To provide more context to the system (such as from organizational structure) research could be conducted on the usage of graph techniques or on the encoding of this information into features, as was performed in [34]. Another disadvantage of the system described here is the lack of temporal aggregation of activity across days, because, indicators of the insider activity could be spread across multiple days, as is the case in scenario 2. This has been implemented in many studies

[33,37]. Perhaps temporal aggregation across days without impacting granularity could be implemented into the IForest detector by concatenating feature vectors (known as "shingling") from multiple days.

The single feature scores used by [23], [29], [27] aims to highlight features that are the most useful (feature importance) in detecting anomalies. Additionally to extracting relevant features, decision-rule-based descriptions contain information about the feature value ranges. Also, the number of instances encompassed by the rule can be used in determining how anomalous the instance is or how correct the rule is; a rule generated for a nominal instance likely encompasses many instances, whereas a rule generated for an anomaly encompasses fewer instances. This may further assist the analyst in determining if the activity poses a threat.

Goldberg et al. [27] labelled each user-day with human-friendly description for each of 12 types of concerning user activities on a training dataset and then created transformation from these features to labels. Similar approach could be applied to create more human-friendly descriptions for known concerning activities. Ideally, the description would also include terms that describe the threat (such as "data exfiltration", "sabotage" or "espionage"). This requires information about the threats and the use of signatures or supervised learning may become a more feasible approach.

The description techniques could be used with other approaches to improve interpretability further. For example, visualising relationships from the model features back to log entries may be beneficial.

Studies have found that user behaviour may change during different periods, such as morning, midday, evening and night [36,37]. Constructing features based on these observations may be worthwhile. Also, weekends were not filtered out from the data used in this work, which may have a negative effect on the results because the user behaviour may change during weekends.

In the AAD experiment, ranked lists were created for each user instead of for every day. AAD IForest may require normalization when used in the construction of a single ranked list from multiple users, and this may have an impact on the performance of the AAD. Additionally, the use of thresholds in producing alerts could be valuable, so the analyst can be alerted automatically when suspicious activity was observed. The aforementioned issues could be studied in future research.

The next step would be to evaluate how useful AAD and descriptions are in a real corporate environment. Possibly simulated insider scenarios could be inserted into a dataset collected from a corporate entitys's environment. The approach could be extended to role- or device-based activity monitoring [31] because in some insider scenarios, role- or device-based activities may be better indicators of insider threat activity than comparing the user's current activity to the user's past activity. Also, evaluation of the IForest variants discussed in the anomaly detection section would be interesting.

7. CONCLUSIONS

The objective of this thesis is the evaluation of the recently developed active anomaly detection and description methods in the context of insider threat detection. Active anomaly detection, or more generally known as active learning is important because purely unsupervised techniques tend to detect all abnormal events, and this may not completely align with what the security analyst is interested in. Additionally, description methods can provide value because besides detecting anomalies, the analyst should be able to reason why a particular instance is an anomaly; the purpose of descriptions is to aid the analyst in this process. Active anomaly detection and rule-based descriptions operate well together, because accurate and interpretable descriptions allow an analyst to label more effectively if an instance is true or false positive, and this label can be provided as feedback to the active learning algorithm.

Active anomaly detection was evaluated for isolation forest because it was found to have the highest detection performance of the evaluated unsupervised detectors. It was found to improve detection by ranking anomalies higher, allowing analysts to identify more true positives while examining less data.

Both description methods could be used to analyse why a particular instance was assigned a high anomaly score. However, the created rules cannot completely distinguish insider threats from nominal activity. Furthermore, how interpretable these rules are depends on the input features of the detector. Now that the evaluation is done on synthetic data, the next step is to evaluate how practical the evaluated active anomaly detection and description techniques are in a real-world environment.

8. REFERENCES

[1] Theis M., Trzeciak R., Costa D., Moore A., Miller S., Cassidy T. & Claycomb W. (2019) Common sense guide to mitigating insider threats. Tech. Rep. CMU/SEI-2018-TR-010, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=540644.

[2] PWC (2018), Global economic crime and fraud survey. URL: https://www.pwc.com/us/en/services/forensics/library/global-economic-fraud-survey.html, [Online; accessed 23-Aug-2019].

[3] Kellett A. (2015) Trends and Future Directions in Data Security, Insider Threat Report. Tech. rep., Vormetric Data Security.

[4] (2019) Data Breach Investigations Report. Tech. rep., Team, Verizon RISK.

[5] Brackney R.C. & Anderson R.H. (2004) Understanding the insider threat. proceedings of a march 2004 workshop. Tech. rep., RAND CORP SANTA MONICA CA.

[6] Pfleeger S.L., Predd J.B., Hunker J. & Bulford C. (2009) Insiders behaving badly: Addressing bad actors and their actions. IEEE Transactions on Information Forensics and Security 5, pp. 169–179.

[7] Spitzner L. (2003) Honeypots: Catching the insider threat. In: 19th Annual Computer Security Applications Conference, 2003. Proceedings., IEEE, pp. 170–179.

[8] Greitzer F.L. & Frincke D.A. (2010) Combining Traditional Cyber Security Audit Data with Psychosocial Data: Towards Predictive Modeling for Insider Threat Mitigation, Springer US, Boston, MA. pp. 85–113. URL: https://doi.org/10.1007/978-1-4419-7133-3_5.

[9] Team C.I.T. (2013) Unintentional insider threats: A foundational study. Tech. Rep. CMU/SEI-2013-TN-022, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA. URL: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=58744.

[10] Hunker J. & Probst C.W. (2011) Insiders and insider threats-an overview of definitions and mitigation techniques. JoWUA 2, pp. 4–27.

[11] Lane A. (2010), Understanding and selecting siem/lm: Use cases, part 1. URL: http://securosis.com/blog/understanding-and-selecting-siem-lm-use-cases-part-1, [Online; accessed 26-Aug-2019].

[12] Canner B. (2018), 7 key siem capabilities to look for in your solution. URL: hhttps://solutionsreview.com/security-information-event-management/7-key-siem-capabilities-look-solution/, [Online; accessed 26-Aug-2019].

[13] User behavior analytics (uba) explained. URL: https://www.rapid7.com/fundamentals/user-behavior-analytics/, [Online; accessed 10-September-2019].

[14] Shashanka M., Shen M.Y. & Wang J. (2016) User and entity behavior analytics for enterprise security. In: 2016 IEEE International Conference on Big Data (Big Data), IEEE, pp. 1867–1874.

[15] Satria M., Ngadi M. & Abdullah H. (2008) A survey on manet intrusion detection. International Journal of Computer Science and Security 2.

[16] Kemmerer R. & Vigna G. (2002) Intrusion detection: a brief history and overview. Computer 35, pp. supl27–supl30. URL: https://doi.org/10.1109/mc.2002.1012428.

[17] Fernandes G., Rodrigues J.J.P.C., Carvalho L.F., Al-Muhtadi J.F. & Proença M.L. (2018) A comprehensive survey on network anomaly detection. Telecommunication Systems 70, pp. 447–489. URL: https://doi.org/10.1007/s11235-018-0475-8.

[18] Denning D. (1987) An intrusion-detection model. IEEE Transactions on Software Engineering SE-13, pp. 222–232. URL: https://doi.org/10.1109/tse.1987.232894.

[19] Scarfone K. & Mell P. (2007) NIST Special Publication 800-94, Guide to Intrusion Detection and Prevention Systems (IDPS). 2-5 p.

[20] Axelsson S. (2000) Research in intrusion-detection systems: A survey .

[21] Patcha A. & Park J.M. (2007) An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer Networks 51, pp. 3448–3470. URL: https://doi.org/10.1016/j.comnet.2007.02.001.

[22] Eldardiry H., Bart E., Liu J., Hanley J., Price B. & Brdiczka O. (2013) Multi-domain information fusion for insider threat detection. In: 2013 IEEE Security and Privacy Workshops, IEEE, pp. 45–51.

[23] Ted E., Goldberg H.G., Memory A., Young W.T., Rees B., Pierce R., Huang D., Reardon M., Bader D.A., Chow E. et al. (2013) 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, ACM, pp. 1393–1401.

[24] Gavai G., Sricharan K., Gunning D., Hanley J., Singhal M. & Rolleston R. (2015) Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data. JoWUA 6, pp. 47–63.

[25] Legg P.A., Buckley O., Goldsmith M. & Creese S. (2015) Caught in the act of an insider attack: detection and assessment of insider threat. In: 2015 IEEE International Symposium on Technologies for Homeland Security (HST), IEEE, pp. 1–6.

[26] Sun L., Versteeg S., Boztas S. & Rao A. (2016) Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study. CoRR abs/1609.06676. URL: `http://arxiv.org/abs/1609.06676`.

[27] Goldberg H.G., Young W.T., Memory A. & Senator T.E. (2016) Explaining and aggregating anomalies to detect insider threats. In: 2016 49th Hawaii International Conference on System Sciences (HICSS), IEEE, pp. 2739–2748.

[28] Goldberg H., Young W., Reardon M., Phillips B. et al. (2017) Insider threat detection in prodigal .

[29] Tuor A., Kaplan S., Hutchinson B., Nichols N. & Robinson S. (2017) Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In: Workshops at the Thirty-First AAAI Conference on Artificial Intelligence.

[30] Yuan F., Cao Y., Shang Y., Liu Y., Tan J. & Fang B. (2018) Insider threat detection with deep neural network. In: Y. Shi, H. Fu, Y. Tian, V.V. Krzhizhanovskaya, M.H. Lees, J. Dongarra & P.M.A. Sloot (eds.) Computational Science – ICCS 2018, Springer International Publishing, Cham, pp. 43–54.

[31] Lv Q., Wang Y., Wang L. & Wang D. (2018) 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), IEEE, pp. 6–10.

[32] Gamachchi A., Sun L. & Boztas S. (2018) A graph based framework for malicious insider threat detection. arXiv preprint arXiv:1809.00141 .

[33] Chattopadhyay P., Wang L. & Tan Y.P. (2018) Scenario-based insider threat detection from cyber activities. IEEE Transactions on Computational Social Systems 5, pp. 660–675.

[34] Liu L., De Vel O., Chen C., Zhang J. & Xiang Y. (2018) Anomaly-based insider threat detection using deep autoencoders. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, pp. 39–48.

[35] Le D.C. & Zincir-Heywood A.N. (2019) Machine learning based insider threat modelling and detection. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, pp. 1–6.

[36] Lu J. & Wong R.K. (2019) Insider threat detection with long short-term memory. In: Proceedings of the Australasian Computer Science Week Multiconference, ACM, p. 1.

[37] Meng F., Lou F., Fu Y. & Tian Z. (2018) Deep learning based attribute classification insider threat detection for data security. In: 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), IEEE, pp. 576–581.

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

[39] Siddiqui M.A., Stokes J.W., Seifert C., Argyle E., McCann R., Neil J. & Carroll J. (2019) Detecting cyber attacks using anomaly detection with explanations and expert feedback. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp. 2872–2876.

[40] Chandola V., Banerjee A. & Kumar V. (2009) Anomaly detection: A survey. ACM Computing Surveys 41, pp. 1–58. URL: https://doi.org/10.1145/1541880.1541882.

[41] Hawkins D.M. (1980) Identification of outliers, vol. 11. Springer.

[42] Ertöz L., Eilertson E., Lazarevic A., Tan P.N., Kumar V., Srivastava J. & Dokas P. Chapter 3 the minds-minnesota intrusion detection system.

[43] Ghosh S. & Reilly D.L. (1994) Credit card fraud detection with a neural-network. In: System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on, vol. 3, IEEE, vol. 3, pp. 621–630.

[44] Feather F., Siewiorek D. & Maxion R. (1993) Fault detection in an ethernet network using anomaly signature matching. In: ACM SIGCOMM Computer Communication Review, vol. 23, ACM, vol. 23, pp. 279–288.

[45] Gama J., Žliobaitė I., Bifet A., Pechenizkiy M. & Bouchachia A. (2014) A survey on concept drift adaptation. ACM computing surveys (CSUR) 46, p. 44.

[46] Das S., Wong W.K., Fern A., Dietterich T.G. & Siddiqui M.A. (2017) Incorporating feedback into tree-based anomaly detection. arXiv preprint arXiv:1708.09441 .

[47] Salehi M. & Rashidi L. (2018) A survey on anomaly detection in evolving data. ACM SIGKDD Explorations Newsletter 20, pp. 13–23. URL: https://doi.org/10.1145/3229329.3229332.

[48] Gogoi P., Bhattacharyya D.K., Borah B. & Kalita J. (2011) A survey of outlier detection methods in network anomaly identification. Comput. J. 54, pp. 570–588.

[49] Goldstein M. & Uchida S. (2016) A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. PloS one 11, p. e0152173. URL: https://dx.doi.org/10.1371%2Fjournal.pone.0152173.

[50] Campos G.O., Zimek A., Sander J., Campello R.J.G.B., Micenková B., Schubert E., Assent I. & Houle M.E. (2016) On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery 30, pp. 891–927. URL: https://doi.org/10.1007/s10618-015-0444-8.

[51] Liu F.T., Ting K.M. & Zhou Z.H. (2008) Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, IEEE, pp. 413–422.

[52] Tan S.C., Ting K.M. & Liu T.F. (2011) Fast anomaly detection for streaming data. In: Twenty-Second International Joint Conference on Artificial Intelligence.

[53] Davis J. & Goadrich M. (2006) The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on Machine learning, ACM, pp. 233–240.

[54] Guo X., Yin Y., Dong C., Yang G. & Zhou G. (2008) On the class imbalance problem. In: 2008 Fourth International Conference on Natural Computation, vol. 4, vol. 4, pp. 192–201.

[55] Guha S., Mishra N., Roy G. & Schrijvers O. (2016) Robust random cut forest based anomaly detection on streams. In: International conference on machine learning, pp. 2712–2721.

[56] Weston J., Wang C., Weiss R. & Berenzweig A. (2012) Latent collaborative retrieval. arXiv preprint arXiv:1206.4603 .

[57] Breunig M.M., Kriegel H.P., Ng R.T. & Sander J. (2000) Lof: identifying density-based local outliers. In: ACM sigmod record, vol. 29, ACM, vol. 29, pp. 93–104.

[58] Bhuyan M.H., Bhattacharyya D.K. & Kalita J.K. (2014) Network anomaly detection: Methods, systems and tools. IEEE Communications Surveys & Tutorials 16, pp. 303–336. URL: https://doi.org/10.1109/surv.2013.052213.00046.

[59] Mimmack G.M., Mason S.J. & Galpin J.S. (2001) Choice of distance matrices in cluster analysis: Defining regions. Journal of climate 14, pp. 2790–2797.

[60] Sequeira K. & Zaki M. (2002) Admit: anomaly-based data mining for intrusions. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp. 386–395.

[61] Münz G., Li S. & Carle G. (2007) Traffic anomaly detection using k-means clustering. In: GI/ITG Workshop MMBnet, pp. 13–14.

[62] Boser B.E., Guyon I.M. & Vapnik V.N. (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory, ACM, pp. 144–152.

[63] Schölkopf B., Platt J.C., Shawe-Taylor J., Smola A.J. & Williamson R.C. (2001) Estimating the support of a high-dimensional distribution. Neural Computation 13, pp. 1443–1471. URL: https://doi.org/10.1162/089976601750264965.

[64] Schölkopf B., Williamson R.C., Smola A.J., Shawe-Taylor J. & Platt J.C. (2000) Support vector method for novelty detection. In: Advances in neural information processing systems, pp. 582–588.

[65] Wagner C., François J., State R. & Engel T. (2011) Machine learning approach for IP-flow record anomaly detection. In: NETWORKING 2011, Springer Berlin Heidelberg, pp. 28–39. URL: https://doi.org/10.1007/978-3-642-20757-0_3.

[66] Erfani S.M., Rajasegarar S., Karunasekera S. & Leckie C. (2016) High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. Pattern Recognition 58, pp. 121–134.

[67] Vapnik V.N. (1999) An overview of statistical learning theory. IEEE transactions on neural networks 10, pp. 988–999.

[68] Jakkula V. (2006) Tutorial on support vector machine (svm). School of EECS, Washington State University 37.

[69] Tax D.M. & Duin R.P. (2004) Support vector data description. Machine Learning 54, pp. 45–66. URL: https://doi.org/10.1023/b:mach.0000008084.60811.49.

[70] Rokach L. (2010) Ensemble-based classifiers. Artificial Intelligence Review 33, pp. 1–39. URL: https://doi.org/10.1007/s10462-009-9124-7.

[71] Liu F.T., Ting K.M. & Zhou Z.H. (2012) Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD) 6, p. 3.

[72] Ding Z. & Fei M. (2013) An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. IFAC Proceedings Volumes 46, pp. 12–17. URL: https://doi.org/10.3182/20130902-3-cn-3020.00044.

[73] Marteau P.F., Soheily-Khah S. & Béchet N. (2017) Hybrid isolation forest-application to intrusion detection. arXiv preprint arXiv:1705.03800 .

[74] Das S., Wong W.K., Fern A., Dietterich T.G. & Siddiqui M.A. (2017) Incorporating feedback into tree-based anomaly detection. arXiv preprint arXiv:1708.09441 .

[75] Ng A. et al. (2011) Sparse autoencoder. CS294A Lecture notes 72, pp. 1–19.

[76] LeCun Y., Bengio Y. & Hinton G. (2015) Deep learning. Nature 521, pp. 436–444. URL: https://doi.org/10.1038/nature14539.

[77] Zhou C. & Paffenroth R.C. (2017) Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp. 665–674.

[78] Madani P. & Vlajic N. (2018) Robustness of deep autoencoder in intrusion detection under adversarial contamination. In: Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security, ACM, p. 1.

[79] Malhotra P., Ramakrishnan A., Anand G., Vig L., Agarwal P. & Shroff G. (2016) Lstm-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148 .

[80] Das S., Islam M.R., Jayakodi N.K. & Doppa J.R. (2019) Active anomaly detection via ensembles: Insights, algorithms, and interpretability. arXiv:1901.08930 [Online; accessed 20-Aug-2019].

[81] Das S., Wong W.K., Dietterich T., Fern A. & Emmott A. (2016) Incorporating expert feedback into active anomaly discovery. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, pp. 853–858.

[82] Siddiqui M.A., Fern A., Dietterich T.G., Wright R., Theriault A. & Archer D.W. (2018) Feedback-guided anomaly discovery via online optimization. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, pp. 2200–2209.

[83] Görnitz N., Kloft M., Rieck K. & Brefeld U. (2013) Toward supervised anomaly detection. Journal of Artificial Intelligence Research 46, pp. 235–262.

[84] Wagstaff K.L., Lanza N.L., Thompson D.R., Dietterich T.G. & Gilmore M.S. (2013) Guiding scientific discovery with explanations using demud. In: Twenty-Seventh AAAI Conference on Artificial Intelligence.

[85] Doshi-Velez F. & Kim B. (2017) Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 .

[86] Molnar C. (2019) Interpretable Machine Learning. `https://christophm.github.io/interpretable-ml-book/`.

[87] Letham B., Rudin C., McCormick T.H., Madigan D. et al. (2015) Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics 9, pp. 1350–1371.

[88] Ribeiro M.T., Singh S. & Guestrin C. (2016) Model-agnostic interpretability of machine learning. arXiv preprint arXiv:1606.05386 .

[89] Ribeiro M.T., Singh S. & Guestrin C. (2016) Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp. 1135–1144.

[90] Rudin C. (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 1, pp. 206–215.

[91] Wang T., Rudin C., Velez-Doshi F., Liu Y., Klampfl E. & MacNeille P. (2016) Bayesian rule sets for interpretable classification. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, pp. 1269–1274.

[92] sklearn.preprocessing.robustscaler. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html.

[93] (2016), Insider threat test dataset. URL: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099.

[94] Glasser J. & Lindauer B. (2013) Bridging the gap: A pragmatic approach to generating insider threat data. In: 2013 IEEE Security and Privacy Workshops, IEEE, pp. 98–104.

[95] Zhao Y., Nasrullah Z. & Li Z. (2019) Pyod: A python toolbox for scalable outlier detection. Journal of Machine Learning Research 20, pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

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

[97] Das S. (2018), Active anomaly discovery. https://github.com/shubhomoydas/ad_examples. [Online; accessed 20-Aug-2019].

[98] Liu N., Du M. & Hu X. (2019) Representation interpretation with spatial encoding and multimodal analytics. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 60–68.

9. APPENDICES

Appendix 1 Created decision-rules for anomalies. Shows instances encompassed by rule-based description and their salient features.

Table 10. The user JJM0203 visits job search websites on these days (scenario 2).

(min_logon_session <= 4.70) & (n_http_req > 95.00) & (max_file_time <= 5.90)			
Date	min_logon_session	n_http_req	max_file_time
2010-09-08	4	103	0
2010-09-09	4	101	0
2010-09-14	0	105	0
2010-09-15	0	101	0
2010-09-28	4	96	0
2010-09-29	0	104	0
2010-10-01	0	98	0

Appendix 1 Created decision-rules for incidents. Shows instances encompassed by rules and their salient features.

Table 11. The user AF0535 visits job search websites on these days (scenario 2).

(n_http_req > 32.04) & (n_file <= 0.67)		
Date	n_http_req	n_file
2010-07-02	38	0
2010-07-09	38	0
2010-07-14	34	0
2010-07-19	33	0
2010-07-20	35	0
2010-07-26	39	0

Table 12. The user JJM0203 steals data on these days using thumb drive (scenario 2).

n_dev_conn > 4.96	
Date	n_dev_conn
2010-10-04	7
2010-10-07	6
2010-10-12	5
2010-10-14	6
2010-10-19	7

Table 13. On these days, disgruntled system administrators complain to supervisor FBA0348 and sends out mass email (scenario 3).

(n_email > 14.64) & (n_http_hostname <= 38.94)			
Date	n_email	n_http_hostname	Description
2010-09-30	16	31	BSS0369 complains via email
2010-10-01	15	25	Mass email
2011-04-28	16	29	JLM0364 complains via email

Appendix 1 Created decision-rules for incidents. Shows instances encompassed by rules and their salient features.

Table 14. On these days, disgruntled system administrator logins as supervisor FBA0348 (scenario 3).

n_logon > 1.83	
Date	n_logon
2010-10-01	2
2011-04-29	2

Table 15. Rule for the user WDD0366 (scenario 1). Example of how rule-based descriptions occasionally are overly complex and lack accuracy (TP = true positive).

(n_logon > 5) & (n_logon_pc > 3) & (n_logon_pc <= 5) & (n_http_hostname <= 36) & (min_file_time <= 12)					
Date	n_logon	n_logon_pc	n_http_hostname	min_file_time	TP
2010-01-29	6	5	34	0	false
2010-02-05	6	5	27	0	false
2010-02-22	6	5	34	0	false
2010-02-25	6	5	27	0	false
2010-04-12	6	5	21	0	false
2010-05-25	6	5	23	0	false
2010-06-07	6	5	33	0	false
2010-08-25	6	5	24	0	false
2010-10-22	6	5	32	0	false
2011-01-17	6	5	25	0	false
2011-01-24	6	5	27	0	false
2011-01-28	6	5	26	0	false
2011-03-02	7	5	24	0	true
2011-03-03	6	5	35	0	true