FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	Automatic detection of malicious activity in the internal network
Student:	Bc. Mikuláš Formánek
Supervisor:	Dawid Machala, Ph.D.
Study program:	Informatics
Branch / specialization:	Computer Security
Department:	Department of Information Security
Validity:	until the end of summer semester 2021/2022

Instructions

The proposed thesis focuses on automatic detection of malicious activity of users (or other agents) within the internal network of Showmax and blacklisting (sinkholing) them. Minimum scope of work assumes that the student will:
1. Prepare a summary of several state-of-the-art approaches of automatic detection of malicious activity, concentrating on approaches as close to real-time as possible. Selected algorithms should be capable of at least detecting and preventing brute-force attacks aiming to steal private user data.
2. Select and implement a proposed solution as a proof-of-concept.
3. Test the proposed solution on a prepared dataset.
4. Discuss the scalability of the solution and accuracy of the results.
Ideally, the student should be able to apply statistical or machine-learning procedures to model and detect the behavioural traces of a typical attack in a large real-life dataset. Thesis is conducted in cooperation with an African movie-streaming company, Showmax.

Electronically approved by prof. Ing. Róbert Lórencz, CSc. on 15 December 2020 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Automatic detection of malicious activity in the internal network

Bc. Mikuláš Formánek

Department of Information Security
Supervisor: Dawid Machala, Phd.

May 6, 2021

Acknowledgements

I would like to thank Dawid Machala for his never-ending positivity throughout the work, as well as my family and friends for their support

Declaration

In Prague on May 6, 2021 …………………

Citation of this thesis

Formánek, Mikuláš. *Automatic detection of malicious activity in the internal network.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Práce se zabývá automatickou detekcí škodlivého chování na vnitřní síti, srovnává nejnovější metody pro detekci. Praktická část je zaměřená na detekci útoků hrubou silou pomocí strojového učení bez učitele. Navrhované řešení bylo odzkoušenu na datové sadě, z dřívěji použitého algoritmu v infrastruktuře firmy Showmax. Citlivostní analýza ukázala, že řešení může být použito v infrastruktuře Showmax.

Klíčová slova Strojové učení, automatická detekce, útoky hrubou silou

Abstract

The thesis focuses on automatic detection of malicious activity in the internal network. The subject literature is reviewed, with emphasis on unsupervised anomaly detection methods. The proposed algorithm was evaluated against a threshold-based algorithm used at Showmax, on the same dataset. Sensitivity analysis has been performed and supports the idea of applying the algorithm on Showmax's infrastructure.

Keywords Machine Learning, Deep learning, intrusion detection, brute force attacks

Contents

List of Figures

List of Tables

Introduction

“Good to all, evil only to whom seek it.”

– ancient wisdom from Don Quixote

A trend leading towards better protection against cyberattacks has taken place since the dawn of the internet. It is especially true nowadays, for organisations that have access to large amounts of personal data of their users: healthcare, governmental, financial, and large businesses [1]. For example, 66% of American healthcare providers have experienced ransomware attacks in 2019 [1]. Similarly, a threefold increase in ransomware attacks has been reported between 2018 and 2019 among larger businesses, accompanied by a decrease of ransomware attacks on individual users, suggesting a trend of cyberattackers to concentrate on larger organisations [2].

While the impact of attacks on the critical infrastructure (such as hospitals, governmental institutions, or banks) is the most impactful for individual users, large companies that were subject to a successful attack can suffer the loss of trust of their partners and customers. Showmax is a video-on-demand service that provides access to movies, series, and sport events, often produced by third parties. As such, assuring appropriate security measures is a contractual obligation to the third parties. Similarly, each security breach needs to be reported publicly, due to GDPR regulations. Therefore, protection against cyberattack is an important issue.

This thesis is conducted in cooperation with Showmax, and is focused on an autonomous anomaly detection in the internal company network.

Detection of attacks becomes more challenging with higher amounts of data that needs to be analysed. Unlike old-school firewalls used in PCs, network attack detection has to deal with issues of scalability and real-time detection. Modern-day Detection Systems can operate on a model-based approach, with machine learning algorithms aiding them with detecting coordinated attacks across different networks, identifying infiltrated devices in the network, or with quick detection of brute-force attacks.

Goals and outline

The thesis is focused on detecting brute-force attacks at Showmax infrastructure, with the emphasis on unsupervised detection.

The goal of this thesis can be summarized in the following way:

- Provide a state of the art analysis of unsupervised anomaly detection and detection system
- Prepare a data set for brute-force detection
- Propose a proof-of-concept of brute-force detection mechanism which could detect brute force attacks
- Test the proof-of-concept on the prepared data set and discuss its and accuracy

Structure of the works is the following. In the first chapter, we show how Detection Systems can be categorised, how are anomalies and types of attacks defined, and what are the basic techniques and theory of outlier detection.

In the second chapter, we provide a summary of state-of-the-art approaches, from commercial products and community solutions to scientific papers and the latest approaches from the last 6 years.

The third chapter contains the analysis of the Showmax's environment, data and feature availability, followed by the proposed solution for detecting brute-force attacks. Afterwards, the realisation of the proposed solution and its evaluation using sensitivity analysis, are presented.

Chapter 1

Theory

"Space is only noise if you can see.
Grab a calculator and fix yourself"– Nicolas Jaar

1.1 Detection System Taxonomy

We can divide Detection systems into several cases (as shown at fig. 1.1): response to the attack and its delay, architecture, source of data, the mechanism for detection attacks. From a functional point of view, we have Intrusion Detection System (IDS) and Intrusion Prevention System (IPS), IDS is trying to alert for incoming attacks, possibly as soon as possible, IPS is trying to protect the infrastructure by itself for: example with denying access for the attacker or hacking attacker itself. However, such countermeasures could easily balance on the edge of lawfulness.[3, p. 1]

The second view on Detection System can be based on the source of activity, Network Intrusion Detection System (NIDS) is getting data from all switches, routers, and Network Interface Card (NIC) in servers, whereas Host Based Intrusion Detection System (HIDS) can look even to application logs, File System (FS) logs as well to the application database and network devices, therefore we call NIDS being a subset of HIDS.

The third dimension of Detection Systems is the mechanism of detection. Firstly, the detection can be signature-based where we detect malicious activity by comparing parts of registered activity(in other words, using fingerprinting) with already known attacks from the database of attacks. However, this approach is limited to the current state of the database, so potentially zero-day exploits can potentially remain undetected. The second option is anomaly detection, where the algorithm is trying to detect outliers in infrastructure activity, which provides more False Positive (FP)s but is capable of detecting previously-unknown forms of attack. IDS based on Anomaly Detection will be of the primary interest in the rest of this thesis.

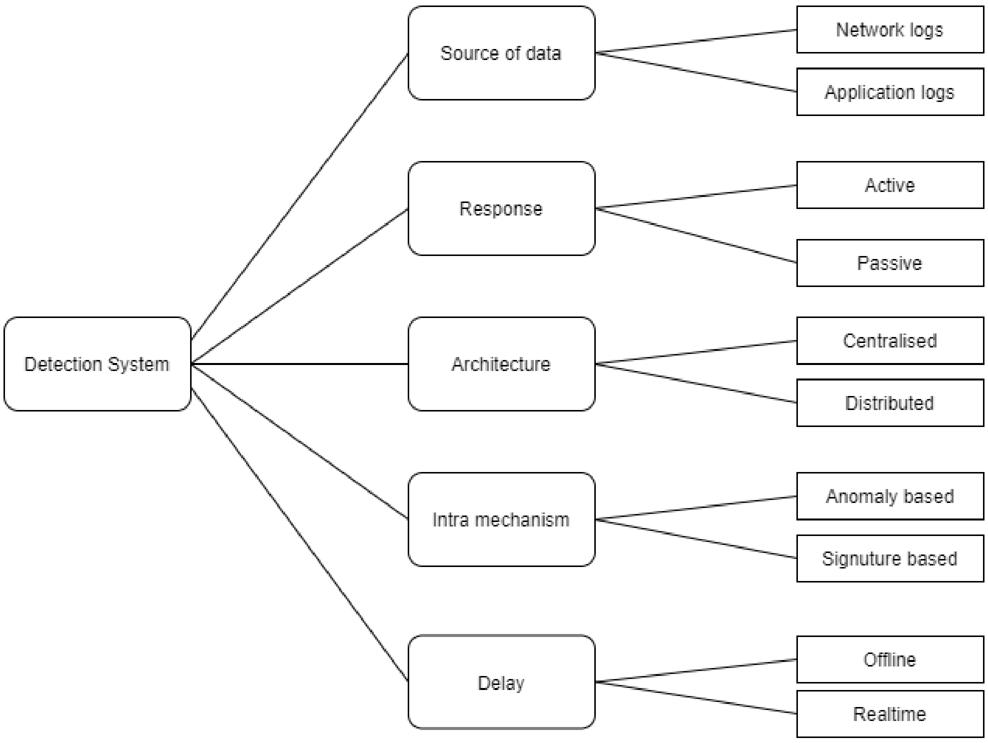


Figure 1.1: Taxonomy of detection systems

The fourth dimensions could be based on the architecture type: either centralised or distributed. Yet another view in taxonomy could be based on providing detection offline or in real time.

1.1.1 Anomaly detection

Stephen Hawking defines an outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.[4, slide 3]

There are many ways of anomaly/outlier definition. Since the data set can be noisy, the best way of defining anomalies is based on analyst expertise: especially taking advantage of the previously known examples of anomalies.[5, p. 4]

To make a distinction between an outlier and anomaly, Aggarwal et al. [5, p. 3] have concluded to the fact they are interchangeable(anomaly being subset of outlier based on interest). The general definition could say that we consider an instance to be an anomaly, if we haven't seen such an instance before, in terms of some measures.[6, p. 4]

1.1.1.1 Types of anomalies

There are several types of anomalies based on domain, therefore, only the anomalies in the context of the network security domain will be considered hereinafter (also shown at fig. 1.3).

The first type of anomaly is the point anomaly, which is extreme in terms of irregularity or deviation. This anomaly is related to User2Root (when the user tries to escalate his privileges) and Remote2Local (when an attacker tries to obtain internal access from outside of the network) attacks.

The second type is a contextual anomaly, that is: an anomaly based on surrounding-context data points. This can be related to probe attacks[7, p. 4], context attributes can be, for example, a time or location, opposite for contextual attributes is the behavioral attribute for example temperature. [8, p. 8]

The third type is a collective anomaly, which is connected to DOS attacks and corresponds to a situation when multiple data-streams are abnormal together although individual values from each data-stream are not anomalous on their own..

1.1.1.2 Attack classification

Although each cybersecurity attack is unique, they are usually classified by several taxonomies, with the most common being provided in 2003 by Simon Hansman [9, p. 31]. The first dimension of the classification is by the attack vector, which represents the way attacks try to reach their target. When performing such a classification, we should classify the attack vector by its earliest behavior.

The second dimension describes the target of such a cyber attack. Hansman et al. [9, p. 32] provides a full table for each dimension, but most importantly the target must be specified as much as possible. For example, a virus destroys the firmware in a keyboard connected to our laptop, then we should not classify the target as the operation system, but rather the hardware itself.

The third dimension concentrates on the exploits and vulnerabilities which has been used by the by the attacker. Many vulnerabilities are described by Common Vulnerabilities and Exposures (CVE). *It should be noted that vulnerabilities are wide and varied and usually apply to specific versions of a piece of software or operating systems. This means a classification scheme would have to include every piece of software in use today.*[9, p. 35]

The fourth dimension captures the effect of the attack into these categories:

- First dimension Attack
- Corruption of Information
- Disclosure of Information

- Theft of Service
- Subversion

Other dimensions such as: damage, costs, propagation or the defence mechanism could be potentially added as well.

This taxonomy above is especially useful when reporting incidents, however when trying to stop attackers, there has been proposed Intrusion Kill Cain [10, p. 4], which is derived from the military model, where stages established to identify, prepare to attack, engage, and destroy the target were converted into (Shown at 1.2):

- Reconnaissance: research, selection, using any OSINT information about target
- Weaponization: preparing adequate forms of attacks
- Delivery: sending attack into the desired environment
- Exploitation: running attack's code on the target machine
- Installation: Obtaining backdoor access
- Command and Control: obtaining covert channel for further actions on target
- Actions on Objectives: taking actions on objectives

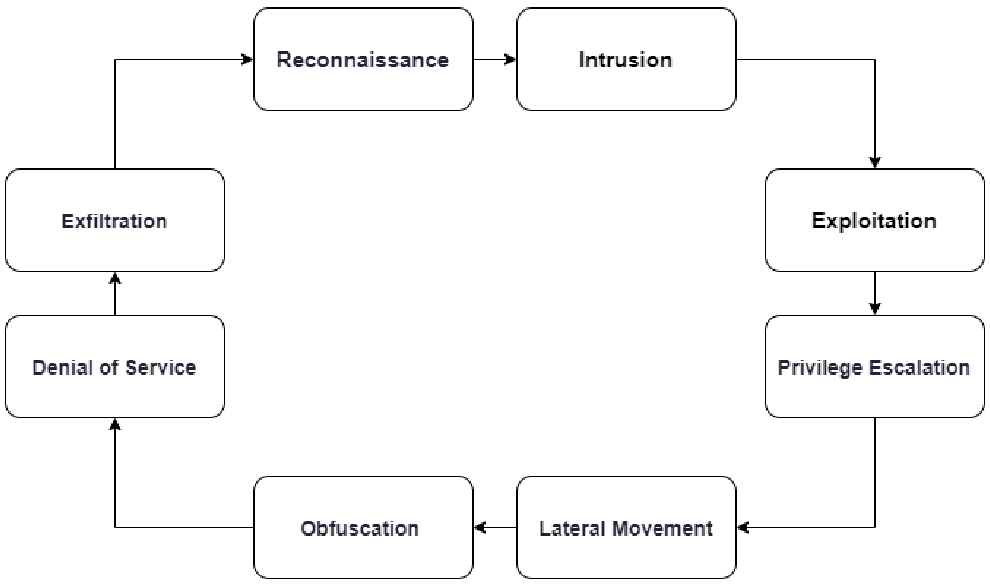


Figure 1.2: Cyber Kill Chain

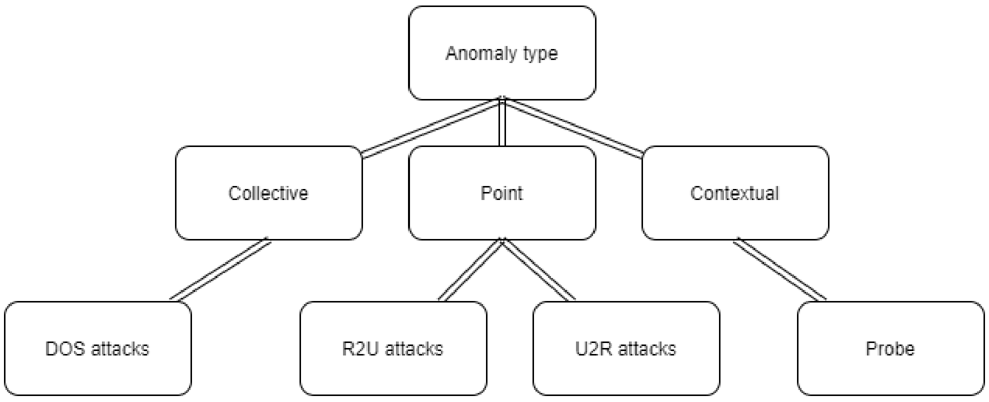


Figure 1.3: Taxonomy of anomalies with network attacks

It should be noted that while point anomalies can occur in any data set, the collective anomalies can occur only in data sets in which data instances are related. In contrast, the occurrence of contextual anomalies depends on the availability of context attributes in the data. A point anomaly or a collective anomaly can also be a contextual anomaly if analyzed with respect to a context. Thus a point anomaly detection problem or collective anomaly detection problem can be transformed into a contextual anomaly detection problem by incorporating the context information.[8, p. 10]

1.2 Data Modeling

In short, one can say that all outlier algorithms create a model of normal data and then try to compare it with each new data point on the basis of deviations from these patterns. Different data models create different assumptions about normality in data. Therefore, choosing a good data model is crucial for anomaly detection: inadequate data model leads to poorer performance.

1.2.1 Supervised and unsupervised models

Supervised learning assumes labels are available for each data point. In such a framework, classification of new data points can be considered as equivalent to an anomaly detection task. For example, let's assume that an existing data set consists of data mostly corresponding to the standard behaviour of the system. Although it probably contains some anomalies, assumed to be in minority, we can classify new data points based on based on their deviation from the normal model behaviour. This deviation can be interpreted as as outlier scores. Therefore we can use each classification algorithm as a supervised anomaly detection algorithm. [11, p. 25] One [6, p. 8] would say that this kind of one class setting is equivalent to a multi-class setting, however, creating a distinction between two and more classes is much easier than comparing an unknown instance to a normal class. Many supervised algorithms got their unsupervised replacement.

In unsupervised learning we do not have labels for instances available. In supervised learning we can divide data set into training and testing for scoring, comparing different algorithms. Without labels, the typical approach of dividing the data set into the training and testing subsets becomes impossible. *Nevertheless, the influence of individual points on over-fitting is often small in real-world settings because explicit generalization methods tend to create a concise summary (i.e., generalized representation) of a much larger as the training data errors made in the assumption of "pretending" that all training data set. Since the same data D is used for training and testing, one can view outlier scores points belong to the normal class.*[11, p. 9]

1.2.2 Selecting features

Data sets can consist of multiple features, possibly of different variable types, and, it is a key value of an analyst to determine which feature is useful and some of them can even worsen the performance of the data model. Another option is to introduce subsampling methods: which are used in ensemble techniques and will be discussed later. Our set of features is divided into several subsets and used with several instances of the same or various data models.

1.2.2.1 High dimensional issues

Data sets for anomaly detection can contain hundreds of features, causing the problem to become a high dimensionality problem. Not every method can deal with data of higher dimensions, as each data point becomes sparsely located and therefore [5, p.149] can be equidistant from each other. This problem is sometimes called *the curse of dimensionality.* There is also the possibility that the most important dimensions are outnumbered by dimensions of lower importance. One way out is to execute subspace analysis. *Such an approach filters out the additive noise effects of the large number of dimensions and results in more robust outliers. An interesting observation is that such lower-dimensional projections can often be identified even in data sets with missing attribute values. This is quite useful for many real applications, in which feature extraction is a difficult process and full feature descriptions often do not exist.* [5, p. 151] As shown in [12, p. 1] subspace analysis have advantage of showing outliers in subspace view, however traditional full feature algorithms or trying to detect deviations in global view, which could potentially hide some anomalies(shown at fig. 1.4).

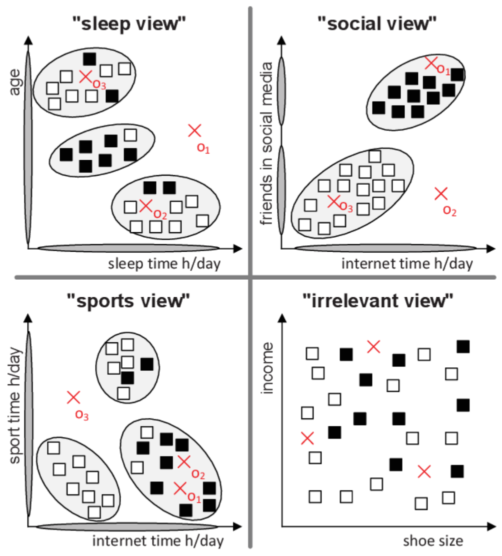


Figure 1.4: Hidden anomalies in sub-views from [12, Figure 1.; Page 1]

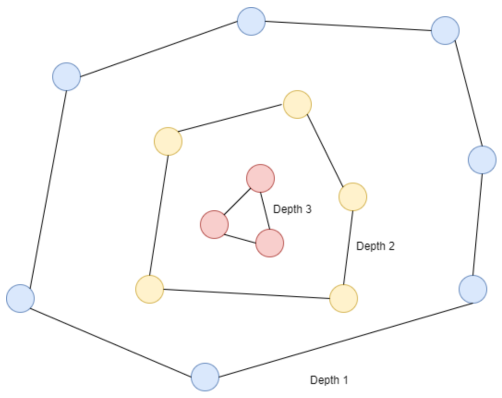


Figure 1.5: Outliers on convex hull

1.3 Basic models for anomaly detection

1.3.1 Depth based methods

Depth-based methods can be treated as extreme-value analysis methods with emphasis on convex hull analysis. The method assumes that the outlier is most likely in a corner of the convex hull. Sets of points that are corners of the convex hull in the data set are removed in an iterative manner, with the iteration taking place over each i-th corner. The process takes place until the data set is emptied, after which the sets of points with the highest depth are reported as outliers. (shown at fig. 1.5).

Such a technique is scales badly with the number of dimensions and cannot find an inner outlier, due to the assumption that outliers are located around the corners of the convex hull. Therefore, the method cannot be applied unless the problem is of lower dimensions, and doesn't have inner outliers[13, p. 225].

1.3.2 Proximity-based methods

Proximity-based techniques define a data point as an outlier when its locality (or proximity) is sparsely populated. The proximity of a data point may be defined in a variety of ways, which are subtly different from one another but are similar enough to merit unified treatment within a single chapter.[5, p. 111]

Three-way of describing proximity are:

- Cluster based
- Distance based
- Density based

Cluster based methods use association within-cluster or distance from cluster centroids as outlier score. The aim of clustering algorithms is to separate data points into dense clusters. Outlier are naturally becoming a side product, however, sometimes outliers are whole clusters or become a smaller cluster on its own.

Therefore is introduced distance from data point into cluster centroid, a distance of a data point to its closest cluster can be used as a proxy for outlier score. [5, p. 113] Clustering is a fast method compared to distance based methods. *The main disadvantage of clustering methods is that they might not always provide insights at the required level of detail in smaller data sets. The granularity of outlier analysis methods is generally better when using distance computations directly with respect to the original data points, rather than with respect to aggregated representatives such as cluster centroids.*[5, p. 118]

Distance-based methods use the distance of nearest k-th data-point as metric for outlier score, where the underlying assumption is that larger distance is related to the larger probability of being an outlier in data set as opposite normal points should have lesser distance. *Distance-based methods generally have a higher granularity of analysis as compared to clustering-based methods. This property of distance-based methods can enable a more refined ability to distinguish between weak and strong outliers in noisy data sets.* [5, p. 119] However, in the base version of the k-nn algorithm, where we output outlier scores, we need to compute N*N distances, which for large data sets very impractical. Therefore is introduced several options, to output just binary label and prune a certain amount of computation:

- Threshold based data-point is outlier if its exact k-nearest neighbor distance is at least beta

- Cell based pruning - data-points are divided into cells at each dimension and is independent of number of data-points
- Sample based pruning, where we limit pair-wise distance computation on a subset of data set.

Density-based methods use the number of points in specific regions of the space to define outliers. They are very closely related to clustering and distance-based methods. In fact, some of these algorithms can be more easily considered clustering or distance-based methods, depending on how they are presented. This is because the notions of distances, clustering, and density are closely related and inter-dependent.[5, p. 131]

1.3.3 Linear models

Linear models assume that normal data lies in lower-dimensional subspace and focus on the use of dependencies between the features. Linear models can be viewed as an orthogonal point of view to clustering- or nearest-neighbor methods, which try to summarize the data horizontally (i.e., on the rows or data values), rather than vertically (i.e., on the columns or dimensions). [5, p. 65]

Well known algorithms for outliers detection are:

- Principal Component Analysis (PCA)
- One Class Support Vector Machine (OC-SVM)
- Deep Learning approach

PCA is dimension-reduction technique, output of PCA is set of vectors which are called principal components. Vectors capture the highest variance in the original data set. According to [14] the main assumption is that subspace returned by number of k vectors corresponds to regular trends. According to [5] PCA is quite robust technique for number of outliers in data set, but [14, p. 110] points out, that PCA cannot always flag beginning of anomaly in time-series detection.

OC-SVM is spatial setting for classic Support Vector Machine (SVM), where it is assumed that all instances are having only normal data points label. OC-SVM learns the boundary for normal data point, in classification phase, anything which does not fall into boundary is classified as outlier. According to [15, p. 52] is especially useful for high-dimensional data in semi-supervised setting. Trying to apply OC-SVM in unsupervised setting puts validity of results on stake, because in training data set can be outliers, which would be lately classified as normal data-points.

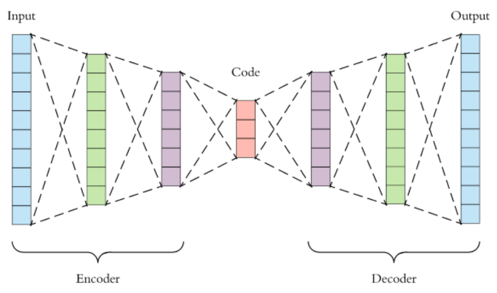


Figure 1.6: Architecture of Auto-Encoder available on: [16]

1.3.3.0.1 Auto-Encoders Deep Learning approach became popular in last decade, with applications ranging from computer vision to generative adversarial networks. Anomaly detection algorithm within the deep learning framework have been proposed as well, with the most common being neural networks called Autoencoder (AE). They consist of two parts - encoder and decoder. The encoder tries to reduce set of feature to lower-dimensional subspace and when it is reduced decoder attempts to reconstruct reduced features to original input, therefore training phase in unsupervised manner, where the the label of x is x' itself and the goal is to make reconstruction error $x - x' = 0$. Underlying assumption is that outliers are reconstructed with higher error ξ, because outliers are resistant to compression. According to [5, p. 105] a symmetric architecture(shown in fig. 1.6) of AE provides hierarchically reduction on different level of compression.

One of disadvantages of AE is the long training time due to complexity of computations, but since GPU's are well provided in cloud computing we can use its potential for scaling. Next disadvantage is similar to OC-SVM, where having noise in training data will cause overfitting when used on test data, this could be partially solved with some level of cross-validation technique, where just subset of data is used for training. Next factor is interpretability issue, where we with lot of linear models we cannot provide reasoning to why we consider data-point anomalous.

1.3.3.0.2 Boltzman machines Boltzman machine is special type of bidirectional neural network, it has many specifics, they consist only of hidden nodes and visible nodes (shown at fig. 1.7), which are the only we can interact with. They lack the training phase [15, p. 180] and their purpose was to learn data distribution. They are also able to create generative model: creating new data according to data set. Each neuron has fixed weight and makes probabilistic decision if it will fire or not. Their neurons are connected to each other, therefore scaling the network up is difficult. Boltzman machines are unsupervised deep learning aproach.

A much more scalable variant of Boltzman machines is Restricted Boltzmann Machine (RBM), which architecture is a bipartite-graph of two lay-

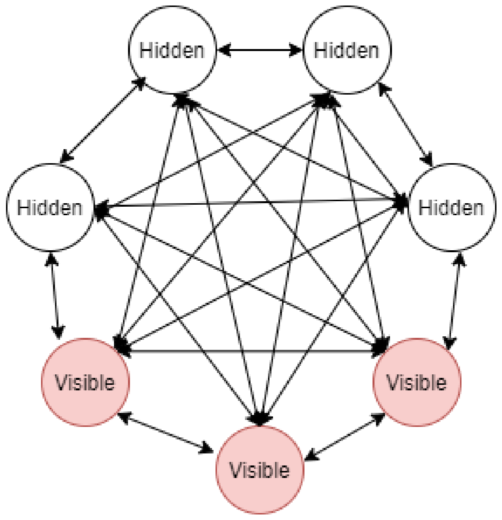


Figure 1.7: Architecture of Boltzman Machine

ers (input and hidden), where nodes in one layer are not connected between themselves. Similarly to Boltzman Machines it is generative model, but has a training phase. *Some RBM might also incorporate a feature known as momentum, which basically allows for an increase in learning speed and can be thought of as simulating a ball rolling down a hill in terms of optimizing the target function.*[15, p. 186]

1.3.3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are special type of neural network. They consist of multiple layers stacked together. Typical use case is in Computer Vision, or together with AE, where we can benefit from unsupervised setting, but as can be seen in section 2.1.1 the usage is much broader than pixel-wise operations on images. Main block of CNN is convolutional layer, which main objective is to extract high dimensional features. The next common layer is pooling layer, which can apply min or max function to certain number of pixels like in the fig. 1.8 Dropout layer is used against overfitting, where models tends to be too much focused on training data and generalization fails. Dropout layers has one parameters, which says how many inputs are going to be randomly set to 0.

1.3.4 Ensembles

In supervised setting we benefit from having labels, therefore calculating any metrics such as F1, AUC, precision, recall or accuracy is very intuitive, allowing easier minimisation of the bias or variance.

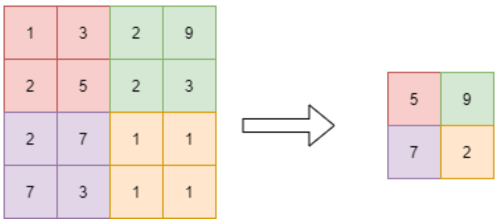


Figure 1.8: Max pooling operation

While this is not possible in the unsupervised setting, there is an option to treat one or more feature as unobserved and dependent, which does not tell us the ground truth if such data-point is anomaly or not, but we have certain possibility to measure our model.[11, p. 24]

Ensemble methods consist of joining several base detectors, which are in some way united together for obtaining better results. According to [17] [5, p.188] we can divide ensemble methods on several points of views:

- Independent ensembles
- Sequential ensembles

Independent ensembles consists one or more types of base detectors and output of single base detector does not affect rest of detectors. Motivation is to explore independence between base detectors, which can potentially lead to better results.

Sequential ensembles consists also one or more types of base detectors and their output affects each other, so when detector miss-classify data-points, sequential detector can learn (boost) and obtain better results. This option is [5, p. 29-30] hard to use in unsupervised setting, because missing the ground-truth for awareness of getting worsen results.

Next dimension for ensemble methods is to divide them into:

- Model-centric
- Data-centric

Model-centric ensembles do consist of different models and their hyper-parameters for example in a randomized manner.

Data-centric models are a subset of model-centric models, but they split data sets by columns and each base detectors receives a different subset of data set.

Three options [17] for improvement:

- Boosting for reducing bias, sequence ensemble
- Bagging for reducing variance, data-centric ensemble

- Stacking for reducing bias and variance

Stacking methods are different that they have a meta estimator, which gets as input the output of the base detector, such approach is currently very successful for many data science competitions like `https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335` Using meta estimator could be seen as an alternative for independent ensemble models, where the output of these is used for voting in classification, naturally is also used some weighting average or some maximization function.

Outlier ensembles have seen an increasing interest from the research community in recent years, which is motivated, in part, by increasingly challenging data sets that are resistant to accurate outlier discovery. An example of such a challenging instantiation is the high-dimensional case in which no single subspace can capture all the outliers. In spite of the superficial differences between the two problems, the theoretical foundations of outlier en- sembles are similar to that in the supervised case.[5, p. 217]

1.4 Conclusion

In this chapter we introduced basic anomaly detection techniques, showed their advantages and weak spots. Several machine learning concepts have been shown, which will be used in the next State of the art chapter. Definition of outliers was explained and presented in the context of cybersecurity attacks types of anomalies and taxonomy of IDS. In the end, an overview of typical ensemble methods has been shown.

Chapter 2

State of the art of malicious activity detection

Previous chapter shows basic methods for anomaly detection, this chapter introduces state of the art techniques, that have been recently proposed by research papers

Motivation for reviewing state-of-the-art is to collect useful approaches which will in latter chapters enhance proposed algorithms. Well known techniques were described in previous sections, but following chapter describes innovative approaches.

Firstly anomaly detection models will be reviewed as main focus of theses is on unsupervised learning, next part of this chapter will compare currently used IDS commercially or open-source.

2.1 Recent techniques for anomaly detection

Techniques in this section are built on the foundation shown in Chapter 1, showing mostly unsupervised of each paper/technique from the last 6 years.

2.1.1 Deep-Ant

Deep-Ant [18] tries to address periodically-changing data (i.e. seasonality in data) that density-based models are unable to treat. Deep-Ant belongs to family of time-series unsupervised detection. Consists of two modules: time-series forecaster, and anomaly detection component. The latter can detect the point and contextual anomalies, as well as discords shown atfig. 2.1.

The time-series predictor uses deep CNN, applying windows of time-series as context, than generates the next time-stamp. The purpose of using CNN is to reduce the number of training samples compared to LSTM. The predictor consists of several convolutional layers (shown at fig. 2.2), each of them firstly uses a convolution function and then applies a linear activation function. After

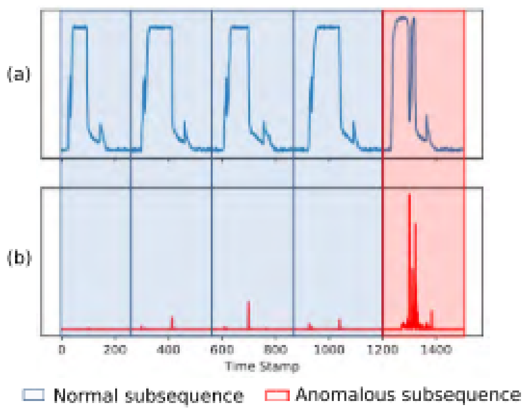


Figure 2.1: DeepAnT is capable of detecting time series discords: subsequence (part) of time series which differs from other subsequences. Normal subsequence time series are highlighted in blue, whereas discord sequences are highlighted in red (a). Plot (b) shows point-wise anomaly score of a subsequence. [18] Reprinted from: [18]

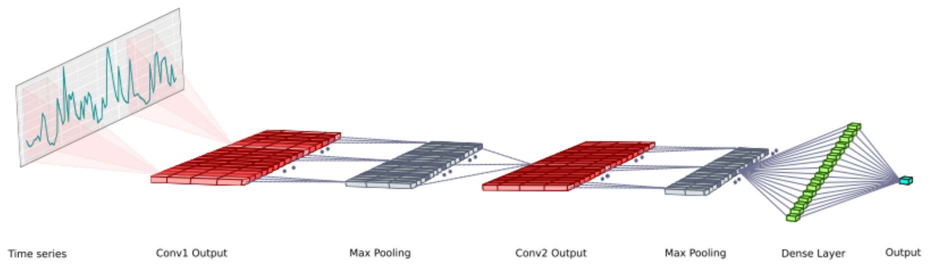


Figure 2.2: Architecture of Deep-ant architecture predictor available from: [18]

each convolutional layer is the max-pooling layer, which summarizes the output of the convolutional network by max function with respect to neighbors. The last layer in the architecture is a fully connected layer of neurons, which are then used for computing the single output y then output is passed into the Detector module. If needed, the predictor does not need to output a single value but can forecast even further, just by adding more output neurons. The outlier score is the Euclidean distance:

$$(y_t, y'_t) = \sqrt{(y_t - y'_t)^2}$$

For each element in time-series has element x^{t} its label $\mathrm{x}^{\mathrm{t+1}}$, based on size of window we incorporate number of data-points before X^{t} into account, also called history window for example

$$x_{t-3}, x_{t-2}, x_{t-1}, x_t$$

is used for predicting x_{t+1}.

This distance based euclidean score is used in paper used within 15 different detector modules. Authors rely on assumption that better prediction makes better anomaly detection.

They evaluated on Numenta Anomaly Benchmark[19], Yahoo dataset, and 7 other datasets from UCI Machine Learning Repository and obtained outperforming results compared to iForest, OC-SVM, Local Outlier Factor, where the metric was Area Under Curve (AUC). Authors claim that CNN are less hungry than LSTM based predictor which obtained lower score in 3 sub-benchmarks out of 4, sharing the same amount (40%) of data for training. *The method is capable of handling minor data contamination (less than 5%). This technique is accurate even in the detection of small deviations/anomalies in time series cycles which are generally overlooked by other distance based and density based anomaly detection techniques*[18, p. 2002] There is a 5% limitation of handling anomalies in the training set, of the level is increased the Time-series forecaster tries to forecast anomalous data-points as well, which then ruins outlier score for Detector module. The size of the history window needs to be defined in heuristic way, specifically to an applied domain, because selecting a single size for each possible application is a comprehensive quest. All the results have been obtained using two convolutional networks, but this can be also extended with another layer, depending on the domain(for example size of dataset).

2.1.2 EGADS

EGADS is a framework developed at Yahoo, the purpose of creating this framework was to enhance the current state-of-the-art algorithms which suffer from scalability and a large number of false positives [20].

Architecture (shown at fig. 2.3) consists of time-series forecaster and anomaly detection similar to section 2.1.1 called an alerting module. The whole integration consists of storing data on a cluster, then having batch model generators that learn models on stored data, which are then saved into the database. Online data are processed by anomaly detection module, if detected there are specific rules under the anomaly is an alert event.

The emphasis on scalability is done via precomputing models as soon as possible, not storing them in hard drives and sharing them across multiple time series, which saves I/O operations to model databases. Another possibility to save some CPU time would be to use self-tuning models, which would change according to incoming data, but this would lead [20, p. 1941] to higher I/O to model databases.

EGADS detector is based on 9 features extracted from time-series:

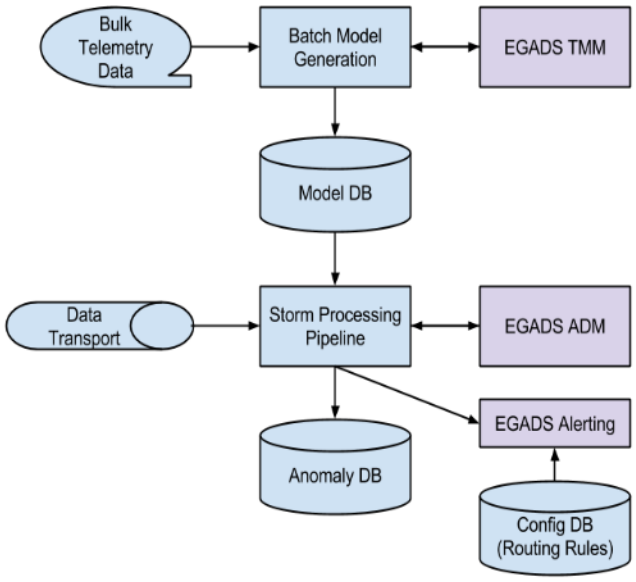


Figure 2.3: EGADS architecture: [20]

Frequency

Trend: rising, still or falling

Seasonality: measures if subsequences are repeating

Auto-correlation: measures correlation of delayed signal copy

Non-linearity: measures linearity of data

Skewness: measures symmetry

Kurtois: measures peaks

Hurst: measures long-term memory of time-series

Lyapunov Exponent: measures rate divergences

After the features of the time series are extracted, the sequence is divided into subsequences, or clusters.

Division into clusters is dependent on thresholds. that need to be selected prior. The first one with underlying assumption, that data are normally distributed and three-sigma(or k-sigma, where k is positive integer) rule, where 99.73% samples within three standard deviations of the mean. *Therefore, depending on the value of K in K-sigma, one can be confident as to the probability of observing a sample at time t. Depending on the desired level of sensitivity, one can measure if a given sample lies within the 95.45% or 68.27% of all the* *samples for K = 2 or 1 respectively.* [20, p. 1942] Second approach is for cases when the deviation metric is not normally distributed and uses density based models called Local Outlier Factor.

The framework was tested in three variants against open-source models(Twitter Outlier,ExtremeI and II R Outlier, BreakOut Twitter CP, ChangePt1 R CP), using an F1 metric for evaluation. No single model has outperformed all data sets, and Twitter_outlier performed similar or better than each from single EGADS models and on 2 datasets is EGADS not competitive, therefore is not considered as good option. (fig. 2.4).

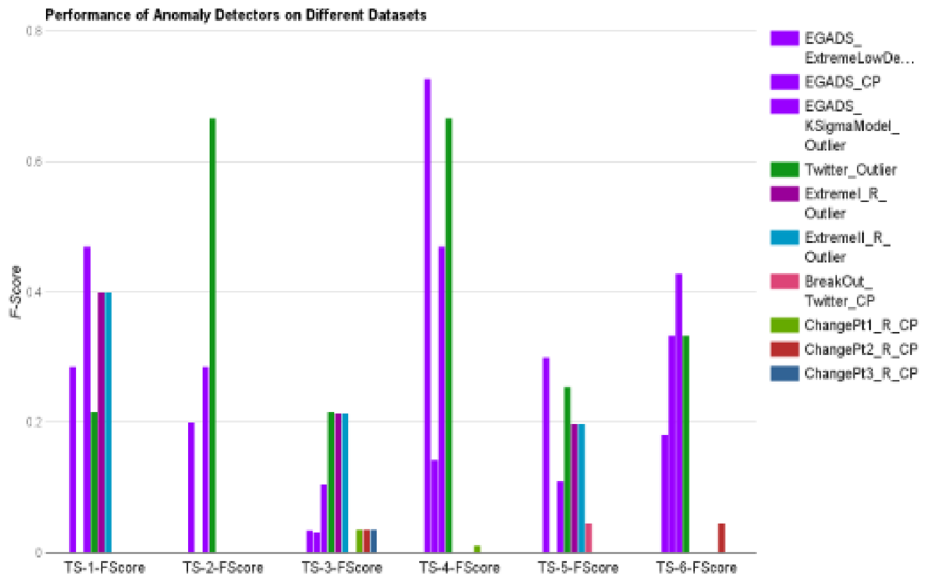


Figure 2.4: EGADS perfomance. Reprinted from: [20]

2.1.3 Numenta Anomaly Benchmark

Numenta Anomaly Benchmark (NAB) addresses issues specific for streaming services, such as: a real-time anomaly detection allowing an early detection of arising problems, or the difficulties with dividing the dataset into a training and testing sets. NAB corpus includes metrics like network utilization, sensors from industrial machines to social media chatter as well as artificially generated data.

NAB has a set of rules, which determines the quality of the anomaly detector:

- Detect all anomalies
- Detect anomalies as soon as possible

- No false positives
- No look ahead in data - real-time
- Data specific parameter tuning must be done online without human intervention

Early detection is performed using anomaly windows. They are located around these data points that were manually-labeled as anomalous. Then, a detector is trained to detect these anomalies within each window: the sooner the detector announces the anomaly the better. Penalisation of late detections is applied as well. The size of these windows is 10% of data points divided by a number of anomalous instances but [19] points out that even the increased size of windows to 20% or 5% does not affect the scoring function. If multiple detections within single anomaly windows only the earliest is taken into account. NAB scoring function requires weights for true and false positives and negatives. Here they are noted as: W_{TP},W_{FP},W_{TN},W_{FN}: The scoring function itself is sigmoidal (show at fig. 2.5), where y stands for a relative position in the anomaly window:

$$f^W(y) = (W_{TP} - W_{FP})(\frac{1}{1+\epsilon^5 y}) - 1$$

Raw score for for data-file is:

$$S^W = (\sum f^W(y)) + W_{FN} f_d$$

and final score is then:

$$S^W_{NAB} = 100 * \frac{S^W - S^W_{null}}{S^A_{perfect} - S^A_{null}}$$

Where perfect and null refers to a detector that detects only TP and no FP respective detector which does not detect anything.

The main advantages are the use of penalization for a late-detection of anomalies, as well as the possibility of creating own profile based on a specific domain. The critique of NAB approach has been presented in [18]

2.1.4 Extended Isolation Forest

This approach [21] improves upon technique for unsupervised learning: Isolation Forest.

Isolation Forest creates multiple binary trees. Each node in binary tree have at most 2 children and start in root node. Each splitting from parent node to children divide set of datapoints by it's feature values by comparing values $<$ or $>$. Leaf in binary tree has no child and represent datapoint, thus parent represent this comparison. Isolation Forest improves upon assumption from

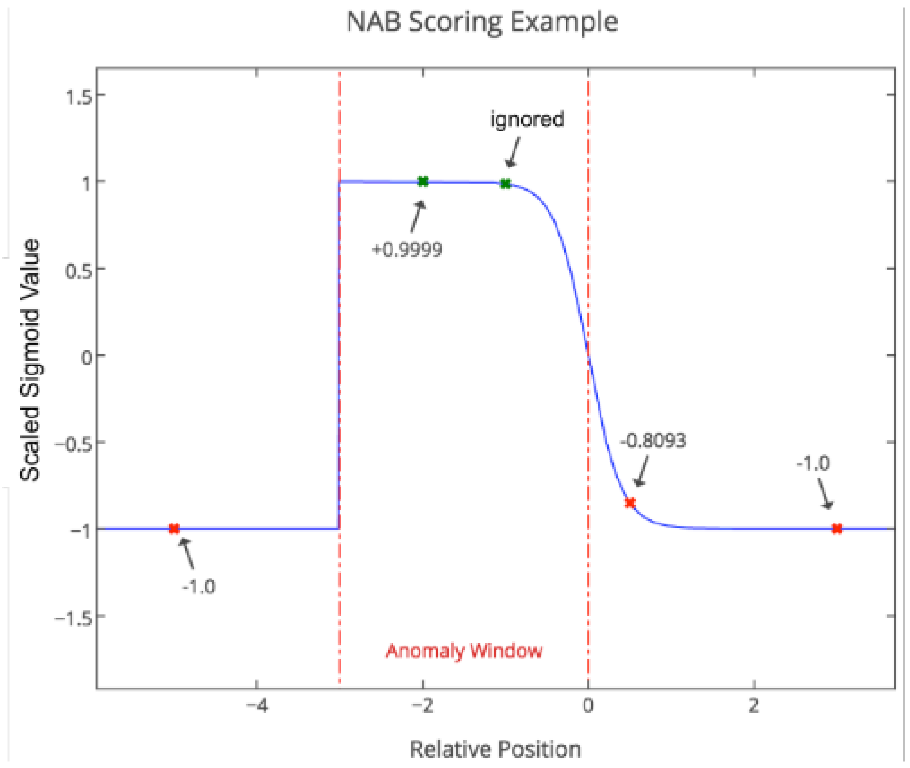


Figure 2.5: Numenta scoring function from: [19]

distance-based techniques: outliers will be in lesser depth than normal points fig. 2.6. Feature for splitting is randomly selected, as well as the partition point.[21]. The process is repeated for each data point. When a new data point appears, we insert it into n binary trees and calculate the mean depth according to each tree and result is anomaly score.

Categorical values are transformed into binary values. One way is to convert non-numerical values to numerical one $true = 1$ and $false = 0$. Problematic part is they do not say how exactly convert data like different browsers(multiple different strings) into numerical value. We can assume two variants - creating one-hot binary label, therefore adding much more features or create order and iterate from 0 to n, which is known from others field of machine learning.

2.1.4.1 Conclusion

The paper showed results with enterprise data-set ”CA RiskMinder user payroll access logs”, which is not publicly available. Results depends on selected features, but overall 98% TP and 50% accuracy was obtained.

Isolation forest is a scalable option, does not need anomalous instances in

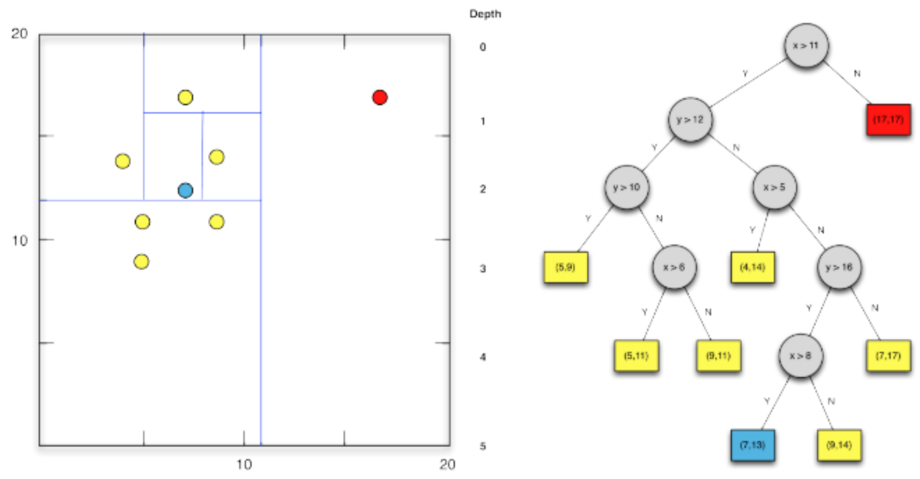


Figure 2.6: On the left we can see all data points (each of them having x and y axis) and right is one of the binary trees, where the red one is considered an outlier. Isolation forest from: [21]

the training set, but requires a threshold based on the tree-depth: which is independent from the data set.

2.1.5 C-LSTM

C-LSTM [7] created a novel idea for anomaly classification, although it is a supervised neural network, where it connects CNN together with LSTM networks.

2.1.5.1 Architecture

Architecture(shown at fig. 2.7) is created via CNN which takes as an input, a time-series window, then uses LSTM network *In the LSTM layers, we use memory cells rather than simple recurrent units to store and output temporal features of web traffic data. This capability makes it simpler to understand temporal relationships on a large time scale.*[7] After LSTM pass through is full dense Neural Network (NN) applied with softmax classifier, then trained and compared with KNN, SVM, MLP, Decision Tree, and Random Forests, where C-LSTM outperformed rest. Several combinations of CNN, LSTM, and Deep Neural Network were tried and all combination of were connected together and overall obtained better results than rest of models.

C-LSTM shows superiority comparing to other conventional machine learning methods and is able to extract patterns from high dimensional spatial-temporal information. But requires labels and only explanation of this model is to visualize intermediate values.

2.1.6 TadGAN

TadGan uses Generative Adversarial Networks (GAN) to perform anomaly detection. GAN become known for being for deepfake video and pictures and are evolving threats in social media and evidence manipulation.

Similarly to PCA and AE in sec 1.3.3.0.1 TadGan used assumption, where reducing time-series to lower dimension and then reconstruction involves higher reconstruction error, thus outliers are recognized by higher error.

Basic GAN architecture consists of two blocks Generator, which tries to generate artificial samples, which are input to Critic which tries to classify if the input is real data or artificially generated one. Part of GAN is fact that input to Critic is also mixed with the real input from the data set. The problematic part of GAN could be ineffective at learning the data-set distribution. They also could suffer from collapse problem, which is when Generator only outputs, which has fooled the Critic and not creating new ones.

2.1.6.1 Architecture

In TadGAN architecture(shown at fig. 2.8) are used two Generators ϵ and G. ϵ serves as encoder, maps to lower dimensions, and G as a decoder to the original dimension. Next are two Critics C_x and C_z. C_x tries to discriminate between artificially generated from G and real time-stamps inputs and C_z measures the performance of the mapping.

Having two critics and two generators have a twofold advantage, C_x can provide an outlier score because is trained to classify if data is fake or real.

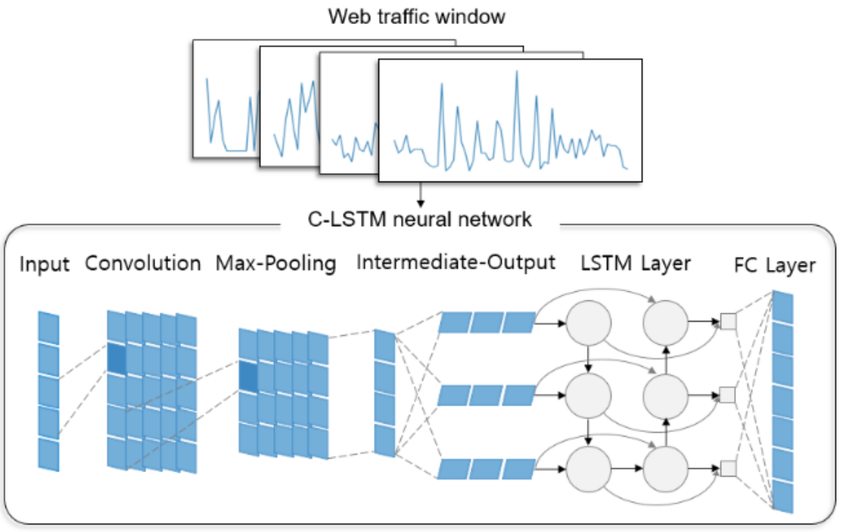


Figure 2.7: C-LSTM architecture from: [7]

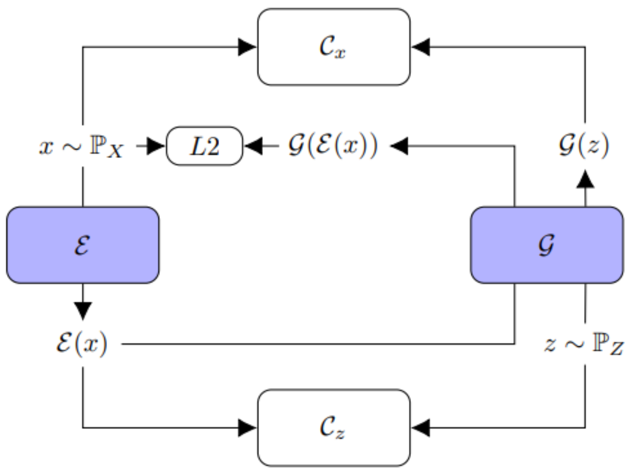


Figure 2.8: TadGAN architecture. Reprinted from: [22]

Second, the two Generators trained with cycle consistency loss allow us to encode and decode a time series sequence. The difference between the original sequence and the decoded sequence can be used as a second anomaly detection measure. [22]

To estimate anomaly score using reconstruction error was proposed:

point wise difference

$$s_t = |x_t - \hat{x}_t|$$

area difference

$$s_t = \frac{1}{2 * l} | \int_{t-l}^{t+l} x_t - \hat{x}_t |$$

is applied to window to measure similarity, good approach for windows where exist small difference over long time

Dynamic time warping

$$\min_W \frac{1}{K} \sqrt{\sum_{k=1}^{K} w_k}$$

also used for similarity between two windows. Involve using an matrix W, where each element is distance measure between two points w_{ij} , then aim is to find warp path W^* that defines minimum distance between two curves and also can define

Critic's anomaly score can be used directly, so applying kernel density estimation with maximum function gives a smothered score. Authors have shown that Critic outputs different scores to normal and anomalous windows.

	MSL	**SMAP**	**A1**	**A2**	**A3**	**A4**	**Art**	**AdEx**	**AWS**	**Traf**	**Tweets**	**Mean±SD**
TadGAN	0.623	0.704	0.8	0.867	0.685	0.6	0.8	0.8	0.644	0.486	0.609	0.700±0.123
LSTM	0.46	0.69	0.744	0.98	0.772	0.645	0.375	0.538	0.474	0.634	0.543	0.623±0.163
Arima	0.492	0.42	0.726	0.836	0.815	0.703	0.353	0.583	0.518	0.571	0.567	0.599±0.148
DeepAR	0.583	0.453	0.532	0.929	0.467	0.454	0.545	0.615	0.39	0.6	0.542	0.555±0.130
LSTM AE	0.507	0.672	0.608	0.871	0.248	0.163	0.545	0.571	0.764	0.552	0.542	0.549±0.193
HTM	0.412	0.557	0.588	0.662	0.325	0.287	0.455	0.519	0.571	0.474	0.526	0.489±0.108
Dense AE	0.507	0.7	0.472	0.294	0.074	0.09	0.444	0.267	0.64	0.333	0.057	0.353±0.212
MAD-GAN	0.111	0.128	0.37	0.439	0.589	0.464	0.324	0.297	0.273	0.412	0.444	0.35±0.137
MS	0.218	0.118	0.352	0.612	0.257	0.204	0.125	0.066	0.173	0.166	0.118	0.219±0.145

Table 2.1: F1 Scores on several benchmarks

.

A combination of reconstruction and Critic's scores is further used and benchmarks shown that using just Critic's score is unstable. Several datasets for benchmarking have been used and in a lot of cases, TadGan outperformed other baseline models as can be seen in Table table 2.1.

2.1.6.2 Conclusion

TadGan shows briliant results in comparision with other baseline models, however fails when used on synthetic datasets with point anomalies. Next results also shown that using single Critic's single output as anomaly score is not stable and better results were with area difference or dynamic time-warping approach.

2.1.7 Synthesis of anomaly detection approaches

Several approaches have been shown, comparing them together is complicated, since they are not compared in a scientific manner or using the same dataset and metrics in the original paper, but for further decisions in the next chapter, there is a need to do that.

EGADS is based on nine different features, does not show significant superiority compared to other models, which means it can be used in any time-series, but is limited to this feature set. C-LSTM, DeepAnt, and TadGAN outperform datasets in their papers, compared Extended Isolation Forests show only 50% accuracy (but high TP rate), however, C-LSTM is supervised and therefore not used for further comparison. DeepAnt states it is(CNN based) is not data hungry opposed LSTM, thus could be compared on TadGAN which is based on LSTM in both generators. Therefore for the next usage, we conclude that DeepAnt is an optimal choice.

2.2 Current IDS solutions

In previous section various approaches for anomaly detection have been shown. This section is dedicated to available tools closely related to attack detection or prevention by any means. Research on existing solutions could be potentially useful when developing algorithm in next chapter.

2.2.1 Cisco Secure Network Analytics

Cisco Secure Network Analytics (CSNA) (formerly Stealthwatch) is a complex tool protecting against compromising the network. From first sight it takes Network logs as data, however, the source of input is broad including Routers, Switches, Firewalls, Datacenter logs, VPN endpoints, Cisco Identity Services Engine, and many others with the possibility to incorporate external third tool. They divide their core function into three parts:

- Behavioral modeling, which collects data from each device on network and logs from other Cisco Modules
- Multilayer machine learning modeling
- Global Threat Intelligence

Multilayer machine learning modeling Machine Learning here is used in both settings, unsupervised for anomaly detection and supervised for classification. They claim to use over 70 unsupervised anomaly detectors, which are then used for ensembles that produce a single anomaly score. Three parts are involved:

Layer 1 - trust modeling and anomaly detection, used for filtering out 99% of undesirable data, and only anomalies are passed to next layers.

Layer 2 - Event classification, classifies only output from Layer 1, involving ML algorithms are Neyman-Pearson-based linear models, SVM and NN. They are classified into one of 100 threat categories. Cisco [23] claims they have 90% accuracy. Also in this layer is done entity modeling, basically creating a hypothesis of threat in the existing network.

Layer 3 - Relationship modeling, for taking view from a global perspective, if also other companies are possibly involved in such incident. They label them with confirmed 99-100% confidence because they have been seen before and detected, which is a unique and targeted attack.

One big advantage of CSNA is the fact it can detect malware in encrypted traffic, but with special hardware. *Secure Network Analytics has the ability to detect malware in encrypted traffic without any decryption - an industry first. Telemetry from the next-generation Cisco network as well as the Secure Network Analytics. On Flow Sensor produces two new data elements: the sequence of packet lengths and times and the initial data packet. The initial data packet is a treasure trove of metadata, because, remember, all encrypted*

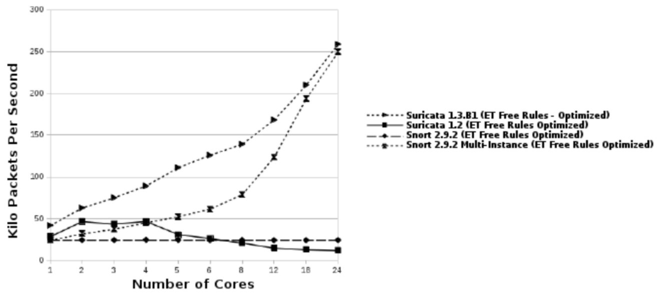


Figure 2.9: Snort-Suricata perfomance from: [24]

sessions start out unencrypted initially. Cisco's unique Application-Specific Integrated Circuit (ASIC) architecture provides the ability to extract these data elements without slowing down the data network[23]

2.2.2 Snort and Suricata

Snort is one of the oldest IDS, become developed in 1998 as open source IDS. Snort is based on recognizing patterns in-network, therefore he cannot detect any novelties in attacks, unlike anomalous-based ones. There are commercially IDS based on Snort, where updates receive immediately only those who have subscriptions and after 30 days everyone who is registered. Suricata is IDS from 2009 with a similar approach as Snort, however is available only as a community version.

One advantage [24] of Suricata over Snort is multithreading support, where outperforms (see fig. 2.9) Snort in until version 3(January 2021) was available only in multi-instance setting, but Suricata gets higher throughput even on a single core. With recent version 3 lot of changes, Snort gets multithreading support, but no valid performance comparison was found.

Qualitative comparison for signature-based IDS mainly depends on the database of rules and signatures. But in [25] evaluated Snort and Suricata in terms of memory, throughput, and accuracy, where build virtualized network and emulate their intrusion data with Python script, which send there 54 attacks in 9 categories. *Overall, Snort had 16 false negatives, 10 false positives, 1720 "gray positives" (including 1640 "evasion techniques"), and 44 true positives. Suricata had 12 false negatives, 8 false positives, 1449 gray positives (including 1275 evasion techniques), and 81 true positives. The evasion techniques involved five port scans, so the amount of traffic generated for this test was significantly more than that of any other test. The recall fraction was 0.73 for Snort and 0.87 for Suricata in our third experiment. We can calculate two kinds of precision, one excluding the grey positives as false positives and one including them. We got 0.81 for Snort and 0.91 for Suricata excluding them, and 0.036 and 0.052 including them.* [25] Grey positives refer to alerts for different attacks than currently operated on the network.

2.2.2.1 Conclusion

Altough Snort and Suricata show differences in architecture, rules, and differences in some features, for example, automatic traffic detection on uncommon ports, according to [24] and [25] seem to be very close to each other in both Qualitative and Quantitative comparison, however, these systems are still developing and for obtaining precise results, we would need to compare up-to-date version.

2.2.3 Zeek

Zeek is a network traffic analyzer but can be also used as IDS for investigating anomalous activity on the network. Zeek collects multiple types of logs in the network on different levels from layer 3 of the OSI model up to application-level including HTTP requests, MIME types, DNS requests, and responses which are then written into JSON log file. Zeek also provides many built-in functions for analysis like extracting files from HTTP sessions, detecting malware by interfacing to external registries, reporting vulnerable versions of software seen on the network, identifying popular web applications, detecting SSH brute-forcing, validating SSL certificate chains.

2.2.3.1 Zeek's architecture

Zeek's architecture can be seen onfig. 2.10, basically, network packets are displaced into events, which are passed through special Zeek's policy scripting language: which captures the semantic of events, where can be written domain specific policies what to further if any occurrence of events happened, therefore Zeek is not typical signature-based IDS but a superset of IDS.

Custom policies can even trigger external programs, Zeek can take proactive measures against potential attackers.

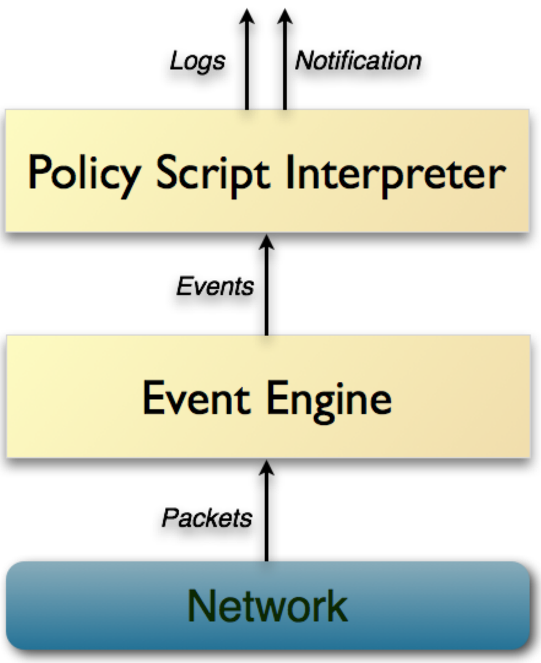


Figure 2.10: Zeek's architeture available: `https://docs.zeek.org/en/master/about.html`

2.2.4 Network Measurements Analysis - NEMEA

Network Measurements Analysis (NEMEA) [26] is CESNET's open-source detection system for network traffic analysis. NEMEA is focused on stream-wise and flow-based detection and built in modular way.

NEMEA architecture(shown at 2.11) is in modular fashion, each module communicates via communication interfaces. Each module is run as independent process, modules are connected together usually in tree manner in logical order based on various purposes of module such as:

Processing modules: Flow Counter, Logger, Anonymizer, Entropy, Email-Reporter, IPFIXcol and many others. Used for processing information about the flow.

Analytical modules: PCA, Astute (looks for changes between two time windows), IP Spoofing Detector, Blacklist Filters, Botnet Detector, Host-Stats(computes statistical measurments about flows) and others.

input modules, alert modules, detector modules. Modules are possible shut-down or even hot-swap depending on actual situation and can be written in any programming language.

Two principles differs from other solutions: firstly each input is processed separately unless module rules are explicitly overridden and no data are stored in hard drive. This leads to fast data processing and fast analysis. Input into NEMEA is usually nfdump, csv or pcap or UniRec(data format) which converts IPFIX fields, processed by IPFIX collector, to NEMEA.

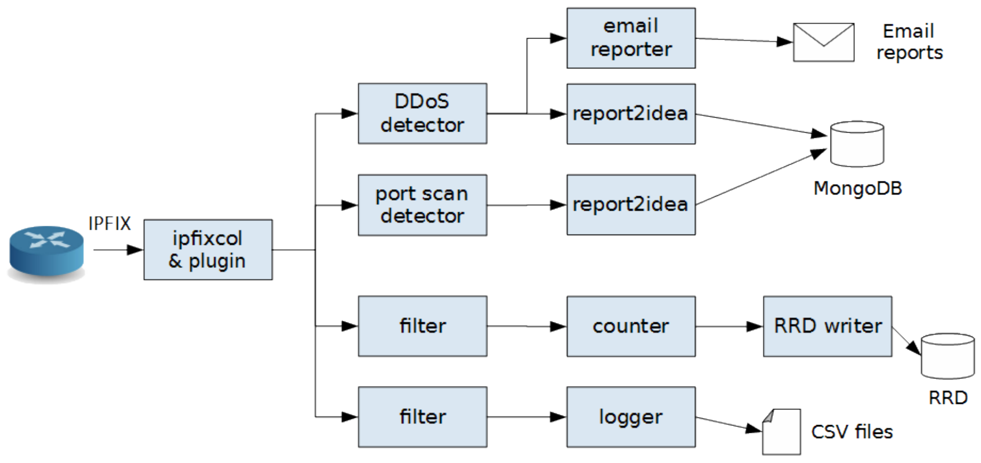


Figure 2.11: NEMEA's architeture available: `https://nemea.liberouter.org/`

2.3 Conclusion

Several approaches from the last few years have been shown, interestingly use of CNN shown in section 2.1.1 for forecasting value of next timestamp or C-LSTM architecture. Also, a new way of evaluation section 2.1.3 has been shown by incorporating a time window. TadGan section 2.1.6 obtained promising results, where worse performance has been observed on artificial dataset with point anomalies.

Few IDS or similar tools were also shown in this chapter. Performance and partial feature comparison was provided in section 2.2.2, where they stand closely to each other, but in compared version the Surricata was better option due to higher throughput and multithreading.

CSNA shows how complex are state-of-the-art IDS, where using over 70 machine learning methods with ensembles are connected across different companies for tackling with global threats. Next advantage of this approach are datasets(of attacks) for perfoming supervised machine learning.

NEMEA and ZEEK are tools interested more in network analyzer than pure IDS system, however in case of NEMEA the difference is thin. Both tools analyze network, provides tools for alerting, enables writing policies or modules. Architecture seems perform better on NEMEA side, and described performance seems to be ready to challenge large scale networks.

Research on this will be used in next chapters for designing and implementing algorithm for detecting bruteforce attacks.

Chapter 3

Analysis and Evaluation

“You get hit by Kafka, in the morning woke up broke.”

– J.Cole, free translation

The previous chapter showed IDS overview with the current state-of-the-art techniques for anomaly detection, this knowledge serves as a basis for choosing a solution adequate for the environment at Showmax.

3.1 Analysis of Showmax data

Showmax’s data is spread on over hundreds servers and instances on Amazon’s AWS platform therefore obtaining IP flows would result in an increased amount of computer power and also needs high amount of memory on each server and problems with synchronizing over several locations.

3.1.1 Available data sources

Showmax stores network data in different places and within the different levels of views, first storage in is ELK stack, which is ElasticSearch with Kibana interface, but data there are stored just for the last 30 days and then are moved for backups into compressed-gzipped files. The contained information is divided into:

- Logs on a lower level
- Events on a high programming level
- Alerts
- Proxy communication between servers

Most useful category appeared to be Logs, firstly because they contain much more datapoints than events, which sometimes could contain interesting attributes of data. Logs particularly contain information about possible bruteforce attacks within filtered-out out unsuccessful attempts on login:

- http_code: returned http code for request
- service
- normalized_url_simple: shortened url
- http_headers.rx.showmax-int-ua-os: user agent operation system
- http_headers.rx.showmax-int-ua-class: category of user device
- http_headers.rx.showmax-int-platform: used platform
- http_headers.rx.user-agent: user agent string
- client_ip
- client_geopoint: json containing latitude and longitude
- country - country code
- ip address

Especially interesting are logs with HTTP error code 403, which shows unauthorized access and normalized URL simple to adequate sign-in URL. For obtaining these data from ELK stack elasticsearchdsl `https://elasticsearch-dsl.readthedocs.io/en/latest/` was used.

3.1.2 Limitation of dataset

The existing dataset consists of logs about activity coming from various IP addresses. It is, however, unbalanced, due to existing services, that count these unauthorized requests and blocks certain IP address if the number of received requests exceeds the threshold, therefore once is IP address blocked there won't be next datapoint in next time period, so obtained data set is distorted. Obtaining non-distorted data was not possible due to the security policy in the company. No labels of brute force attacks were provided, the only option was to look for banned IP addresses, but using them as labels would yield machine learning techniques trained in detecting the threshold, rather than anomalies in general, which would not be the desired goal.

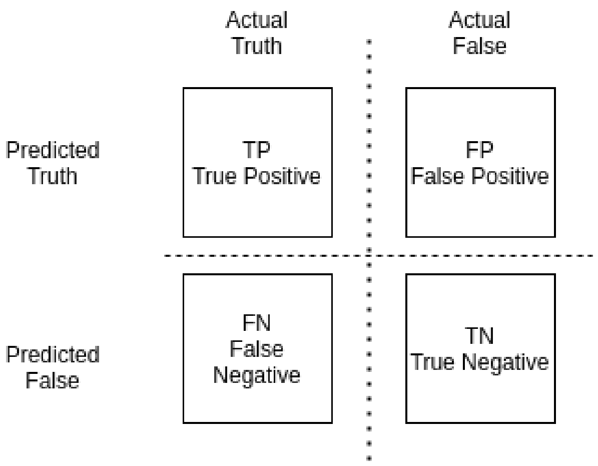


Figure 3.1: Confussion matrix

3.1.3 Feature engineering

Data points were aggregated together, depending on features, within time-windows. Firstly we needed to choose the time period, in similar approaches we were able to find value varying from few seconds to one minute. One minute interval was chosen and the following features were used:

- timestamp as index for Time-series
- Number of unique user agent strings
- Number of unauthorized requests
- Number of unique ip addresses
- Number of unique geopoint locations
- Modest ip address within interval

3.1.4 Evaluation and model choosing

Since no labels were provided as ground-truth, this leads naturally to an unsupervised setting for anomaly detection. According to [5] the only way to evaluate unsupervised anomaly detection is to compare it within the ensemble of other methods. Thus we consider the output of the ensemble to be the ground truth and we can compare the following metrics for binary classification, which are based on the Confusion Matrix shown at fig. 3.1.

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

$$Recall = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP}$$

However metrics above are not very suitable for heavily imbalanced datasets such as anomaly detection, thus F1 measure has been proposed for this purpose:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

We decided to use a voting ensemble, which could be realized in the hard and soft settings. Hard setting simply treats each model equally, whether soft setting uses:

$$argmax \sum_i p_i$$

for decision.

We have chosen these models for anomaly detection:

- Deep-Ant from section 2.1.1, as proposed optimal model from previous chapter, should not be data hungry.
- Isolation Forests, where it can be connected to Subspace analysis section 1.2.2.1. One can limit number of features for fitting as well as proportion of data.
- OC-SVM from section 1.3.3 as representative of linear models, could provide insight into dependencies between features.
- Local Outlier Factor (LOF) as representative from Proximity method section 1.3.2 , this method mentioned in EGADS framework.
- EllipticEnvelope, as statistical method, which assumes data are Gaussian distributed.

3.2 Dataset

Dataset for training and evaluation data were obtained and stored in parquet files using pandas. The collected dataset was containing data from 1.1.2021 to 18.3.2021 and can be seen at fig. 3.2. Such a time range was selected because it was considered as a good compromise between model accuracy and computation complexity: taking a longer range would result in a longer period of processing, but from observation, we could say that the only feature which vary across different months is the number of unauthorized requests (doc_count). This could be due to it being a combined metric: last month of data has been taken from Kibana, while the rest from backup files. The rest of the features behaves in consistent manner, in the figure we can see some empty discords in last month, probably indicating missing information in the database. There is clearly day-seasonality visible, where more people are using service during a certain period. Several spikes are also visible: these can be related either to brute force attacks, or higher usage of service taking place, for example, while streaming sport events.

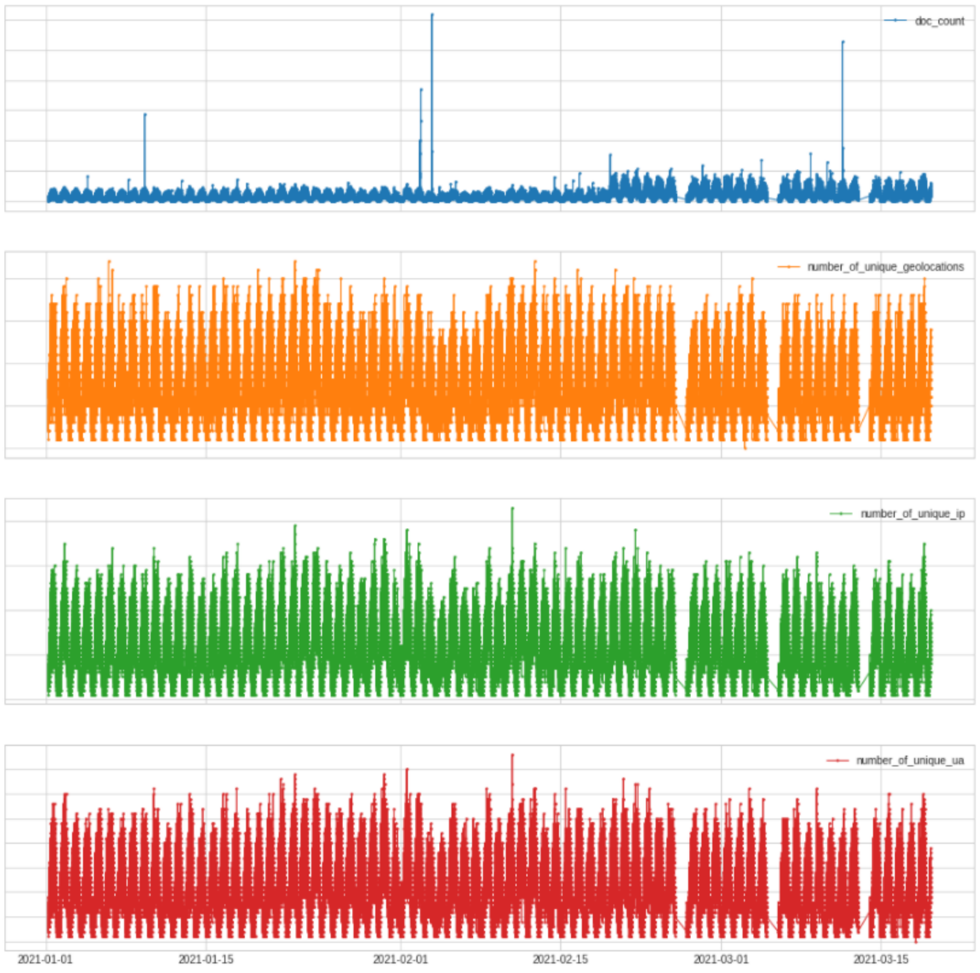


Figure 3.2: Dataset visualization. Dataset visualization. Vertical axis has the values removed due to confidentiality reasons.

3.3 Machine Learning

For Isolation Forests, OC-SVM, Elliptic Envelope and LOF the scikit library has been used `https://scikit-learn.org/stable/`. VotingAnomalyEnsemble (shown at fig. 3.3) was developed as a class, which takes as input list of models and then extracts anomaly scores and provides hard majority voting.

DeepAnt was selected as fifth model, firstly requires less datapoints compared to other deep learning approach, shows good performance on several datasets even with NAB scoring function.

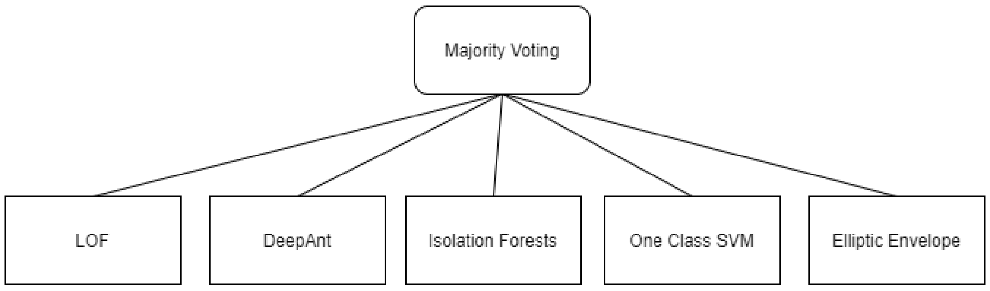


Figure 3.3: Proposed ensemble

3.3.1 Evaluation within Voting Ensemble

F1, balanced accuracy, recall, and precision were used for evaluation, output from VotingEnsemble was used as proxy for the ground truth, then a particular metric was computed against each model. The table is shown at table 3.1, we can see that no single model leads on each metric, mostly differs results in recall metric, especially OC-SVM obtained only 0.43 recall, where most gets over 0.9. Most stable results across metrics has EllipticEnvelope. All hyperparameters were heuristically selected, or left to the default value.

	f1	balanced accuracy	recall	precision
DA	0.994847	0.521352	**0.999434**	0.990302
IF	**0.996161**	0.684469	0.998746	0.99359
OC	0.604851	0.71677	0.433539	**1**
EE	0.952418	**0.953388**	0.90918	0.999973
LOF	0.959637	0.780235	0.925854	0.995979

Table 3.1: Evaluation within to VotingEnsemble

3.3.2 Evaluation within known banned IP addresses

We were able to obtain banned IP addresses within the period and decided to compare each single model against them. These banned IPs are shown at fig. 3.4, where banned IPs are the red vertical lines, we can see that banned IPs are related to point anomalies, in many cases, which is coherent with anomaly to attacks taxonomy shown at fig. 1.3.

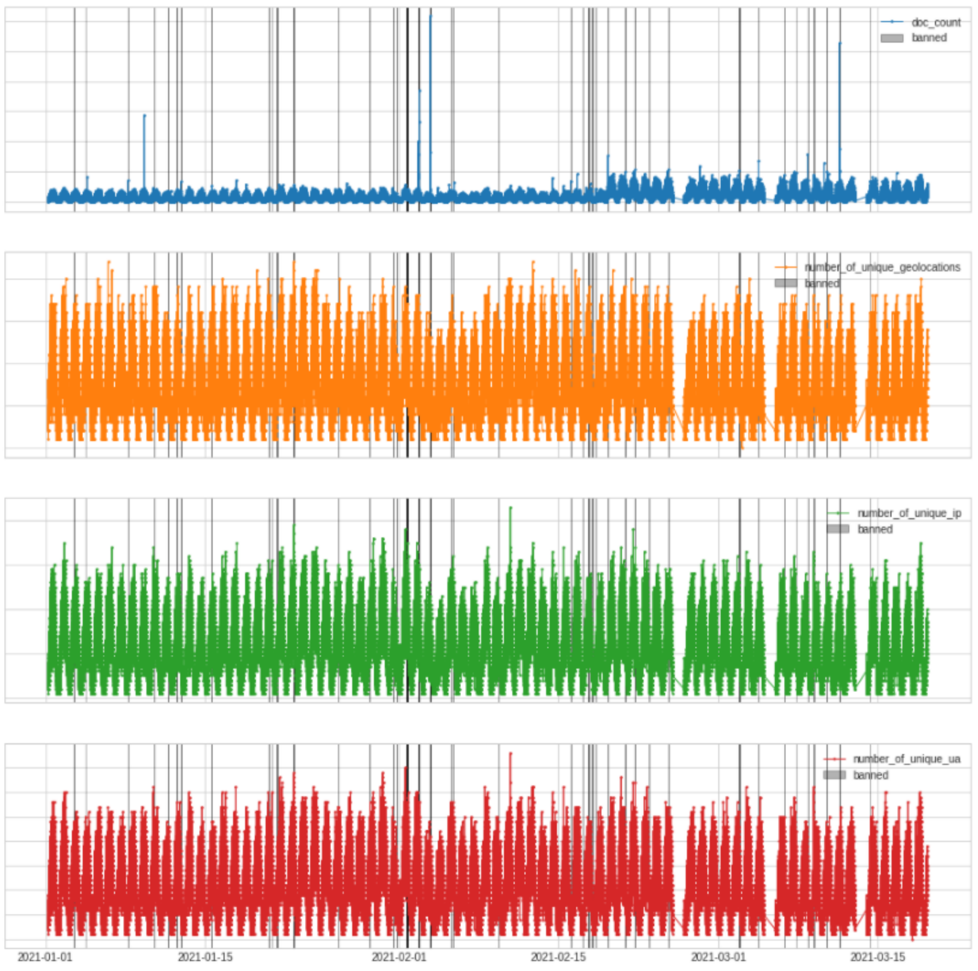


Figure 3.4: Banned IPs visualization. Red vertical lines shows ban at the moment

3.3.2.1 Model parameters sensitivity

Each model incorporates several hyperparameters since no ground truth is provided, hyperparameter optimization is a difficult task because we don't know how good we made the selection, thus it is rely on the knowledge and desired outcome of an expert to choose them carefully and according to the domain. The subset of relevant hyperparameters can be seen at table 3.2

We decide to perform sensitivity to parameters, to see if the metric changes rapidly on different parameters, such observation would mean that results are parameter dependent and not data dependent. For this evaluation, we took banned IP addresses from a current banning scriptsection 3.1.2. We took them as ground truth, to see if anomaly detection can replace current solution, then output from anomaly detection models would result in TP or FN. A similar

DeepAnt	number_epochs	threshold		
IsolationForests	contamination	n_estimators	max_samples	max_features
EllipticEnvelope	contamination	assume_centered	support_fraction	
OC-SVM	kernel	degree	gamma	
LOF	contamination	n_neighbors	algorithm	

Table 3.2: Hyper parameters of each model

approach was shown at section 2.1.3. Ten minutes interval within banned IP was chosen for deciding if anomaly detection was TP or FN. GridSearchCV class from scikit was used for this, but since the cross-validation option was disabled by unsupervised modeling of our problem.

Results for each model are shown at fig. 3.5. Isolation Forest mostly depends on with contamination parameter, which is in other words threshold for outliers. Interestingly also maximal features peaks at 50% (2 features) which shows how important is subspace analysis (described at section 1.2.2.1), sometimes the outliers are hidden in higher dimensionality, but are visible in lower. Max samples and the number of estimators do not affect the recall-like metric, bit higher score with lower number of estimators are probably caused by in-equally splitted feature set, with high number of estimators this does not hold.

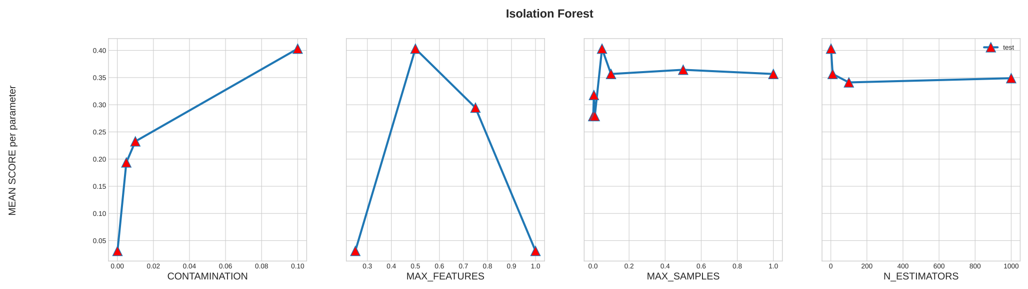
OC-SVM clearly performs better with gamma equal to auto rather than set to scale option. Also linear and polynomial kernel performs poorly although of not that much high dimensionality. Linear kernel could be expected since we are not having linear data. Degree(for polynomial kernel only) does not affect the results at all.

Elliptic Envelope shows very robust results, where the only affecting parameter was contamination, thus there is hypothesis that dataset is Gaussian distributed.

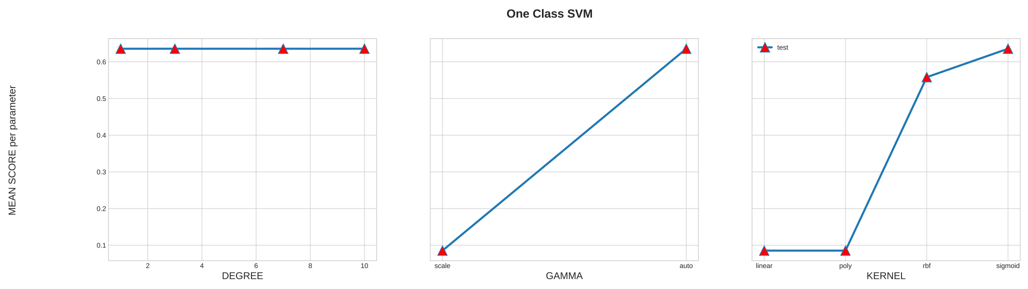
And DeepAnt from section 2.1.1 shows the lower correlation between banned IPs from the script and detected anomalies. This could be caused by several reasons:

1. Low number of training samples
2. Low dimensionality, hence kernel size in the convolutional layer cannot be modified
3. Feature rescaling before fitting, although is usually approach for deep learning

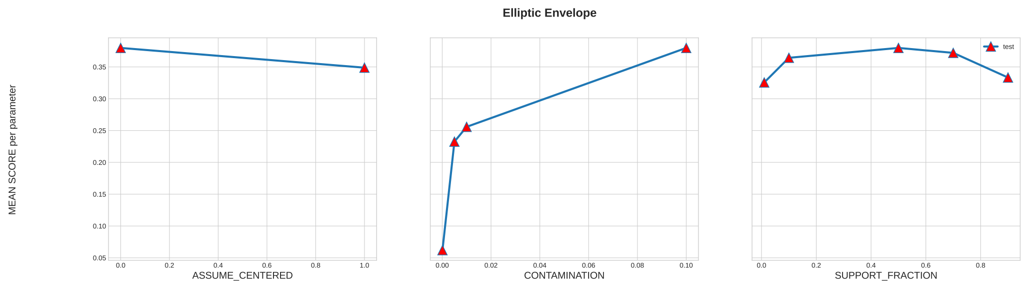
Since all models were most prone for contamination or other named parameter which corresponds to threshold(gamma in OC-SVM) we conclude that finding these potential brute force attacks is mostly data-dependent which means we can focus on desired outcome(number of expected anomalies in dataset) the detected outliers are expected to be brute force attacks.



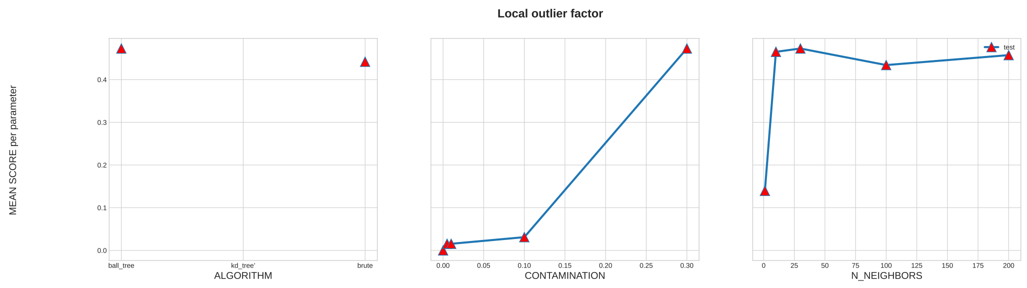
(a) Isolation Forest, contamination parameter sets the proportion of outliers in dataset, should be in range [0,0.5], but we used [0,0.3] for having less FP and result shows linear dependency, so for next models we focused on range [0,0.1]. Max_features defines number of features used in each tree. Max_samples tells proportion of dataset used for tree construction. N_estimators defines number of trees used for computing outlier score.



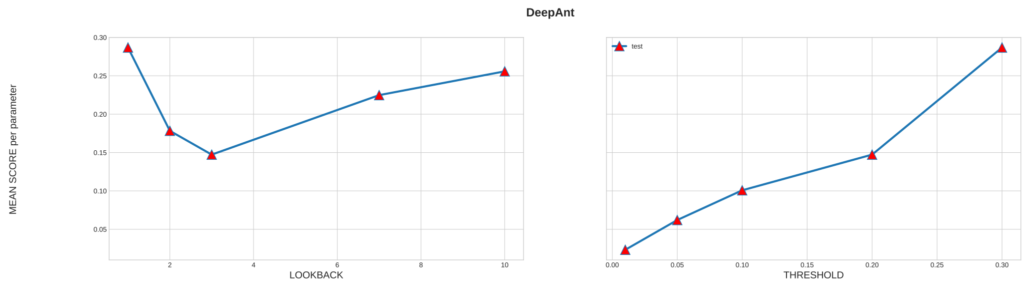
(b) One Class SVM, degree is parameter just for polynomial model, gamma is kernel coeficient for poly,rbf and sigmoid kernel, where if set to scale than $\frac{1}{((n\_features*X.var())}$ is used for value and auto uses $\frac{n}{features}$.



(c) Elliptic Envelope, assume_centered for setting if data are closely to zero but not exactly, if False than it is covariance recomputed without any treatment. Contamination parameter defines proportion of outliers in data. Support_fraction [0,1] as proportion of points to be included in the support of the raw Minimum Covariance Determinant estimate.



(d) Local Outlier Factor, algorithm chooses the way how proximity will be computed. Contamination parameter defines proportion of outliers in data. N_neighbors as number of neighbours



(e) DeepAnt. Lookback defines number of previous timestamps included for forecasting, threshold defines how outliers will be selected.

Figure 3.5: Sensitivity Analysis

3.4 Results, scalability and future work

All results in this chapter were more than promising, having the ”recall-like” score in higher than 10% was desired internal result. In case of bruteforce attacks, we are mainly aware of false negatives i.e not detected attacks, however false positives are important too, banning innocent ip address results in impossible access for many users, because of NAT. Absence of having groundtruth labels does not provide confidence in the obtained results, but we can conclude our approach works even on such distorted data.

Scalability comes play important role, it was one of the reasons of selecting highly aggregated data from central database, independently how Showmax infrastructure will grow, this approach could remain untouched. But caveat in aggregation is to choose proper time window and brings limitation of possible banning single ip address in one time window.

3.4.1 Future work

Although the results are promising, they can be possibly further enhanced, using meta machine learning, trying to increase speed time of models via supervised way. Using proposed VotingEnsemble for generating data set with labels, then train supervised model on such network should yields in lower predicting time, thus selecting higher period for time window.

Conclusions

Protecting the network from attacks has been necessary from the beginning of the Internet itself, and every company which operates on the internet needs to address the issue of cybersecurity. In case of Showmax, the importance of assuring that malicious activity in network is stopped, stems from the legal reasons. Showmax is a video-on-demand service providing access to movies, series, and sport events that are often produced by third parties. As such, assuring appropriate security measures is a contractual obligation to the third parties. Brute force attacks which try to get unauthorized access to the Showmax site can result in stealing private information about users, breaking trust with third party studios and stealing intellectual property.

This thesis is dedicated to the automatic detection of malicious activity within the internal network of Showmax. A preexisting implementation used in Showmax has been done in a threshold-based approach: if too many requests are received, the given IP address is blocked. Methods proposed in this thesis aim to extend this idea with the use of unsupervised anomaly detection.

The thesis reviews state of the art of malicious activity detection yielding in proposed unsupervised models, combined together in ensemble method of these models:

- DeepAnt, state-of-the-art model, based on deep learning with CNN
- Isolation Forest
- OC-SVM linear based model
- EllipticEnvelope a statistical model based on Gaussian distribution
- LOF a density-based model

Unsupervised detection of brute force attacks has been performed on a dataset corresponding to a typical traffic occurring on Showmax's onboarding pages. Unsupervised settings bring the absence of need for the label of data, which is especially useful when reliable labels cannot be obtained, but with drawbacks on the reliability of results. We propose a solution based on aggregated features, arranged in time-windows. The proposed solution is scalable due to the use of host aggregated features. Therefore, any additional work on improving accuracy can be done in an infrastructure-agnostic way. Moreover, the recall-like metric of the whole ensemble reaching up 60% in evaluation proves that anomaly detection based on aggregated features can detect and help to prevent brute-force attacks.

Results have shown the proposed algorithm is capable of detecting brute force attacks via aggregated features. The proposed algorithm can be implemented after necessary tweaks in the enterprise environment. Its evaluation was tested on banned IP addresses via the baseline models. The proposed solution has been developed with the limitations stemming from the possible testing options: shadow infrastructure does not exist and brute force attacks would create a bad user experience for customers, since using the aggregated features creating a data set in a virtual environment would not bring additional value. A possible limitation in use are lower response time, but this could be enhanced via a shorter aggregation time window.

Bibliography

[1] Gladstein, B.; Murphy, R.; et al. Healthcare Cyber Heists in 2019. Technical report June, VMware Carbon Black, 2019. Available from: `https://www.carbonblack.com/wp-content/uploads/2019/06/carbon-black-healthcare-cyber-heists-in-2019.pdf`

[2] Malwarebytes. CYBERCRIME TACTICS AND TECHNIQUES: Ransomware Retrospective. Technical report, Malwarebytes, 2019. Available from: `https://resources.malwarebytes.com/files/2019/08/CTNT-2019-Ransomware_August_FINAL.pdf`

[3] Leder, F.; Werner, T.; et al. Proactive Botnet Countermeasures An Offensive Approach. Technical report, Institute of Computer Science IV, University of Bonn. Available from: `https://ccdcoe.org/uploads/2018/10/15_LEDER_Proactive_Coutnermeasures.pdf`

[4] Orair, G. H.; Teixeira, C. H.; et al. Distance-based outlier detection: Consolidation and renewed bearing. *Proceedings of the VLDB Endowment*, volume 3, no. 2, 2010: pp. 1469–1480, ISSN 21508097, doi: 10.14778/1920841.1921021.

[5] Aggarwal, C. C. *Outlier Analysis*. 2017, ISBN 9783319475776, 81–98 pp., doi:10.1016/b978-012724955-1/50180-7.

[6] Aggarwal, C. C.; Yu, P. S. Outlier detection for high dimensional data. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, volume 30, no. 2, 2001: pp. 37–46, ISSN 01635808, doi: 10.1145/376284.375668.

[7] Kim, T. Y.; Cho, S. B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications*, volume 106, 2018: pp. 66–76, ISSN 09574174, doi:10.1016/j.eswa.2018.04.004. Available from: `https://doi.org/10.1016/j.eswa.2018.04.004`

[8] Prasad, N. R.; Almanza-Garcia, S.; et al. Anomaly detection. *Computers, Materials and Continua*, volume 14, no. 1, 2009: pp. 1–22, ISSN 15462218, doi:10.1145/1541880.1541882. Available from: `https://doi.org/10.1145/1541880.1541882`

[9] Hansman, S. *A taxonomy of network and computer attack methodologies.* Dissertation thesis, University of Canterbury, 2003. Available from: `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:A+Taxonomy+of+Network+and+Computer+Attack+Methodologies`

[10] Hutchins, E.; Cloppert, M.; et al. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *6th International Conference on Information Warfare and Security, ICIW 2011*, , no. July 2005, 2011: pp. 113–125.

[11] Aggarwal, C. C.; Sathe, S. Theoretical Foundations and Algorithms for Outlier Ensembles. *ACM SIGKDD Explorations Newsletter*, volume 17, no. 1, 2015: pp. 24–47, ISSN 1931-0145, doi:10.1145/2830544.2830549.

[12] Müller, E.; Assent, I.; et al. Outlier ranking via subspace analysis in multiple views of the data. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2012, ISBN 9780769549057, ISSN 15504786, pp. 529–538, doi:10.1109/ICDM.2012.112.

[13] Johnson, T.; Kwok, I.; et al. Fast Computation of 2-Dimensional Depth Contours. In *KDD*, 1998, pp. 224–228. Available from: `https://www.aaai.org/Papers/KDD/1998/KDD98-038.pdf`

[14] Ringberg, H.; Soule, A.; et al. Sensitivity of PCA for traffic anomaly detection. In *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems - SIGMETRICS '07*, New York, New York, USA: Association for Computing Machinery (ACM), 2007, p. 109, doi:10.1145/1254882.1254895. Available from: `http://portal.acm.org/citation.cfm?doid=1254882.1254895`

[15] Alla, S.; Adari, S. K. *Beginning Anomaly Detection Using Python-Based Deep Learning.* 2019, ISBN 9781484251768, doi:10.1007/978-1-4842-5177-5.

[16] Applied Deep Learning - Part 3: Autoencoders — by Arden Dertat — Towards Data Science. Available from: `https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798`

[17] Ensemble Learning to Improve Machine Learning Results — by Vadim Smolyakov — Cube Dev. Available from: `https://blog.statsbot.co/ensemble-learning-d1dcd548e936?gi=4ce114a9fb40`

[18] Munir, M.; Siddiqui, S. A.; et al. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access*, volume 7, 2019: pp. 1991–2005, ISSN 21693536, doi:10.1109/ACCESS.2018.2886457.

[19] Lavin, A.; Ahmad, S. Evaluating real-time anomaly detection algorithms - The numenta anomaly benchmark. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, 2015: pp. 38–44, doi:10.1109/ICMLA.2015.141. Available from: https://github.com/numenta/NAB

[20] Laptev, N.; Amizadeh, S.; et al. Generic and scalable framework for automated time-series anomaly detection. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 2015-Augus, 2015: pp. 1939–1947, doi:10.1145/2783258.2788611. Available from: https://github.com/yahoo/egads

[21] Sun, L.; Versteeg, S.; et al. Detecting Anomalous User Behavior Using an Extended Isolation Forest Algorithm: An Enterprise Case Study. *arXiv*, volume abs/1609.0, 9 2016. Available from: http://arxiv.org/abs/1609.06676

[22] Geiger, A.; Liu, D.; et al. TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks Alfredo Cuesta-Infante. Technical report. Available from: https://github.com/signals-dev/Orion

[23] Challenges, B.; Overview, S. Cisco Secure Network: pp. 1–5. Available from: https://www.cisco.com/c/en/us/products/collateral/security/stealthwatch/white-paper-c11-740605.html

[24] White, J. S.; Fitzsimmons, T.; et al. Quantitative analysis of intrusion detection systems: Snort and Suricata. *Cyber Sensing 2013*, volume 8757, no. May 2014, 2013: p. 875704, ISSN 0277786X, doi:10.1117/12.2015616.

[25] Albin, E.; Rowe, N. C. A realistic experimental comparison of the Suricata and Snort intrusion-detection systems. In *Proceedings - 26th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2012*, 2012, ISBN 9780769546520, pp. 122–127, doi:10.1109/WAINA.2012.29.

[26] Cejka, T.; Bartos, V.; et al. NEMEA: A Framework for Network Traffic Analysis. In *12th International Conference on Network and Service Management (CNSM 2016)*, CESNET, 2016, doi:10.1109/CNSM.2016.7818417. Available from: http://dx.doi.org/10.1109/CNSM.2016.7818417

Appendix A

Acronyms

AE	Autoencoder.
AUC	Area Under Curve.
CNN	Convolutional Neural Networks.
CSNA	Cisco Secure Network Analytics.
CVE	Common Vulnerabilities and Exposures.
FP	False Positive.
FS	File System.
GAN	Generative Adversarial Networks.
HIDS	Host Based Intrusion Detection System.
IDS	Intrusion Detection System.
IPS	Intrusion Prevention System.
LOF	Local Outlier Factor.
NAB	Numenta Anomaly Benchmark.
NEMEA	Network Measurements Analysis.
NIC	Network Interface Card.
NIDS	Network Intrusion Detection System.
NN	Neural Network.

OC-SVM	One Class Support Vector Machine.
PCA	Principal Component Analysis.
RBM	Restricted Boltzmann Machine.
SVM	Support Vector Machine.

Appendix **B**

Contents of enclosed media

```
readme.txt ............ the file with contents description
src ..................... the directory of source codes
  dip.zip ............... implementation sources
  thesis ................ the directory of LaTeX source codes of the thesis
text .................... the thesis text directory
  thesis.pdf ............ the thesis text in PDF format
```