Network Anomaly Detection by Means of Spectral Analysis

Diploma Thesis

Bc. Peter Boráros

Study programme: Open Informatics
Specialisation: Artificial Intelligence
Thesis supervisor: Ing. Martin Rehák, PhD.

Prague, 2013
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Peter Boráros
Study programme: Open Informatics
Specialisation: Artificial Intelligence
Title of Diploma Thesis: Network Anomaly Detection by Means of Spectral Analysis

Guidelines:

The goal of this thesis is to design and implement an anomaly detection method based on spectral analysis.
In particular, use the proposed method to distinguish normal and anomalous network traffic and evaluate the properties of this method on real network traffic data.

Bibliography/Sources: Will be provided by the supervisor.

Diploma Thesis Supervisor: Ing. Martin Rehák, Ph.D.
Valid until: the end of the winter semester of academic year 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, September 13, 2012
ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Peter Boráros

Studijní program: Otevřená informatika (magisterský)

Obor: Umělá inteligence

Název tématu: Detekce anomálií v počítačové síti pomocí spektrální analýzy

Pokyny pro vypracování:

Navrhněte a implementujte metodu, ve které využijete spektrální analýzu k detekci nežádoucího chování v počítačové síti, zejména k rozlišení normálního a anomálního provozu.
Vlastnosti této metody experimentálně vyhodnoťte na datech z reálného síťového provozu.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Martin Rehák, Ph.D.

Platnost zadání: do konce zimního semestru 2013/2014

prof. Ing. Vladimír Mařík, DrSc. vedoucí katedry

prof. Ing. Pavel Ripka, CSc. děkan

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 3.1.2013.  Bc. Peter Boráros
Acknowledgement

First I would like to thank the Karel Bartoš and Martin Rehák for support. Next, I would like to thank my family and Ludmila for patience.
Název práce: Detekce anomálií v počítačové síti pomocí spektrální analýzy

Autor: Bc. Peter Boráros (e-mail: peter.boraros@pborky.sk)

Katedra (ústav): Katedra kybernetiky

Vedoucí diplomové práce: Ing. Martin Rehák, PhD. (e-mail: rehakmar@fel.cvut.cz)


Klíčová slova: detekce anomálií, spektrální analýza, bezpečnost počítačových sítí

Title: Network Anomaly Detection by Means of Spectral Analysis

Author: Bc. Peter Boráros (e-mail: peter.boraros@pborky.sk)

Department: Department of Cybernetics

Thesis Supervisor: Ing. Martin Rehák, PhD. (e-mail: rehakmar@fel.cvut.cz)

Abstract: In the present work we study an statistical analysis of frequency spectrum as an anomaly detection technique. Our goal is to develop method that will able to detect malicious behavior in computer network. We are focused on detection of tunneled connections over HTTP protocol in order to circumvent restrictions by malicious agents.

Keywords: Anomaly Detection, Spectral Analysis, Network Security
# Contents

Abstract 11

1. Introduction 15
   1.1. Related Work ............................................. 16
   1.2. Our Contribution ......................................... 18
   1.3. Organization ............................................. 18

2. Theoretical Introduction 19
   2.1. Anomaly Detection ......................................... 19
   2.2. Computer Network Security ................................. 25

3. Proposed Method 27
   3.1. Data Collection ............................................ 28
   3.2. Feature Creation .......................................... 29
   3.3. Model Estimation and Anomaly Detection .................. 33
   3.4. Parameter Selection and Empirical Evaluation .......... 39

4. Experiments 43
   4.1. Data gathering ............................................ 43
   4.2. Implementation ........................................... 45

5. Assessment and Conclusion 49
   5.1. Results .................................................. 49
   5.2. Conclusion ............................................... 50

Bibliography 55

Appendices 1

A. PyNfSA Documentation 1

B. Results XV
Chapter 1.

Introduction

An anomaly or anomalous behavior is the behavior that deviates from established “normal” habits. In many cases it refers to some actionable or critical state. For that reason it has been researched in wide variety of domains and wide variety of anomaly detection techniques has been emerged. For example and anomaly detection techniques are used in patients health data to detect possible symptoms of disease. In safety critical environment an anomalous behavior can indicate performance degradation with possible catastrophic outcomes.

In many cases the anomaly detection is related to outlier detection. In statistics, an outliers are a data instances that are deviate from given sample in which they occur. Grubbs in [1] defined “an outlying observation, or ‘outlier’, is one that appears to deviate markedly from other members of the sample in which it occurs”. When using outlier detection for detection of anomalies, an assumption that anomalies are distant from the rest of observations apply. However an outlying observation may be result of data acquisition error, or numerical error, or it may indicate faulty estimation of the models.

In field of computer security anomaly often refers to malicious behavior – to behavior of some agent that can be harmful and usually unwanted at given context. The computer network is space that provides opportunity for malicious agents to perform activities such as unauthorized access, privacy violation, reduction of availability of services, fraudulent operations, etc. Detection and prevention of most of malicious activities is important in ensuring the availability and reliability of the computer systems.

Systems used to detect malicious behavior in computer security are traditionally referred to as an intrusion detection systems (IDS) and are divided into three main categories: a signature detection systems, an anomaly detection systems or hybrid systems comprising both approaches. While signature based systems depend on database of attack signatures, anomaly detection systems rely on models of normal behavior. Decision in a hybrid system is based on both approaches – on a normal model as well as the malicious behavior of the attacker.

Crucial assumption in anomaly detection based intrusion detection systems is
that the malicious activity is subset of the anomalous activity. The example of such activity is the possible malicious agent trying to penetrate information system without knowledge of its typical use. Such activity is likely to be detected as anomalous. On contrary if similar agent uses knowledge of normal usage it may be difficult to detect its activity.

In present work we introduce an anomaly detection technique that uses model based on spectral analysis in order to identify a malicious behavior. Our technique is based on assumption that the attacks can introduce irregularities at given periodic contexts. These context are referred to as a frequency domain or frequency spectrum. We claim that the malicious agents performing attacks can be detected as their habit can deviate from normal behavior. Such difference can be shown on an example of typical user that follow an circadian rhythm that can lead in oscillations in network usage at about 24 hours. On contrary malware that would not follow same rhythm does not leave trails in same frequency context. Different example could be a violation of specification of the network protocol. In case the malicious user tries to escape security restrictions for example by misusing a allowed protocol to tunnel the restricted one this can imprint a different pattern in frequency spectrum, as showed by Chen and Hwang in [2, 3].

1.1. Related Work

Chandola et al. [4] addressed anomaly detection in general and also identified various approaches and application domains. They described methods based on classification, clustering, nearest neighbour, statistical, information theory and spectral analysis. They covered several application domains such as cyber-intrusion detection, fraud detection, industrial damage detection, sensor networks etc. Their contribution with respect to our work is mainly an exact definition of anomaly detection and deep, structured overview of the known techniques in various application domains. In the domain of our interest – the network intrusion detection, they depicted fact that although available data has an temporal content, known techniques typically do not exploit this aspect explicitly. The data is mostly high-dimensional with continuous as well as categorical attributes. The challenge faced by techniques in this domain is the changing nature of anomalies as the intruders adapt to the existing intrusion detection solutions and the high dimensionality and high amount of the data. They showed that the existing methods in this domain are based on statistical analysis, classification, clustering, spectral decomposition and information theory.

Patcha and Park [5] covered cyber-intrusion domain focusing on statistical, data-mining and machine learning techniques. They provided overview of the solutions in use and are state-of-the art in the cyber-intrusion detection and referenced number of research systems.

Davis and Clark [6] focused on data preprocessing techniques for network intru-
sion detection. They described dataset creation, feature construction and reduction techniques. In this comprehensive review they grouped a related works according to the type of features and data preprocessing techniques they addressed. They identified aggregation of packets into flows as useful as it enforces contextual analysis and statistical measures to detect anomalous behavior. They noticed that packet header based approaches are not sufficient as the use of defense against attacks forced attackers to use different attack vectors such as crafted application data. They suggest that there is need to use features derived from contents of packets but as there is little research in this area they expect that more results would emerge in future. Onul and Ghorbani [7] derived taxonomy of features used for anomaly detection. Furthermore they introduced anomaly network intrusion detection systems which use them. Gogoi et al. [8] focused on comparison of specific techniques used for network anomaly detection. They covered supervised and unsupervised approaches covering several techniques in detail, such as statistical, signal processing, graph theoretic, clustering or rule-based techniques.

In our study we focused on papers that uses spectral analysis. Chen and Hwang in [2, 3] invented an anomaly detection technique involving spectral analysis of the network traffic to analyze spectral characteristic of network protocols (TCP, UDP). They were able to distinguish between different protocols using statistical methods on frequency spectrum of the packet arrival process. In addition they introduced statistical anomaly detection method to distinguish between legitimate and malicious TCP flows. An spectral characteristics of the network traffic has been researched also by X. He, et al. [9]. They used an technique of spectral analysis to show the signatures of different layers of the network protocols. Also time-frequency based methods has been used by Salagean [10] or Gao et al. [11] involving a wavelet transform. In [10] used a wavelet transform and higher-order statistics to discriminate attacks from normal traffic.

We were also concerned about spectral techniques in discriminating tunneling protocols. We observed that Wright et al. [12] and Dusi et al. [13] investigated an detection of encrypted tunnels inside the application layer. They addressed the problem of bypassing an network-boundary security inspection by encapsulating of data subject to restrictions (peer-to-peer, chat, e-mail and others) into protocols that are considered safe and necessary (HTTP, HTTPS, SSH, DNS etc.). Estèvez-Tapiador et al. [14] and Yamada et al. [15] studied anomalies in encrypted traffic. Wright et al. [12] used features derived from packet headers aggregating packets over protocols, and time span of arrival. They counted packets in categories during an epoch resulting in vector. Then they used k-nearest neighbor (kNN) and hidden Markov model (HMM) techniques. They constructed models for different kind of encrypted tunnels such as single- or multi-flow tunnels. They were able to infer application protocols even in multiplexed packet flows without need of demultiplexing. Dusi et al. [13] brought a statistical approach to detect an tunnel inside application layer. In the paper they described different tunneling techniques and designed statistical pattern recognition classifier to identify
them. Classification of the encrypted traffic has been also researched by Ingham et al. [16, 17] or Alshammari et al. [18].

1.2. Our Contribution

A frequency spectrum based anomaly detection technique has been proposed by [2, 3, 9]. They analyzed properties of network protocols and also developed method for detection of network attack causing deviation of the spectral characteristics at given context.

In our work we are going further and we use the analysis of the frequency spectrum in detection of the tunnelled connections inside application layer and also in the detection of the malware-like behavior.

First we apply the detection technique at short time span (approx. 2 sec) to model a properties of the HTTP protocol and differentiate it from the tunnelled protocols. Tunneled protocols misuse the encapsulating protocol (in this case HTTP) in order to circumvent restrictions in computer networks (e.g. corporate proxy server). Detection of tunneling protocols has been researched by many authors [12–18], although none of them used frequency analysis.

Next we look at higher time span (approx. 24 hour) in order to detect an malware that uses HTTP protocol to leak data, but it is assumed that is behaves differently at this time context.

The detection methods are going to be part of system comprising different detection methods and providing agregation of partial outcomes.

1.3. Organization

This work is organized into 5 chapters. Chapter 2 brings theoretical introduction to anomaly detection and introduces computer network security aspects. Chapter 3 provides detailed description of proposed anomaly detection method. In chapter 4 an implementation is introduced as well as the experimentation on real data is depicted. Finally, chapter 5 concludes the work.
2 Theoretical Introduction

2.1. Anomaly Detection

In general, an anomaly detection is the problem of finding patterns in data that do not conform to expected behavior. A term anomaly refers to these non-conforming patterns. Similar term, an outlier refers to patterns that are numerically distant from the rest of sample. In most cases outlier can indicate an anomaly.

However, the origin of diversion can be caused by other factors such as artifacts or systematic error during data acquisition or numerical error during computations. Outliers caused by such agents are usually not in researcher’s interest. But the knowledge about non-conforming patterns is important due to fact that they may refer to significant information, in many cases also critical and actionable, e.g. a tumor presence may be indicated by anomalous magnetic resonance imaging (MRI) scan, network intrusion may cause observation of anomalous signature of the packets, and unexpected deviation of the physical measures in nuclear plant can have catastrophic consequences.

The anomaly detection has been studied as early as the 19th century by statisticians as a statistical method. Due now, several techniques have been developed, using domain-independent approach or developed specifically for particular domain.

Apparently simple approach of anomaly detection is to define a region representing normal behavior and declare any patterns which does not conform to this region as anomaly. This naïve approach is obfuscated by several factors:

- definition of normal behavior must contain every possible normal behavior and it is difficult to achieve,

- the boundary between anomalies and normal behavior is not accurate and can introduce wrong interpretation of particular patterns laying near the boundary,
• adaptation of malicious agents to make their outcomes appear like normal in given feature space,

• normal behavior is evolving in time and thus an normal model defined in one time span can be inaccurate or invalid in future,

• an amount of labeled data needed for derivation of the normal model is insufficient,

• presence of the noise that can be similar as anomalies, and thus it can be difficult to suppress,

• different application domains have different notion of an anomaly, thus development of domain-independent method is complicated.

In general the anomaly detection problem is difficult to solve. Most techniques solve a specific formulations of the problem, induced by a factors specific for a particular domain. The anomaly detection techniques itself were developed by adoption of the concepts from diverse disciplines such as statistics, machine learning, data mining, information theory, spectral theory.

2.1.1. Input Data

Input is generally a collection of data instances, referred as pattern, sample or observation. Each data instance is represented by non-empty set of attributes, also referred as variable or feature. Attributes can be instances of different data types e.g. continuous, categorical, or binary. Furthermore in case of each data instance consist of single attribute it is referred to as univariate otherwise it is multivariate. For multivariate instances the data types of the attributes might be mixed as well as the domain of definition might be different.

Relationship Among Data Instances. Based on presence of the relationship in data, the input data can be further categorized as point data, sequence data, spatial data, and graph data. In point data no relationship is assumed among the instances. In sequence data, presence of the total order relation among data instances is assumed. The sequence data can be time-series, protein sequences, etc. In spatial data presence of metric is required. The metric determines an neigh-

1In set theory a total order is a binary relation on some set $X$. The relation of total order is defined by axioms of antisymmetry, transitivity and totality. Total order is usually denoted as $\leq$.

2 Metric, or distance function, is a non-negative function which defines distance or similarity between elements of the set. Metric is required to satisfy axioms of coincidence, symmetry and triangle inequality. A metric space is mathematical structure $(X, d)$, where $X$ is a set and function $d : X \times X \to \mathbb{R}$ is a metric.
bourhood of each data instance. The examples of metrics are Minkowski metric\(^3\) (e.g. Euclidean distance or Manhattan distance), Levenshtein distance (editation distance between strings of characters) or Mahalanobis distance. Typical example of spatial data is the coordinate in geographic coordinate system or, assuming our definition, also textual data (notice that Levenshtein distance is metric among the strings of characters). The graph data instances are represented by graph structure\(^4\). As an example of the graph data can be a map of social interactions on community.

In case context are mixed we refer to spatio-temporal (e.g. climate data) or graph-temporal data (computer network packet flows).

Data Labels. Labels associated with particular data instances denote if instance is anomalous or normal. Labeling is often done by human expert hence it is very expensive and requires huge effort. Obtaining labels for all possible normal behavior is often less difficult than obtaining labels for anomalous behavior. Moreover, anomalous behavior is dynamic so new types of the anomalies might originate. Newly formed anomalies might be then missing from models and hence might elude undetected in detection process.

Instead of dichotomous labeling (marking instances as normal or anomalous) an more comprehensive classification can be provided. This may have advantage in construction model of specific normal behavior.

2.1.2. Anomalies

Based on presence of the relationship between data instances and problem formulation, anomalies can be divided into point anomalies, contextual anomalies and collective anomalies.

Point anomalies. In the simplest case, if an individual data instance is considered as anomalous with respect to the rest of data. No information about relationship between data instances is assumed. This type of anomaly is target of most of the research studies.

Contextual anomalies. In many cases, an context is present in data set. Context is induced by the structure of the data. In case a data instance is anomalous

\(^3\) Minkowski metric, defined as \(d(x, y) = (\sum_{i=1}^{n}(x_i - y_i)^k)^{\frac{1}{k}}\), is a distance between \(n\)-vectors \(x\) and \(y\). By choosing value of parameter \(k = 1\) we get a Mahattan or a Hamming distance, for \(k = 2\) we get an Euclid distance, or for \(k = \infty\) we get a Chebyshev distance.

\(^4\) In most common sense, a graph \(G\) is mathematical structure \(G = (V, E)\) comprising a set of vertices \(V\) with set of edges \(E\). Edges can be two-element subsets of \(V\) (undirected graph) or ordered pairs of elements of \(V\) (directed graph). In addition if weight function \(-w : E \to R\) is defined, assigning a number (e.g. weight, price, etc.) to each edge, we call structure \(G = (V, E, w)\) a weighted graph.
only within a given context, it is called \textit{contextual anomaly}. The notion of the context has to be specified within problem formulation. By introducing the context in data, the features are divided to \textit{contextual features} and \textit{behavioral features}.

The \textit{contextual features} are used to determine the context for particular data instance. As an examples of the contextual features are: a timestamp denoting temporal context in sequential data, a geographic coordinate denoting spatial context.

The \textit{behavioral features} define non-contextual characteristics of an instance. For example, the number of arrived packets during network communication within a specific time span is considered as an behavioral attribute. Identical data instances (in terms of behavioral attributes) may be considered as anomalous or non-anomalous in a different contexts.

\textbf{Collective anomalies.} If a collection of related data instances is anomalous with respect entire dataset it is called \textit{collective anomaly}. The collective anomaly is defined only in data set where an relationship among instances are related, e.g. in sequence data, graph data or spatial data.

It is important to note that \textit{point anomalies} can occur in any data set, while \textit{contextual anomalies} depend on notion of the context and its definition in problem formulation, and \textit{collective anomalies} are relevant for data where relationship among instances is defined (e.g. distance metric). So by taking in account the
context information a point or collective anomaly detection problem can be con-
verted into contextual anomaly detection problem.

2.1.3. Techniques

Chandola et al. in [4] provided an comprehensive overview of anomaly detection

techniques. They covered a wide range of application domains as well as wide
range of used techniques depicting the advantages and drawbacks of each. We
provide brief extract of them in this section.

Availability of the data labeling significantly affects usability of particular
anomaly technique. Based on extent, in which labels are present in data, following
modes of operation of anomaly detection techniques are available: supervised,
semi-supervised and unsupervised.

**Supervised anomaly detection** In supervised anomaly detection availability
of labeled data is assumed. This approach has two major issues. First is that
it requires model for both – the normal and the anomalous instances. As the
anomalous instances are less frequent it takes huge effort to obtain and to label
data instances for all possible anomalous behaviors. Secondly it is an problem of
imbalanced class distribution. Obtaining a accurate and representative labels for
all classes is difficult.

**Semi-Supervised anomaly detection** assumes availability of data labels only
for one class. Generally, it is not easy nor possible to model anomalies as this
might entail previously unseen catastrophic event, to be predicted using the given
method. Thus the vast majority of semi-supervised techniques model normal
behavior.

**Unsupervised anomaly detection** do not require training data. The unsuperv-
vised techniques are based on assumption that normal instances are more frequent
or have denser distribution than anomalous.

Typically, the output produced by anomaly detection technique are scores or
labels. Score is assigned to each data instance depending on the degree to which
that instance is considered as anomaly. Label is assigned to each data instance
to distinguish normal or anomalous instance.

Techniques can be further subdivided based on method into following subcate-
gories: classification, nearest neighbor, clustering, statistical, information theoretic,
Network Anomaly Detection by Means of Spectral Analysis

spectral decomposition⁵, etc. Some of the techniques are inherently unsupervised, e.g. nearest neighbor and clustering based.

Classification is the problem of identification to which of a categories a observation belongs to. It operates in two phases: first it “learns” a model based on subset observations (training set) and second it infers a class for new observations (testing set) based on learnt model. Examples of classification techniques are:

- **Rule based** classification or concept learning uses a rule or concept based on logical representation of the data instance.

- **Bayesian network** uses a model represented by a probabilistic graphical model.

- **Support vector machines** is a problem of finding discriminating hyperplane in a feature space such that it maximizes distance from the data instances of particular classes. A kernel trick can be used to map a observation into inner product space without need to compute the product explicitly.

- **Artificial neural networks** is a network of artificial neurons, an abstractions of biological neurons. They try to resemble learning proces of the biological neural networks.

**Nearest neighbor.** Under assumption that anomalies are more distant from other data instances, than normal data instances. A detection method based on neighbourhood can use the distance to k-th nearest neighbor or the relative density of each data instance to compute anomaly score.

**Clustering** is unsupervised technique that groups instances based on their similarity measure. It uses assumption that normal data belongs to a larger and denser clusters of similar instances, while the anomalies occur alone or in smaller or sparser clusters.

**Statistical** anomaly detection techniques uses principle stated by Anscombe in [19]: “An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed.” Statistical techniques assume that anomalies occurs in low probability areas of

⁵ Spectral decomposition refer to canonical decomposition provided under spectral theorem, called also eigendecomposition. However, in present work we are concerned with different decomposition – a Fourier transform which is part of spectral theory as well. The Fourier transform is used in field of statistical signal processing to transform time-domain signal into frequency-domain.
the stochastic model. A model estimation is done by fitting to given data (training set) and prediction if unseen observation belongs to a normal model is done by statistical hypothesis testing. Based on nature of the model, statistical techniques can be subdivided into parametric and nonparametric. While parametric techniques rely on presence of distribution over data, the fitting is then process of estimation of distribution parameters. Non-parametric techniques do not rely on data belonging to any particular distribution, they may involve histogram model or the kernel function based model.

**Information theoretic** anomaly detection techniques use information theoretic measures, such as entropy, in analysis of data. The key assumption is that the anomalies induce irregularities in information content.

**Spectral decomposition** is used to embed the data in lower dimensional subspace in which the data instances can be discriminated easily. Many techniques based on Principal Component Analysis has been emerged (e.g. [20]). Some of them decompose space to normal, anomaly and noise subspaces. The anomalies can be then detected in anomaly subspace.

Spectral analysis is also related to statistical analysis of the time domain signal in frequency domain. The power spectral density (PSD) of a stochastic process is estimated and statistical properties are analyzed in this representation. In present work we are concerned about this method and more comprehensive description is provided in chapter 3.

### 2.2. Computer Network Security

Computer security focuses on maintaining confidentiality, integrity and availability of information and information systems. The information or the information system that is accessible from the network must face the problem with possibility of unauthorized access, violation of integrity or reduction of the availability. These concern are true especially systems that are in economical interest of the potential malicious agents such as banking portal, Virtual Private Network (VPN) access to the companies, etc.

Detecting and preventing the activity of malicious agents is the goal of the intrusion detection systems (IDS). The intrusion detection systems can be categorized into network, host-based. In the Host-based intrusion detection systems (HIDS) an malicious activity is identified by examining the actions performed on the attacked computer. Network intrusion detection systems (NIDS) are independent platforms that identifies malicious activities by examining the network traffic.

The intrusion detection systems can be also divided to active and reactive. The active intrusion detection systems are monitoring and detecting potential malicious activities and in case of positive findings they raise alert or record an audit.
log. On the ohter hand the reactive systems also known as intrusion prevention systems (IPS) responds to the malicious activity by blocking the network traffic from the suspected source.

According to [5] the network intrusion detection systems can be further subdivided into signature or misuse detection, anomaly detection systems or hybrid systems comprising both approaches.

Signature detection is technique that relies on database of patterns of known attacks. The patterns are then compared with data from the environment. The model of intrusive process is represented by the patterns. These systems try to collect an evidence of malicious activities irrespective to the normal behavior. While the signatures of attacks are often conclusive evidence of malicious activity, an detection of alarm is accurate. However system fails to detect and malicious activity in case the attack model is changing or is new, unknown to a system.

An anomaly detection systems detect the malicious activity by comparing the observations whith model of normal behavior. They rely on assumption that malicious activity is subset of anomalous ones. As it models a normal behavior it able to detect new previously unseen anomalies. In case the malicious behavior is not anomalous, it is not detected. Moreover if the model is not accurate an legitimate activity can be treated as malicious.

Hybrid system combines advantages of both approaches. Is models a normal behavior and provides detection of anomalies as well as it models an malicious activity combining outcomes of both approaches.

Chandola et al. in [4] provided an comprehensive overview of anomaly detection techniques. They covered a wide range of application domains as well as wide range of used techniques depicting the advantages and drawbacks of each. Patcha et al. in [5] provided comprehensive overview of anomaly detection techniques used specifically in intrusion detection systems. Some of the techniques has been introduced in section 2.1.3 in this section we focus specifically on techniques used in network intrusion detection not mentioned above.
Chapter 3.

3 Proposed Method

In this section we derive anomaly detection method based on spectral decomposition of time-domain signal. Our goal is to involve statistical analysis of the frequency components of a time-domain signal for detection of the malicious behavior.

On the high-frequency scale (sample rate our method ought to be capable of detection of the tunneling protocols in application layer, while on the very low frequency scale our method is intended to be capable to detect anomalous behavior in wider time contexts. The sample rate is induced by data format in use. Data providing traffic flow statistic alread has been sampled at very low rate (e.g. \( s = 0.0033Hz \)) and thus it is not possible to analyze higher frequencies. The frequency scale is generally a tradeoff between computational complexity and the needed resolution.

Tunneling protocol refer to encapsulation of one network protocol in payload of the other. By using tunneling protocol a malicious agent can transfer data belonging to prohibited protocol over a network, bypassing a security policies. Our hypothesis is that application protocol imprints specific pattern in a power spectral density distribution. If the tunneling protocol is in use, these patterns ought to be different and anomalous behavior can be detected.

The periodicity can be analyzed on different scales, while tunneling protocol violates normal pattern on higher frequency scales, we are also concerned about anomalous patterns in very low frequency scales. This other patterns come up not of violation of the standard protocols, but of departure from the typical behavior or the habits. E.g. an malware communicating with the remote server using the HTTP protocol does not violate the application protocol. But by exchanging information in regular manner, not depending on time of the day, its behavior will be distinct from the behavior of the typical web user who uses the network in a way, periodic manner.

He et al. [9] showed that the different layers of the network protocols imprint distinct patterns in a power spectral density distribution. Further work of Chen and Hwang [3] used the features derived from frequency spectrum to classify
malicious and normal traffic. They noticed that transport protocols (transmission control protocol – TCP and user datagram protocol - UDP) have distinct power spectral density distribution. They exploited this property to identify low-rate denial-of-service (DoS) attacks on TCP protocol. In their work they focused on reduction-of-service (RoS) attack. The RoS attack unlike denial-of-service (DoS) attack does not attempt to completely deny the service by throttling the resources using a fake requests, but the attacker’s focus is on reduction of the quality (e.g. prolong the response times) by using small ammount of the requests. Due to low traffic during attack, RoS attacks are hard to detect with volume-based methods.

Even though it is not possible to analyze payload of particular packets in encrypted connection, it is possible to observe the time of the packet transit, its size, direction, source and destination endpoint, etc. This data is denoted as packet traces and it is extracted from unencrypted part of the packet. The goal is to develop feature creation and pattern recognition method for the network packet traces involving spectral analysis. The method is supposed to detect tunneling protocols only by observation of the packet traces.

3.1. Data Collection

The input data for our method consists of a timestamped packet traces or a timestamped traffic flow statistics.

Both can be obtained by capturing network packet headers. While first one stores each packet (or the packet header) as a single record, other one contains total count of the packets and ammount of bytes for related sequence of the packets. Relation between packets is determined depending on the traffic flow protocol. E.g. for Transmission Control Protocol (TCP) relation is determined using following information: capturing interface, source and destination IP address, source and destination port.

Due to differences in described formats, we extract following attributes from packet trace or traffic flow data in our experiments, to allow usage of unified processing methods:

- packet or packet flow timestamp, i.e. the time when the (first) packet passed through the capturing gateway,

- flow 5-tuple – i.e. transmission protocol specification and identification of source and destination endpoint (e.g. for the TCP or the UDP protocols it is address and port, destination address and port),

- size of packet’s payload or total size of the packets in case of flow data,

- count of the packets (for packet trace data it is always one),
• direction\(^1\) with respect to initial packet within given flow,

We define source endpoint as the endpoint that initiates connection and the destination endpoint as the endpoint that accepts the connection. Inbound and outbound packets are distinguished by direction attribute.

We further require that traffic flow data are always captured periodically in defined time span. We can look at this process as a sampling process. We consider the flow capturing period is lower bound for the sampling period. This can of course cause aliasing as the original signal contains higher frequency components than the sampling frequency induced by data capturing process. For packet trace data there is no upper bound on the sampling frequency induced by capturing process, but higher sampling frequency raises the memory and processing time requirements.

For purposes of evaluation (see sections 3.4.3 and 3.4.2) data labeling must be provided. The label is basically and mark that identifies arbitrary flow and groups several flows into classes. For purposes of anomaly detection two classes are expected: normal, anomalous. However it can be complicated when discrimination between specific classes is needed. As we are exploiting properties of HTTP protocol, we take in account following classes: http normal, http tunnel and http malware, which are a subset of all possible classes available in traffic. The process of labeling of the data is described in section 4.1.3.

3.2. Feature Creation

3.2.1. Stochastic process

For a specified flow \( f \) the packet arrival process \( x_f[t] \) (or simply packet process) is defined as a count of packet arrivals at given timespan \( I = [t/s, (t+1)/s] \):

\[
x_f[t] = |\{p: f = \text{flow}(p) \wedge \text{time}(p) \in I\}|
\]

\forall t \in \mathbb{N}, \quad (3.1)

where \( s \) is the sample rate, function \( \text{flow}(p) \) yields the flow 5-tuple and function \( \text{time}(p) \) yields the timestamp of given packet \( p \). We can also define packet process for inbound and outbound flows separately:

\[
x_{f,d}[t] = |\{p: \text{flow}(p) = f \wedge \text{dir}(p) = d \wedge \text{time}(p) \in I\}|
\]

\forall t \in \mathbb{N}, \quad (3.2)

where \( \text{dir}(p) \) yields a direction of the packet.

Note that we will refer to single flow and to avoid confusion we will denote packet process as \( x \) and the inbound and outbound packet processes as \( x_{in} \) and \( x_{out} \) in equations.

\(^1\)We embed direction information into the size parameter using negative size if packets travel from destination to source and otherwise positive.
3.2.2. Spectral density estimator

According to a Wiener–Khinchine theorem the power spectral density $S_{xx}(f)$ (PSD) of the wide-sense stationary stochastic process is obtained by application of discrete-time Fourier transform $\mathcal{F}(\omega)$ on autocorrelation function of the packet process $R_{xx}[\tau]$:

$$R_{xx}[\tau] = E[x[t]x[t+\tau]], \quad (3.3)$$

$$S_{xx}(\omega) = \mathcal{F}_{R_{xx}}(\omega) = \sum_{\tau=-\infty}^{\infty} (R_{xx}[\tau] \exp(-i\omega\tau)) \quad \forall f \in \langle -\frac{s}{2}, \frac{s}{2} \rangle, \quad (3.4)$$

where $\tau$ is the time-lag, $E[\cdot]$ is expected value of a random variable, $i$ is the imaginary unit and $\omega$ is the angular frequency $\omega = 2\pi f$.

Analyzing the inbound and outbound packet process can lead to definition of the cross spectral density[21]. As the power spectral density is Fourier transform of the autocorrelation function of the stochastic process cross spectral density is the fourier transform of the cross-correlation function of two stochastic processes. The cross spectral density is complex as because the cross-correlation function is not symmetric. For inbound and outbound packet process we define cross-correlation function $R_{io}$ as well as the cross-spectral density $S_{io}$ as follows:

$$R_{io}[\tau] = E[x_{in}[t]x_{out}[t+\tau]], \quad (3.5)$$

$$S_{io}(\omega) = \mathcal{F}_{R_{io}}(\omega) = \sum_{\tau=-\infty}^{\infty} (R_{io}[\tau] \exp(-i\omega\tau)) \quad \forall f \in \langle -\frac{s}{2}, \frac{s}{2} \rangle, \quad (3.6)$$

Equations (3.3), (3.4), (3.5) and (3.6) hold under assumption that packet process is wide-sense stationary random process. This assumption seems to be false for infinite time span in network traffic. Furthermore the time span is usually limited to finite number of samples. For practical reasons we involved an windowing function $w(n)$ and discrete Fourier transform instead of discrete-time Fourier transform. The simplest windowing function – a rectangular windowing is defined as follows:

$$w(n) = \begin{cases} 
1, & \text{if } n \in (0, M) \\
0, & \text{otherwise} 
\end{cases} \quad (3.7)$$
Proposed Method

Figure 3.1: Rectangular and hamming windowing function for \( M = 250 \)

Windowing function is nonzero inside specified interval \((0, M)\) otherwise it is zero (see fig. 3.1).

Parameter \( M \) is length of sub-sequence selected from packet arrival process. If the parameter \( M \) is too high the packet process is unlikely to be stationary, on the other hand selecting too low value causes spectral leakage, i.e. the energy of the main lobe of a spectral response "leaks" to the sidelobes distorting the spectral responses [22] (fig. 3.2a, 3.2b).

![Graph A](image1.png)  
(a) rectangular, \( M = 1000 \)

![Graph B](image2.png)  
(b) rectangular, \( M = 25 \)

Figure 3.2: Detail of the discrete-time Fourier transform (DTFT) of a sinusoid with rectangular windowing. The unit on x-axis is a Discrete fourier transform (DFT) bin. The red dots show values returned by DFT. It is visible that for decreased window size (fig. 3.2b) the distortion is higher. Note that this is artifact of the selected windowing and the period of the sinusoid.

We iteratively apply windowing function to the whole sequence of samples generating non-overlapping adjacent sequence of windows. We identify the particular window in this sequence with upper index – e.g. \( S_{xx}^i \) is a power spectral density function of an \( i \)-th window. Thus, we rewrite equations (3.3) and (3.4)
for non-overlapping windows as follows:

$$R_{xx}^{i}[m] = \frac{1}{M} \sum_{t=0}^{M} x[t + iM] x[t + m + iM] ,$$  \hspace{1cm} (3.8)

$$S_{xx}^{i}(k) = \mathcal{F}R_{xx}^{i}(k) = \sum_{m=0}^{M-1} \left( R_{xx}^{i}[m] w(m) \exp \left( -\frac{i2\pi m k}{M} \right) \right) \quad \forall k \in \{0, 1, 2, ..., M - 1\} .$$  \hspace{1cm} (3.9)

and similarly for cross spectral density:

$$R_{io}^{i}[m] = \frac{1}{M} \sum_{t=0}^{M} x_{in}[t + iM] x_{out}[t + m + iM] ,$$  \hspace{1cm} (3.10)

$$S_{io}^{i}(k) = \mathcal{F}R_{io}^{i}(k) = \sum_{m=0}^{M-1} \left( R_{io}^{i}[m] w(m) \exp \left( -\frac{i2\pi m k}{M} \right) \right) \quad \forall k \in \{0, 1, 2, ..., M - 1\} .$$  \hspace{1cm} (3.11)

Note that the windowing function is inherent in equations (3.8) and (3.9) by using limited ranges of summation, and domain of definition of the power spectral density.

![Figure 3.3: Detail of the discrete-time Fourier transform (DTFT) of a sinusoid with hamming windowing. The unit on x-axis is an Discrete fourier transform (DFT) bin. The red dots show values returned by DFT. It is visible that for decreased widow size (fig. 3.3b) the distortion is higher.](image)

By involving windowing function we introduced spectral leakage. Use of apodization function, e.g. Hamming (see fig. 3.3a and 3.3b), could be appropiate in decreasing leakage:

$$w(n) = \begin{cases} 
0.54 - 0.46 \cos \left( \frac{2\pi n}{N-1} \right), & \text{if } n \in (0, M) \\
0, & \text{otherwise}
\end{cases} .$$  \hspace{1cm} (3.12)
The hamming window still allows to leak the power to nearby DFT bins as the main lobe is wider, but the attenuation of in side lobes is higher. Even the hamming window cannot provide good results for the low values of the $M$ parameter. In addition there is always tradeoff between effective bandwidth and sidelobe attenuation. Yoon et al. [23] windowing technique involving a Butterworth filter to overcome the problems with reduction of bandwidth while elevating sidelobe attenuation.

The sampling rate $s$ must be selected according to the Nyquist theorem. Too low value entails aliasing\(^2\), while too high value incurs data storage and processing overhead. By application of the Fourier transform original features has been mapped into new space. We denote features in new space as spectral components (or frequency components). Temporal context of the features has been altered. In new feature space the temporal context is determined by sequence of the detection windows of size $M$.

The determination of the changes in spectrum in local time context is related to short-term Fourier analysis or short-term Fourier transform. Whereas this approach introduces resolution problems wavelet transform or multiresolutional analysis can be used for time-frequency analysis. We are not concerned with temporal context at this point. Although temporal aspect is still present it has not been used in further analysis in present work. Anomalies in new feature space are thus regarded to as point anomalies.

### 3.2.3. Parameters

There are few parameters affecting generation of the features: the sample rate $s$ and the window length $M$. In addition, spectral components can be subject to further feature extraction comprising combination of existing features or discarding irrelevant, redundant or noisy\(^3\) ones using data-mining methods or based on the empirical domain knowledge.

This aspects and process of seeking of proper parameters are subject of further research and they are discussed in chapter 4. Another aspects related to searching for proper parameters (statistical validation procedure) are discussed in subsection 3.4.

### 3.3. Model Estimation and Anomaly Detection

In order to detect anomalies a model of normal or anomalous behavior or even both must be constructed. The naive approach is to estimate an multivariate

\(^2\)The aliasing is caused by folding of the frequencies above Nyquist frequency $\frac{s}{2}$ symmetrically below this frequency. To properly reconstruct the signal that contain no frequency higher than $f_{max}$ the sample rate is bounded by $s > 2f_{max}$.

\(^3\)In case of low signal-to-noise ratio, a feature is typically not useful for discriminative outcome.
gaussian model under assumption of central limit theorem. This approach encounters several problems. The most notable is the curse of dimensionality—an phenomena occurring when analyzing data in high-dimensional spaces. The statistical view on this problem is that by applying transformations to create high-dimensional features the data become sparse. It is problem to have statistically significant outcome as the amount data needed is emerging exponentially with dimensionality.

Another facet of mentioned naïve approach is assumption of central limit theorem, meaning that the data instances are independent and identically distributed. In addition, to obtain an sound and reliable result sufficient number of data instances must be present. This is often problem as the data instances that are related to anomalies are less common in data. Additional data instances can be artificially simulated in order to enable ability to construct an reliable model. Introducing systematic errors during simulation can lead to biased model estimator.

The solution to course of dimensionality is to reduce dimensionality of the feature space. The reduction can be performed using linear transformation. The goal is to find a linear basis such that the resulting data is denser and thus contains more information. This idea is used by several linear dimensionality reduction techniques e.g. Principal Component Analysis (PCA). We can also refer to finding manifold of denser data in original sparse feature space. Reduction of dimensionality is then process of estimation of topological properties of such manifold and the estimation of relations between data instances on the topological surface. This technique is used in Kohonen maps (KM) or local linear embedding (LLE) as the representatives of non-linear dimensionality reduction techniques.

Another approach is to use empirical domain knowledge to infer relations in the data to perform transformation in lower dimensional space.

Spectral density of a signal in time domain represents continuous decomposition of signal to periodic components present in the original signal. Althought it is continuous function in practice only estimations are available. In section 3.2.2 we derived discrete spectral density estimator based on auto-correlation function. The analyzed signal is connected to stochastic processes that occur in computer network. The frequency distribution of signals in such a system is affected by many attributes. Our assumption is that te stochastic processes in computer network are stationary. It is false though to try to pick an specific characteristic frequency component and treat it as a discriminative feature for specific underlaying process. As example when an saturated link is changing its speed the characteristic frequency changes as well, it has been explored by He et al. in [9].

Chen et al. [2] presented an approach where modeled distribution of a standard deviation over spectral densities and also multivariate distribution over frequency bands. Their model used diagonal covariance matrix. They performed hypotheses testing in order to classify an cut-off specific type of denial-of-service (DoS) attacks. In present work we consider above methods of statistical analysis of the spectral density and go further.
In following subsections we introduce proposed dimensionality reduction and model estimation methods.

### 3.3.1. Feature Extraction

At first it should be noted, that the Fourier transform of the autocorrelation function is real and symmetric. The reason is that periods in input data turn into positive and negative components. Negative components contains same information as the positive. First step in reduction of the dimensionality is discarding redundant features.

**Band Filters.** Other possible linear transformation is frequency-band filtering. The transformation matrix used to perform band-filtering is sparse and easy to interpret by human analyst. Transformation is basically dot product:

\[
X_t = XA, \tag{3.13}
\]

where \(A\) is \(n \times m\) transformation matrix, \(X\) is original \(n\)-dimensional sample of length \(N\) (\(N \times n\) matrix) and \(X_t\) is new sample of \(m\) dimensions; data instances are row vectors. Recall the equation (3.9); the Power Spectral Densities \(S_{xx}(k)\) are the element of matrix \(X\) at \(k\)-th row and \(i\)-th column.

By transformation matrix

\[
A = \begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1 \\
0 & 1
\end{bmatrix} \tag{3.14}
\]

we provide sum of the lower half and the upper half of the features:

\[
\begin{bmatrix}
x_{11} & x_{12} & x_{13} & x_{14} \\
x_{21} & x_{22} & x_{23} & x_{24} \\
\vdots & \vdots & \vdots & \vdots \\
x_{N1} & x_{N2} & x_{N3} & x_{N4}
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
1 & 0 \\
0 & 1 \\
0 & 1
\end{bmatrix} = \begin{bmatrix}
x_{11} + x_{12} & x_{13} + x_{14} \\
x_{21} + x_{22} & \vdots \\
x_{N1} + x_{N2} & x_{N3} + x_{N4}
\end{bmatrix} \tag{3.15}
\]

In similar manner band filters can be constructed in higher dimensional space.

**Principal Component Analysis.** Different approach is to involve a Principal Component Analysis (PCA). The PCA is used to perform orthogonal transformation to convert a set of correlated data instances into another set that is uncorrelated. The components in new feature space are ordered such that the first component has the largest variance,

i.e. the components are ordered from the most informative to the least informative ones. As the variance is decreasing data is more sparse and less informative. In our analysis we use first \(m\) components to reduce dimensionality and enforce
density of the data. Each component is represented by vector in orthogonal transformation basis called eigenvector denoted $e_i$ for $i$-th component. An scalar value $\lambda_i$ representing an variance of $i$-th component called eigenvalue and it is associated with each eigenvector. A method of computation of the eigenvectors $e_i$ and eigenvalues $\lambda_i$ based on covariance matrix decomposition is defined as follows: Given a sample $X$ ($N \times n$ matrix), compute sample mean vector $\overline{x}$ and sample covariance matrix $S$:

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i ,$$

$$S = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \overline{x}) (x_i - \overline{x})^\top .$$

Eigenvectors and eigenvalues are then solution to equation:

$$Se = \lambda e .$$

Ordering eigenvectors and eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$ and selecting first $m$ eigenvectors results in linear transformation:

$$E = [e_i^\top]_{i=1}^{m} ,$$

$$X_t = EX .$$

The portion of variance explained by each of the components included in transformed sample $X_t$ is a quantitative measure $q$ of the model, defined as:

$$q_m = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{j=1}^{n} \lambda_j} .$$
Proposed Method

To obtain an effective transformation a ratio \( q_m \) should fall off rapidly for sufficiently small number of retained eigenvectors \( m \). For example in image processing only first 3 – 5 eigenvectors are significant while the input data (images) can have hundreds of dimensions. Advantage of using PCA transformation to frequency components is that it introduces new feature space that is much more informative, than the original feature space. Drawback is that the orthogonal basis is harder to explain.

**Moment Estimate Matrix.** Finally, instead of analysing components individually, an statistical aggregation can be applied, e.g. variance estimation of the individual spectral vectors. Extending this approach can lead in construction of new \( k \)-dimensional feature space using first \( k \) moments, i.e. the mean, variance, skewness and kurtosis of the spectra.

An unbiased estimator of the \( k \)-th moment \( \mu^{(k)} \) is a sample \( k \)-th moment \( \bar{m}^{(k)} \):

\[
\bar{m}^{(k)} = \frac{1}{n} \sum_{i=1}^{n} x_i^{k},
\]

applied on sample \( X = [x_1, x_2, \ldots, x_n] \) drawn from population (\( x_i \) are column vectors \( N \times 1 \)). In our case the sample is the finite set of the spectral components that are drawn from infinite spectrum. A matrix of moment estimations \( \bar{M} \) of sample \( X \) is then defined as:

\[
\bar{M} = [\bar{m}^{(k)}]_{k=1}^{m}.
\]

Moments represents statistics that describe the shape of the distribution of a population.

**3.3.2. Model Fitting**

**Gaussian Model.** As stated, we propose use of parametric model. Assuming central limit theorem a model ought to be \( m \)-dimensional Gaussian distribution \( \mathcal{N}(\mu, \Sigma) \), where \( \mu \) is the mean vector and \( \Sigma \) is the covariance matrix. Values on diagonal of the matrix are variances of the particular features. Remaining values are covariances and for uncorrelated data they are equal to zero.

Given the number of dimension \( m \) of feature space, number of free parameters when using full covariance matrix is \( m + m^2 \), while when reducing to diagonal matrix the count of the free parameters falls off to \( 2m \). It can be practical to assume that features are uncorrelated and retain only values on diagonal even if it is known that this assumption is false. It is because number of samples needed to accurately estimate the model grows with count of the parameters. Reducing covariance matrix to diagonal will reduce size of data needed quadratically.

In estimation of parameters of the model one can involve Maximum Likelihood Estimates (MLE). As this estimate is sensitive to presence of noise in data a
robust estimator ought to be taken in account. Rousseeuw in [24] introduced Minimum Covariance Determinant estimator. The idea is to find an observations whose sample covariance has the smallest determinant. This subset is then used to compute standard estimates of mean and covariance.

**Gaussian Mixture Model.** By grasp of the central limit theorem we assume that when modeling a simple behavior captured by sample $X$, the data instances of the samples are independent and identically distributed (i.i.d.). In case, the behavior is complex the sample $X$ can be divided to $k$ equally distributed disjoint subsamples $X_i$ in which the data instances can be assumed to be i.i.d. An attractor distribution in this case is countable mixture of Gaussian distributions $Mix_{i=1}^{k}(\mathcal{N}(\mu_i, \Sigma_i) w_i)$, for each subsample $X_i$, where $w_i$ is the weight of the distribution in mixture and it is proportional to cardinality of subsample $|X_i|$.  

### 3.3.3. Anomaly Score

An anomaly score of test instance $x_t$ is test statistics that is used to determine if the instance is anomalous or not. It is related to statistical hypothesis testing where the null hypothesis $H_0$ is the conjecture that instance is anomalous.

For example following statistic is given by Grubbs in [1]: under assumption that instance is generated by a univariate gaussian distribution a $z$-statistic can be computed:

$$z = \frac{x_t - \bar{x}}{s},$$  \hfill (3.24)

where $x$ is test instance, $\bar{x}$ is sample mean and $s$ is standard deviation of the sample. A instance is declared anomalous if:

$$z > N - 1 \left( \frac{t^2_{0.5\alpha/N, N-2}}{N - 2 + t^2_{0.5\alpha/N, N-2}} \right),$$ \hfill (3.25)

Where threshold $t^2_{0.5\alpha/N, N-2}$ it the value drawn of a Student’s $t$-distribution at significance level $0.5\alpha/N$. A variant for multivariate data uses a Mahalanobis distance as test statistic:

$$y^2 = (x - \bar{x})S^{-1}(x - \bar{x})^T,$$ \hfill (3.26)

where the $\bar{x}$ is sample mean vector and $S$ sample covariance matrix. Same test, using $t$-distribution, is applied to test statistics $y$ as in case of $z$-statistic in Grubbs test.

Our proposal is to provide Mahalanobis distance as an raw anomaly score for further anomaly evaluation. A discrimination threshold ought to be estimated using an empirical validation on test sample.
3.4. Parameter Selection and Empirical Evaluation

The feature creation and model fitting procedures require proper parameter selection to accurately estimate the model. The procedure for estimation a measure of fit has to be introduced. Outcome of the anomaly detection technique based on statistical model uses model to estimate anomaly score for a data instance.

3.4.1. Stratified $k$-fold Cross Validation

For estimation how accurately a model will perform in practice a cross validation technique has been developed. The input to cross validation process is sample set, along with a labeling information. In one round, the cross validation involves partitioning sample set into complementary subsets, preforming analysis and fitting the model on one set (training set) and validation of the model on the second set (testing set). In order of reduce variability multiple rounds are needed.

In $k$-fold cross validation the sample is randomly divided into $k$ disjoint subsamples of equal size. The cross validation round is then invoked $k$ times with each of the subsample used once as a testing set, while the rest is used for fitting the model. When using stratified $k$-fold cross validation the folds are selected such that the distribution of the labels in folds is uniform.

The output of the $k$-fold cross validation is set of $k$ partial outcomes. That can be averaged to produce final outcome.

If we denote a measure of the model performance or the measure of fit as $P$, the cross validation produces an estimate $P^*$ of the expected performance $EP$. If cross validation is performed using multiple subsamples the values for $P^*$ will vary. An estimator $P^*$ is biased estimator for $EP$. The reason of bias is that training set is smaller than the original sample and thus the estimated model is “worse” performing. For value $k$ equal to size of the sample, the bias is very small as always one data instance is left out of training set. This approach unleashes huge computational overhead and it is rarely used. As $P^*$ is always underestimate measure it is not a concern in many cases. The much more concerns are due to its high variance. In case the variance is hight different models are not comparable. This usually holds if the sample size is low. In that case the evaluation process does not provide statistically sound and significant outcome.

3.4.2. ROC curve

For labelled data instances drawn from testing sample, an receiver operating characteristic (ROC curve) can be computed. In order to compute ROC curve an introduction of discrimination threshold on raw score is required. If a value of the raw score of a data instance is higher than discrimination threshold the instance is marked as anomalous. The discrimination threshold in conjunction
with dataset labeling is used to compute true positive and false positive rates. The ROC curve is plot of the true positive rate \( r_{tp} \) vs. the false positive rate \( r_{fp} \) while the threshold is varied. The false positive rates and true positive rates are related to statistical hypotheses testing where the null hypothesis \( (H_0) \) is the conjecture that given instance is anomalous. Anomalous instance is also called “positive”. Table 3.1 categorizes positive and negative outcomes with respect to validity of null hypothesis.

<table>
<thead>
<tr>
<th>Reject null hypothesis</th>
<th>Null hypothesis is true (P)</th>
<th>Null hypothesis is false (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fail to reject null hypothesis</td>
<td>False negative (FN)</td>
<td>True negative (TN)</td>
</tr>
<tr>
<td></td>
<td>True positive (TP)</td>
<td>False positive (FP)</td>
</tr>
</tbody>
</table>

Table 3.1.: Relations between true and false outcomes and the validity of hypothesis \( H_0 \).

Failure to detect anomalous instance is the false positive (FP) and correct detection is true positive (TP). Given a testing sample set a false positive rate \( r_{fp} \) is a proportion of false positives \( n_{fp} \) vs. all negative instances \( n_{fp} + n_{tn} \):

\[
r_{fp} = \frac{n_{fp}}{n_{fp} + n_{tn}},
\]

and the true positive rate (sensitivity) \( r_{tp} \) is proportion of true positives \( n_{tp} \) vs. all positive instances \( n_{tp} + n_{fn} \):

\[
r_{tp} = \frac{n_{tp}}{n_{tp} + n_{fn}}.
\]

The figure 3.5 show the space of the possible ROC curves. The blue line depicts an ROC curve of a random classifier, with no discrimination. The points above the line determines better discriminative performance than random classifier and conversely the red area delimites and worse outcome. If the ROC curve lies in green (or red area respectively) the outcome of the model is consistently good (or bad). The consistently bad model can be turned into consistently good by inverting the raw score.

Analysis of ROC curve provides a framework to select possibly optimal models and to discard suboptimal ones independently from the cost context or the class distribution.

There are several statistics developed based on ROC curve. One of the most important is the area under curve (AUC). It is equal to probability that a model will provide a higher score for a randomly chosen positive instance than a randomly chosen negative one. The AUC value lies in interval \( \langle 0, 1 \rangle \).

In addition to AUC statistics we also define a concept of dominancy based on analysis of the ROC curve.
Figure 3.5.: Analysis of ROC space

\[(r_{fp}(p), r_{tp}(p)),\] where is the parameter of the curve, dominates another outcome \(k_2\) if and only if \(\forall q \in (0, 1) : k_1(q) \geq k_2(q)\) and we denote \(k_1 \preceq k_2\). Strict domination is defined as \(k_1 < k_2 \Leftrightarrow k_1 \preceq k_2 \wedge \neg (k_2 \preceq k_1)\). It requires functions \(r_{tp}(p)\) and \(r_{fp}(p)\) are continuous.

### 3.4.3. Evaluation

By model fitting technique we call and transformation procedures and model fitting procedures. Given transformations \(t_1 : X \rightarrow X_1, \ldots, t_n : X_{n-1} \rightarrow X_n\) and a function yielding raw score (real number) \(s : X_n \rightarrow \mathbb{R}\), we will call \(a : X \rightarrow \mathbb{R}\) an anomaly score estimator, where \(a = t_1 \circ \cdots \circ t_n \circ s\). Let us call function \(A : G \times X \times Y \rightarrow (a : X \rightarrow \mathbb{R})\) a model fitting technique, or learner. Learner yields the anomaly score estimator for given set of parameters \(G\) and training sample \(X\) and labeling of the sample \(Y\) (fits the model to given sample).

For a given set of learners \(\mathcal{M}\) and their parameter sets \(G\) the outcome of ROC analysis, an AUC statistics \(AUC_k : \mathcal{M} \times X \rightarrow \mathbb{R}\), with \(k\)-fold stratified cross validation is \(P^{*}_{m,g}\) (average of partial outcomes). Outcome \(P^{*}_{m,g}\) is biased estimate of the performance of the learner \(A_m\) under given parameter set \(G_g\). The value \(P^{*}_{m,g} = 1\) means that model has perfect discriminative performance.

Described evaluation procedure would give us biased measure of the performance of the given learners. Bias tends to be conservative as it is always underestimate. However the variance of partial outcomes has to be reported. In case of large variance the model instability has to be suspected. Unfortunately
there is no universal statistics in prediction of model instability. In order to avoid instability issues sufficient labeled sample has to be provided.
Chapter 4.

4 Experiments

In this chapter we describe an experimentation with the method derived in chapter 3. Software PyNfSA has been created using Python Programming Language [25] for experimentation purposes. It is intended to be a set of tools to provide convenient environment for performing signal analysis on the network traffic data.

In following sections we introduce data acquisition process (section 4.1) as well as the process of signal analysis using PyNfSA (section 4.2). Finally we depict the results of the experimentation on real data (section 5.1).

4.1. Data gathering

For training and evaluation purposes several datasets has been used. We took in account packet trace data as well as traffic flow statistics.

For the capture and storage of the packet trace data we use PCAP format. This format is used by open-source library libpcap [26], that is available for wide range of platforms (Windows, Linux, BSD, etc.) and has bindings to several programming languages (C, perl, python, etc.).

For traffic flow statistics Cisco Systems developed an NetFlow format and communication protocol. It has now wide range of equivalent implementations, e.g. JFlow (developed by Juniper Networks), and there also exists software to generate NetFlow from PCAP files.

Cisco Systems NetFlow v. 9 protocol is published in RFC3954 [27]. The publication contains information about NetFlow data gathering and exchange process, as well as the specification of data attributes contained within dataset. However the specification is extensive, and for our experiments we use only negligible subset of the attributes as has been defined in section 3.1.

For internal representation of the data we use matrices and array databases. Our experimental software uses HDF5 suite [28] for storage of the data. This suite consists of several open-source libraries, tools and bindings to various languages.
Our experimental software is capable of conversion from PCAP or NetFlow formats to matrix representation in HDF5 files.

4.1.1. Simulation

Virtual testbed has been used to simulate traffic in a low scale. A tcpdump [29] software has been used to capture packet data in experimental testbed. Testbed consist of three or more virtual computers. Communication between hosts has been routed through one of them where tcpdump has been used to capture the communication in the testbed. The tcpdump captures every packet passing through the interfaces and stores the data in PCAP [26] file. Each captured packet has been truncated to 200 bytes before storing.

Various scenarios have been simulated in the testbed in small scale: experimental malware-like software, file transfers, text terminal sessions, tunneled connections, virtual private network (VPN) connections.

The experimental malware-like software called woodpecker has been created for purposes of present work. It has been intended to simulate behavior of the malicious agent sending and receiving information to remote server using HTTP protocol. The delay between consecutive requests as well as the size of the payload has been randomized. At protocol level an agent has not been intended to violate specification, thus standard implementation of HTTP protocol has been used. At least ten instances of the woodpecker operated over several days in experimental testbed simulating several attacked hosts.

For tunneled connection an httptunnel software has been used. This software provides data link between two endpoint encapsulating the data to payloads of the HTTP requests and responses. A random data has been used to pass through the tunnel.

For the VPN connection an OpenVPN software has been used. It is open-source and widely used VPN client and server. It has been used to tunnel the user traffic including world wide web, terminal sessions, or the file transfers. The behavior of the traffic encapsulated in tunnel has been changing. The were present single connection as well as multiplexed ones. It has been also used to pass through the HTTP proxy server to circumvent security restrictions. The VPN server uses an HTTPS service TCP port and is intended to be undistinguishable from the HTTPS service by the port number. However there are signature-based methods used to detect such behavior, it uses inspection of the packet payload. We are not concerned with this option as we intend to build an anomaly detection method instead that does not have the packet payload at the input.

Simulated data has been mixed with data obtained from other sources. Simulated malicious activities has been subject to test the performance of the method.
4.1.2. Anonymized non-public data

Two evaluation datasets has been used. One has been captured on campus routers and has been provided by this work's supervisor. The data format used an NetFlow specification to gather the data. It was an capture of flow statistics at the campus network over two weeks. An 5 minute sampling period has been used. An open-source implementation of NetFlow standards is the nfdump [30] software. It has several options of data formatting and most convenient for further processing is the line text format as the parser should be implemented as the single regular expression.

Another dataset has been captured at different locations and environment using a tcpdump software providing a PCAP format. The data comprises typical use of the computer network as the world wide web (WWW), terminal session (SSH, telnet, RDP), file transfers (SCP, CIFS) etc. It has been collected over period of one week.

4.1.3. Labeling

Labeling of the data is performed in manual manner involving expert knowledge. For the simulated traffic, labels are easy to assign as they are known during the simulation. Labeling data gathered in real network over longer period of time requires huge effort.

An approach often used, is to use state-of-the-art intrusion detection system to perform analysis and obtain labeling from its output, however labelling obtained this way can be biased especially for new unknown malicious behaviors that is subject to research.

The labeling of the data is needed for training in case of supervised or semi-supervised anomaly detection methods and it is always needed for evaluation of the performance of methods (even unsupervised).

In case the labels are not corresponding to reality it can bias the model fitting process and thus decrease performance of the model. Altough there are model fitting methods that are robust to mislabelled sample, it is important to obtain sample with correct labeling information for model finding and evaluation purposes. The evaluation of the robustness of such method has to be performed separately.

In present work labeling of captured data has been derived using manual inspection by human expert. Labeling of the simulated attacks has been know at the time of the data capture.

4.2. Implementation

For purposes of experimentation an PyNfSA [31] software has been created. It integrates several libraries in Python Programming language to provide compu-
Network Anomaly Detection by Means of Spectral Analysis

Introductions introduced in chapter 3. The solution is built around numpy [32] library. Complete list of dependencies as well as the usage and development manual is included in PyNfSA Documentation in Appendix A. Let us introduce the implementation details of the PyNfSA software in terms described in chapter 3:

Data acquisition requires libpcap [26] library for capturing packet data or the nfdump [30] software used for capture netflow statistics. The nfdump software has several output formats but the PyNfSA software relies on “long” line format as it can be easily parsed using regular expression. The PyNfSA reads and converts acquired data into internal matrix representation. The matrices are then stored using HDF5 technology [28] for later use. When parsing the PCAP or the NetFlow format, only subset of information is selected as specified in subsection 4.1.2.

Next step is to compute flow matrix. Rows in flow matrix represents data instances comprising timestamp, packet count and size, direction and flow identification. The flow identification is based on flow 5-tuple i.e. transmission protocol specification, address and port of source and destination endpoint. When modeling behavior of specific source endpoint with respect to specific application service, it is useful to discard source port information in some cases. The reason is that the application client side often requests an socket from operating system to the destination server, performs request-response roundtrip and closes the socket. For successive request new sockets are opened. Based on implementation of the traffic flow protocol in operating system new source port is assigned. In our analysis we consider that successive connections are related together. This is the case especially for HTTP application protocol, the client side often creates new socket for each small set of requests. As the HTTP application protocol are subject to our research our flow identification scheme does not include the source port.

After flow matrix is computed it is stored in HDF5 file for later analysis. Such representation of the data is very compact when compared to netflow or to the pcap format, as it is constructed based on pre-defined set of the attributes. In addition an HDF5 technology allows indexing of the array data and thus filtering in the data is often faster than in raw formats. Drawback of this representation is that addition of new attribute means processing of raw data and re-creation of whole matrix.

From the flow matrix an feature matrix is constructed. Features are created by computation of power spectral density in defined time window and using specified sample rate. Each row in feature matrix represents single data instance comprising the frequency components. A flow identification is assigned to each sample and it stored as a vector along with a feature matrix. While in flow matrix a time context is still present, it is not present in in feature matrix, the only relation between samples is the flow identification.

The labeling is implemented using filtering based on flow 5-tuple. The labels are assigned based on filters selecting the subset of flows identifications. A labeling vector is stored along with the flow identification vector and the feature matrix.
Filters are created manually, JSON media type [33] is used. The format of the filter is specified in PyNfSA Documentation.

The model fitting and evaluation uses $k$-fold stratified cross validation. During model fitting phase an training sample undergoes transformations in order to reduce dimensionality and fitting is performed on transformed sample. A model is then used to obtain raw score. For unimodal Gaussian distribution an Mahalanobis distance is used, for Gaussian Mixture Model and log-probability is used as raw score. The fitting of the unimodal Gaussian distribution for Mahalanobis distance computations and robust covariance matrix estimation can be used. The computation then involves Minimum Covariance Determinant [24].

For the machine learning procedures and SciKit library [34] has been used. It is set of classes in Python Programming Language that implements lot of algorithms for machine learning and statistical analysis. Few extensions to the computational models has been performed during the PyNfSA development.

The PyNfSA allows entering to interactive mode. Interactive mode is used to invoke analysis “on-the-fly” using an ad-hoc commands on the command line. For interactive mode iPython [35] interactive shell is used. It allow to use syntax similar to Matlab, Octave or even R by invoking special commands. In addition HDF5 data file can be used directly by Matlab or Octave as it is natively supported format,
Chapter 5.

5 Assessment and Conclusion

5.1. Results

Analysis has been performed using non-public NetFlow and PCAP data. Summary information about datasets are provided in the table 5.1. We will denote the datasets hereby as flow resp. trace. The overall time span has been two weeks. Simulation of the malicious behavior has been performed at duration of several days and mixed into both data sets.

The sampling frequency for flow has been given by nature of the data. As it is presampled at period 5 minutes, it’s inverse, a value 0.0033Hz has been used. In trace data, a sample rate have been selected based on analysis of the inter-packet time. For the HTTP traffic it has been observed that that the inter-packet time for most of the packets was less than 20ms thus sample rate of 100Hz has been chosen according to Nyquist theorem.

\[
\begin{array}{|c|c|c|}
\hline
\text{Dataset identification} & \text{flow} & \text{trace} \\
\hline
\text{Source data format} & \text{NetFlow} & \text{PCAP} \\
\text{Duration [day]} & \sim 14 & \sim 7 \\
\text{Number of annotated flows [-]} & 774677 & 1148 \\
\text{Sample rate [Hz]} & 0.0033 & 100 \\
\text{Window size [samples]} & 284 & 200 \\
\text{Window size [hour; second]} & \sim 24 & 2 \\
\hline
\end{array}
\]

Table 5.1.: Information about datasets

Table 5.2 contains information about classes present in dataset. A class http has been used to fit the models. All classes including modeled have been used in validation.

On the figures B.1 and B.2 (in appendix B) there is a ROC curve of various methods on datasets described by table 5.1. It depicts overal discriminative
properties between positive and negative classes as described by table 5.2. Quantitative measures are in table B.1. Consistently better results are achieved by using methods based on PCA dimensionality reduction.

On figures B.5, B.6 and table B.3 we compared how is gaussian model complexity affecting discriminative performance and we examined that for packet trace data, decreasing performance with increasing complexity occurred. However for netflow dataset using a bimodal instead of unimodal model significantly improves performance. The plot is showing ROC curves for gaussian mixtures with varying number of components. We decided to use unimodal distribution for packet trace data and bimodal model for flow data.

The figure B.3, B.4 depicts differences between classes. For flow the ROC curves tend to be sharp and good performing in discriminating all measured classes. Table B.2 contains quantitative measures.

Figure B.9 depicts an decrease of variances of eigenvectors. This trend is dependent on data instance of the http class. It can be shown that taking first 2 eigenvectors is sufficient. Plots B.7, B.8 and table B.5 supports this statement. It shows ROC curves and AUC values for varying number of eigenvectors used.

### 5.2. Conclusion

In present work we proposed new detection mechanism of network traffic anomalies based on statistical analysis of the frequency components of the signal.

We were focused on detecting tunneled connections and misuse of the HTTP protocol as it is easy way to circumvent restriction policies.

Our method estimates power spectral density of the packet process to create
features. Packet process is treated as stochastic stationary process of packet arrivals and departures. Power spectral density has been estimated using rectangular windowing and using Fourier transform of autocorrelation function.

Principal component analysis has been used to reduce feature space dimensionality and increase information content in retained features. Analysis of the eigenvalues showed that first 3 dimensions are significant.

Finally statistical analysis based on Mahalanobis distance with unimodal Gaussian model has been used to evaluate anomaly score. Other possibilities of estimating anomaly score can involve Gaussian Mixture Models. The score in this case, is the log-probability estimated by model.

We modeled normal behavior of the HTTP connections. The model has been fitted by means of maximum likelihood estimator (MLE) to a testing dataset comprising approximately two weeks of traffic. The model has been then cross-validated using testing set drawn from same population using stratified k-fold crossvalidation. For evaluation we used an ROC curves. ROC analysis showed that the method is performing well on the packet trace data as well as on traffic flow statistics data.

Further research is required to assess robustness in collaborative environment. The method is intended to focus on specific behavior. In collaborative system it is intended to be useful in detecting various activities by extending a models to cover different behaviors.
Bibliography


1 Introduction

PyNfSA - NetFlow Spectral Analyzer for Python is software to conveniently perform frequency analysis on PCAP or NetFlow dataset.

2 Installation

1. installation of dependencies
2. download pynfsa repository at github.com/pborky/pynfsa
3. enter the directory and invoke python nfsa.py load dataset.h5 (dataset will be created if not existing and interactive mode will be entered)

2.1 Dependencies

- python v2.7.3 - Python programming language
- impacket v0.9.6.0 - library to craft and decode network packets
- python-libpcap v0.6.2 - packet capture library bindings for python
- numpy v1.6.2 - software for mathematics, science, and engineering
- scipy v0.10.1 - software for mathematics, science, and engineering
- matplotlib v1.1.1 - python 2D plotting library
- ipaddr-py v2.1.10 - IPv4/IPv6 manipulation library in Python
- pytables v2.3.1 - package for managing hierarchical datasets and designed to cope with extremely large amounts of data
- h5py v2.0.1 - Python interface to the Hierarchical Data Format library, version 5
- fabulous v0.1.5 - library designed to make the output of terminal applications look fabulous
- scapy v2.2.0 - a powerful interactive packet manipulation program
- iPython - and powerful interactive shell

3 Usage

nfsa.py [-h]
nfsa.py [--version]
nfsa.py [options>] raw|flow|sample|model|filter|load|annotate
<database file> [input file [<input file> ...]]

Positional arguments

- raw|flow|sample|model|filter|load|annotate action to execute; raw stores “pcap” or netflow data in h5 database, “flow” marks flows and extracts attributes, “sample” computes sampling at given sample rate and transformations at given windowing, “model” fits model
to data stored in database, filter converts XML Ip filters to JSON format and "load" loads database into memory

• <database file> hdf5 array database
• <input file> input files to process

Optional arguments

• -h, --help show this help message and exit
• --version show version information
• -f pcap|netflow input file format
• -o <output file> output file
• -m <min packets> min packets per flow
• -n don’t do reverse dns
• -v, --verbose increase verbosity
• -q, --quiet do not dump to terminal

Flow extraction options Required for "flow", "sample" and "model" actions

• -i 3|4 flow identification <3-tuple or 4-tuple>
• -u don’t use SYN packets to distinguish flow start
• -p <protocol> protocol to take in account, default = 6 <TCP>

Sampling options Required for "sample" and "model" actions

• -s <sample rate> sample rate to use, can be specified multiple times
• -w <window length> window lengths to use, can be specified multiple times
• -t csd|psd tranformation to use, can be: "csd" for cross spectral density or "psd" for power spectral density

Model estimation options Required for "model" action

• -a <file> annotation file, specifies filters and labeling of the datased, format is following 
  
  
  ```
  { "srcIPs": [ "<ip>" ], "dstIPs": [ "<ip>" ], "srcPorts": [ "<port>" ], "dstPorts": [ "<port>" ], "type": "FILTER_MALICIOUS" | "FILTER_LIGITIMATE" | "UNKNOWN", "annotation": "<string>", "protocols": [ "TCP" | "UDP" | "ICMP" ] }...
  ```

• c --legit <int>,<int>,.. comma-separated list of classes considered legitimate
• --malicious <int>,<int>,.. comma-separated list of classes considered malicious
• --model <int>,<int>,.. comma-separated list of classes included in model
• --sample <pattern> regex to filter sampleset by name
• --computation <step>,<step>,... computation to evaluate
• --tex <file> append tex-like tables into <file>
4 Indices and tables

- genindex
- modindex
- search

5 Class Reference

Module containing classes for constructing and evaluating models.

class pynfsa.models.FreqBands(n_bands, log_scale=False, mean=False)

Creates projection based on frequency band filtering. The frequency spectrum is divided into equal or logarithmic intervals in which are spectral components summed.

Parameters
  n_bands : number
  count of bands
  log_scale : boolean
  if true logarithmic scale instead of linear is used, default: False

Methods

get_A(X, freqs)
Returns transformation matrix. Used internally.

class pynfsa.models.FreqBaseTransformer
Base class for frequency based transformations. The transformation matrix A is computed according to frequency vector with function get_A (that must be overridden in child classes).

Methods

get_A(X, freqs)
Returns transformation matrix. To be implemented in subclass

class pynfsa.models.FreqThresh(f_thresh, f_thresh_hi=None)

Creates projection based on frequency thresholds. The frequency components that are greater then threshold are retained, others are discarded. If optional upper threshold is provided it is applied in conjunction with lower threshold.

Parameters
  f_thresh : number
  lower threshold
  f_thresh_hi : number (optional)
  upper threshold
Methods

```python
get_A(X, freqs)
```

Returns transformation matrix. Used internally.

```python
class pynfsa.models.LinearTransformer(A)
```

Projection to new space determined by projection matrix.

Parameters

- `A`: array-like, shape `[n_orig_features, n_new_features]` or callable
  
  The transformation is denoted as dot product: \( Y = XA \). If parameter `A` is callable, the matrix determined during fit.

Methods

```python
fit(X, y=None, **params)
```

If callable has been provided instead of matrix `A` it is invoked on **params dictionary to determine `A` in formula: \( Y = XA \).

Parameters

- `X`: array-like, shape `[n_samples, n_features]`
  
  Original feature set.

- `y`: array-like, shape `[n_samples, 1]`
  
  Class information. Not used in linear transform.

- `params`: additional parameters, implemented in subclasses

Returns

- `self`: object
  
  Returns self.

```python
transform(X, y=None)
```

Perform linear transformation: \( Y = XA \)

Parameters

- `X`: array-like, shape `[n_samples, n_features]`
  
  Original feature set.

- `y`: array-like, shape `[n_samples, 1]`
  
  Class information. Not used in linear transform.

Returns

- `X_transformed`: array-like `[n_samples, n_new_features]`
  
  Transformed feature space.

```python
class pynfsa.models.Mahalanobis(robust=False)
```

Mahalanobis distance estimator. Uses Covariance estimate to compute mahalanobis distance of the observations from the model.

Parameters

- `robust`: boolean to determine whether to use robust estimator

  based on Minimum Covariance Determinant computation

Methods

```python
fit(X, y=None, **params)
```

Fits the covariance model according to the given training data and parameters.
**Parameters**  

\( X : \text{array-like, shape } = [n_{\text{samples}}, n_{\text{features}}] \)

Training data, where \( n_{\text{samples}} \) is the number of samples and \( n_{\text{features}} \) is the number of features.

**Returns**  

self : object

Returns self.

**score**  

\((X, y=None)\)

Computes the mahalanobis distances of given observations.

The provided observations are assumed to be centered. One may want to center them using a location estimate first.

**Parameters**  

\( X : \text{array-like, shape } = [n_{\text{samples}}, n_{\text{features}}] \)

The observations, the Mahalanobis distances of the which we compute.

**Returns**  

\( \text{mahalanobis\_distance} : \text{array, shape } = [n_{\text{observations}},] \)

Mahalanobis distances of the observations.

**class**  

\( \text{pynfsa.models.Modeler}(\text{opt, computations=None, steps=None}) \)

Constructs Modeler callable object used to compute and evaluate several models.

**Parameters**  

\( \text{opt} : \text{argparse.NsameSpace} \)

Arguments passed in command line.

**computations** : iterable

Each item defines an computation. Computation is defined by list of keys to steps argument.

**steps** : dictionary

Step of the computation. Steps are iterables that can be variants of same method.

**Methods**

**class**  

\( \text{pynfsa.models.Momentum}(\text{moments='mvks'}) \)

Computation of the moments of an data instance result in features in new feature space where each dimension is dedicated to particular moment.

**Parameters**  

\( \text{moments} : \text{string where characters denote moment estimator:} \)

‘m’ - mean ‘v’ - variance ‘k’ - kurtosis ‘s’ - skewness ordering determines the ordering of the computations

**Methods**

**fit**  

\((X, y=None, **params)\)

Dummy function does nothing

**transform**  

\((X, y=None)\)

Perform calculation of the moments over features.

**Parameters**  

\( X : \text{array-like, shape } [n_{\text{samples}}, n_{\text{features}}] \)
Returns moments: array-like ith shape [n_samples, n_moments]

Moments of the original features. Ordering of the moments is according to argument moments given in initialization of the object

class pynfsa.models.PipelineFixd(steps)

Fixed version of Pipeline. Original has some bugs.

Methods

fit (X, y=None, **params)
    Fit all the transforms one after the other and transform the data, then fit the transformed data using the final estimator.

score (X, y=None)
    Applies transforms to the data, and the score method of the final estimator. Valid only if the final estimator implements score.

pynfsa.models.evaluate (opt, computations, h5grp, model=None, legit=None, malicious=None, model_legit=None, steps=None)

Evaluate given computations on dataset h5grp. Choose model instances based on labels ‘model’, evaluation sample is defined ‘malicious’ and ‘legit’. If specified ‘steps’ determine computation steps.

Parameters opt : argparse.Namespace
        Arguments passed in command line.

computations : iterable
        Each item defines an computation. Computation is defined by list of keys to steps argument.

h5grp : pynfsa.dataset.H5Node
        Group in which sample is present.

model : list
        list of integer labels to be included in model

malicious : list
        list of integer labels to be included in positive instances

legit : list
        list of integer labels to be included in negative instances

model_legit : boolean
        determine if model is normal or anomalous

steps : dictionary
        Step of the computation. Steps are iterables that can be variants of same method.

Returns modeler : Modeler
        modeler instance

results : tuple
result of modeler invocation

```python
pynfsa.models.fapply(fnc, *args, **kwargs)
```

Function application over list of arguments. Applies function to each positional argument and returns list of result. If the particular argument is iterable, it is unfloed before in invocation. Keyword arguments are passed in all invocations.

**Parameters**

- `fnc` : callable
  - function to invoke: `[fnc(arg,**kwargs) for arg in args ]`

**Returns**

- `results` : list of function application results
  - Basically something like `[fnc(arg,**kwargs) for arg in args ]`

```python
pynfsa.models.fiterate(*args)
```

Generate pipeline comprising all of the arguments. Particular arguments ought to be iterable and Cartesian product of them is computed. Each value in iterables must conform to Pipeline specs.

```python
pynfsa.models.plot_roc(res, title='')
```

Given a modeler result plot an ROC curves

**Parameters**

- `res` : list
  - list of tuples: (name, auc, (fpr, tpf, threshold))

- `title` : string
  - Title of the figure

Contains objects related to sampling process.

```python
class pynfsa.sampler.Sampler(opt)
```

Constructs Sampler callable which generates samples.

**Parameters**

- `opt` : argparse.Namespace
  - Arguments passed in command line.

**Methods**

```python
pynfsa.sampler.csd(w, bounds)
```

auto-correlation of packet process

```python
pynfsa.sampler.psd(w, bounds, filter=None)
```

power spectral density of packet process

```python
pynfsa.sampler.psd1(w, bounds)
```

power spectral density of sum packet volume

```python
pynfsa.sampler.psd2(w, bounds)
```

power spectral density of average packet volume

Classes encapsulating HDF5 files and numpy matrices.

```python
class pynfsa.dataset.H5Node(opt, h5=None, grp=None, auto_create_grps=True)
```

Creates group in H5 file. Opens h5 file if not provided in argument `h5`.

**Parameters**

- `opt` : argparse.Namespace
  - Arguments passed in command line.
**h5** : tables.File
An instance of H5 file

**grp** : tables.Group
An instance of H5 group

**auto_create_grps** : boolean
If True, a group is created if it is non-existent

### Methods

**handle_exit**(try_fnc, *arg, **kwarg)
Call `try_fnc(opt, h5=h5)` in try-finally block. At the end close the file and exit.

**printTree**(padding='', last=False, maxdepth=3, maxcount=50)
Print tree contained in file.

**Parameters**
- **padding** : string
  Arguments passed in command line.
- **last** : boolean
  Last node in sequence
- **maxdepth** : number
  Limit depth
- **maxcount** : number
  Limit count

**class** pynfsa.dataset.Table**(data=None, fields=None, h5=None)**
Encapsulation of the numpy array, in order to conveniently select/update data. Parameters ————

- **data** : array-like [n_rows, len(fields)]
  the data representing this table

- **fields** [tuple] tuple of strings representing columns

- **h5** [dataset.H5Node] Initialize Table from HDF5 file

### Methods

**add_field**(field, default)
Add new column called ‘field’ and set its value to ‘default’. It can be vector or scalar (it will be broadcast).

**retain_fields**(fields)
Keep column specified in ‘fields’, others are discarded.

**save**(h5)
Save the content in ‘h5’ group.

**Parameters**
- **h5** : H5Node
  A node to store the content
select (predicate, order=None, retdset=None, fields=None, **kwargs)
Select submatrix based on predicate and fields.

Example: size = PredicateFactory('size') \ r = data.select((size >= 10) & (size < 15), fields = ('time', 'size')) \ # select submatrix containing time and size columns and rows where size is in interval [10,15)

Parameters
predicate : dataset.Predicate
Vector predicate to use for an indexing.

order : string
Determine field to be used for ordering of the result.

retdset : boolean
If true result is wrapped in new dataset.Table instance.

fields : tuple
Fields to retain in result.

Returns
a submatrix or a new Table object based on predicate :

set_fields (predicate, fields, value)
Set column specified by ‘field’, and rows matched by ‘predicate’ set its value to ‘value’. It can be vector or scalar (it will be broadcast).

class pynfsa.dataset.Variable (field)
Factory object to build an predicate instances.

Parameters
field : string
a name of the variable

Methods

class pynfsa.extractor.Extractor (fields)
Used for extraction of various attributes of raw data.

Parameters
fields : tuple
a tuple of fields to be extracted from source data

class pynfsa.extractor.FlowExtractor (fields, pattern=None)
Used for extraction of attributes from netflow data.

Parameters
fields : tuple
a tuple of fields to be extracted from source data

pattern : re.Pattern
a pattern to use instead for predefined one.

Methods

class pynfsa.extractor.PcapExtractor (fields)
Used for extraction of attributes from PCAP data.
Methods

class pynfsa.extractor.TraceExtractor(fields)
Used for extraction of attributes from TCP and UDP packet trace data.

Methods

Contains flowizer object

class pynfsa.flowizer.Flowizer(fields=('time', 'paylen', 'flow'), fflow=('src', 'sport', 'dst', 'dport'), bflow=('dst', 'dport', 'src', 'sport'), usesyns=True, opt=None)

Creates an callable that is used to generate flow data.

Parameters  fields : tuple
          tuple of field names (a columns of dataset
          fflow : tuple
          tuple determinid the flow schema (forward packets)
          bflow : tuple
          tuple determinid the flow schema (backward packets)
          usesyns : boolean
          use syns to determine flow direction
          opt : argparse.Namespace
          Arguements passed in command line.

Methods

pynfsa.util.fig(plt_fnc[, name=None, show=False]) ➔ figure,result
It is convience function for matplotlib figures. Creates an figure and axes and executes plot function(s) (plt_fnc) on axes. count of axes is same as number of plot function(s). It name is is given it must be list-like object of same size as plt_fnc then each axes is named accordingly. If show is true figure is showed.

Parameters  plt_fnc : callable
          callable or list of callables that receive argument ax, which axes.
          name : string or sequence of string
          names, must be same shape as plt_fnc
          show : boolean
          show an figure

Returns  figure : figure object
          can be used to show figure is show=False
          result : -
result of plt_fnc invocation(s)

```
pynfsa.util.get_packets(fn, extractor)
  Extracts information from packets given by file name - ‘fn’
```

```
pynfsa.util.int2ip(ip)
  convert int to string IPv4 address
```

```
pynfsa.util.ip2int(ip)
  convert string IPv4 address to int
```

```
pynfsa.util.opts(args=None)
  User interface features
```

```
pynfsa.util.reverseDns(ip)
  execute reverse lookup of given address
```

```
pynfsa.util.scalar(x)
  convert to scalar if possible
```

```
pynfsa.util.scatter(ax, X, y, normalization=None, transform=None, labeling=None)
  Convenience method for scatter plotting of samples. Input:
  ax - axes object X - sample set [n_samples,n_dim] y - classes of samples [n_dim] normalisation - callable used to normalize of X or None transform - callable used to transform of X to lower dimensional space or None labeling - callable used to determine text labels of y or None

  Output: scatter plot instance
```

```
pynfsa.util.timedrun(fnc)
  decorate function to print time of execution in seconds on standard input Input:
  fnc - function to decorate

  Output decorated function
```

example:

```
In [1]: @timedrun def test():
    
    <invoke time consuming actions>

    test()

Out[1]: ## function <test>: started <output messages of function test> ## function <test>: elapsed time: ...
```
Index

A
add_field() (pynfsa.dataset.Table method), IX

C
csd() (in module pynfsa.sampler), VIII

E
evaluate() (in module pynfsa.models), VII
Extractor (class in pynfsa.extractor), X

F
fapply() (in module pynfsa.models), VIII
fit() (pynfsa.models.LinearTransformer method), V
fit() (pynfsa.models.Mahalanobis method), V
fit() (pynfsa.models.Momentum method), VI
fit() (pynfsa.models.PipelineFixd method), VII
fititerate() (in module pynfsa.models), VIII
FlowExtractor (class in pynfsa.extractor), X
Flowizer (class in pynfsa.flowizer), XI
FreqBands (class in pynfsa.models), IV
FreqBaseTransformer (class in pynfsa.models), IV
FreqThresh (class in pynfsa.models), IV

G
get_A() (pynfsa.models.FreqBands method), IV
get_A() (pynfsa.models.FreqBaseTransformer method), IV
get_A() (pynfsa.models.FreqThresh method), V
get_packets() (in module pynfsa.util), XII

H
H5Node (class in pynfsa.dataset), VIII
handle_exit() (pynfsa.dataset.H5Node method), IX

I
int2ip() (in module pynfsa.util), XII
ip2int() (in module pynfsa.util), XII

L
LinearTransformer (class in pynfsa.models), V

M
Mahalanobis (class in pynfsa.models), V
Modeler (class in pynfsa.models), VI
Momentum (class in pynfsa.models), VI

O
opts() (in module pynfsa.util), XII

P
PcapExtractor (class in pynfsa.extractor), X
PipelineFixd (class in pynfsa.models), VII
plot_roc() (in module pynfsa.models), VIII
printTree() (pynfsa.dataset.H5Node method), IX
psd() (in module pynfsa.sampler), VIII
psd1() (in module pynfsa.sampler), VIII
psd2() (in module pynfsa.sampler), VIII
pynfsa.dataset (module), VIII
pynfsa.extractor (module), X
pynfsa.flowizer (module), XI
pynfsa.labeling (module), X
pynfsa.models (module), IV
pynfsa.sampler (module), VIII
pynfsa.util (module), XI

R
retain_fields() (pynfsa.dataset.Table method), IX
reverseDns() (in module pynfsa.util), XII

S
Sampler (class in pynfsa.sampler), VIII
save() (pynfsa.dataset.Table method), IX
scalar() (in module pynfsa.util), XII
scatter() (in module pynfsa.util), XII
score() (pynfsa.models.Mahalanobis method), VI
score() (pynfsa.models.PipelineFixd method), VII
select() (pynfsa.dataset.Table method), IX
set_fields() (pynfsa.dataset.Table method), X

T
Table (class in pynfsa.dataset), IX
timedrun() (in module pynfsa.util), XII
TraceExtractor (class in pynfsa.extractor), XI
transform() (pynfsa.models.LinearTransformer method), V
transform() (pynfsa.models.Momentum method), VI

V
Variable (class in pynfsa.dataset), X
Appendix B.

B Results

Figure B.1: ROC curves of the flow dataset when discriminating `http` from `woodpecker`, `http_tunnel` and `vpn`. Comparison of different dimensionality reduction techniques.
Figure B.2.: ROC curves of the *trace* dataset when discriminating *http* from *woodpecker*, *http tunnel* and *vpn*. Comparison of different dimensionality reduction techniques.

<table>
<thead>
<tr>
<th>Method</th>
<th><em>flow</em></th>
<th><em>trace</em></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em>AUC</em></td>
<td><em>sAUC</em></td>
</tr>
<tr>
<td>M(var,curt,skew)</td>
<td>0.479</td>
<td>0.011</td>
</tr>
<tr>
<td>PCA(dim=3)</td>
<td>0.760</td>
<td>0.085</td>
</tr>
<tr>
<td>PCA(dim=1)</td>
<td>0.616</td>
<td>0.011</td>
</tr>
<tr>
<td>BF(bands=10)</td>
<td>0.323</td>
<td>0.016</td>
</tr>
<tr>
<td>BF(bands=5)</td>
<td>0.468</td>
<td>0.113</td>
</tr>
<tr>
<td>BF(bands=2)</td>
<td>0.712</td>
<td>0.062</td>
</tr>
</tbody>
</table>

Table B.1.: Sample mean and standard deviation of the AUC ROC when discriminating *http* from *woodpecker*, *http tunnel* and *vpn*. Comparison of different dimensionality reduction techniques.
Figure B.3.: ROC curves of flow dataset when discriminating http from woodpecker, httptunnel and vpn. Comparison of discriminative performance between classes.

Table B.2.: Mean and standard deviation of the area under ROC curve. Comparison of discriminative performance between classes.
Figure B.4.: ROC curves of trace dataset when discriminating http from woodpecker, httptunnel and vpn. Comparison of discriminative performance between classes.

<table>
<thead>
<tr>
<th>Method</th>
<th>flow</th>
<th></th>
<th>trace</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AUC</td>
<td>s_AUC</td>
<td>AUC</td>
</tr>
<tr>
<td>PCA(dim=3),Mahalanobis</td>
<td>0.751</td>
<td>0.003</td>
<td>0.760</td>
<td>0.085</td>
</tr>
<tr>
<td>PCA(dim=3),GMM(comp=1)</td>
<td>0.751</td>
<td>0.003</td>
<td>0.760</td>
<td>0.085</td>
</tr>
<tr>
<td>PCA(dim=3),GMM(comp=2)</td>
<td>0.701</td>
<td>0.020</td>
<td>0.830</td>
<td>0.019</td>
</tr>
<tr>
<td>PCA(dim=3),GMM(comp=5)</td>
<td>0.663</td>
<td>0.014</td>
<td>0.821</td>
<td>0.031</td>
</tr>
<tr>
<td>PCA(dim=3),GMM(comp=10)</td>
<td>0.659</td>
<td>0.021</td>
<td>0.827</td>
<td>0.019</td>
</tr>
</tbody>
</table>

Table B.3.: Mean and standard deviation of the area under ROC curve. Effect of Gaussian Mixture Model complexity on discriminative performance.
Figure B.5.: ROC curves of flow dataset when discriminating http from woodpecker, httpunnel and vpn. Effect of Gaussian Mixture Model complexity on discriminative performance.

<table>
<thead>
<tr>
<th>Window width in samples</th>
<th>$AUC$</th>
<th>$s_{AUC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>0.659</td>
<td>0.004</td>
</tr>
<tr>
<td>500</td>
<td>0.744</td>
<td>0.007</td>
</tr>
<tr>
<td>200</td>
<td>0.751</td>
<td>0.003</td>
</tr>
</tbody>
</table>

Table B.4.: Mean and standard deviation of the area under ROC curve. Effect of windowing on discriminative performance.
Figure B.6.: ROC curves of trace dataset when discriminating http from woodpecker, httptunnel and vpn. Effect of Gaussian Mixture Model complexity on discriminative performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>flow AUC</th>
<th>$s_{AUC}$</th>
<th>trace AUC</th>
<th>$s_{AUC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA(dim=1)</td>
<td>0.757</td>
<td>0.003</td>
<td>0.612</td>
<td>0.019</td>
</tr>
<tr>
<td>PCA(dim=2)</td>
<td>0.760</td>
<td>0.005</td>
<td>0.628</td>
<td>0.045</td>
</tr>
<tr>
<td>PCA(dim=3)</td>
<td>0.751</td>
<td>0.003</td>
<td>0.532</td>
<td>0.053</td>
</tr>
<tr>
<td>PCA(dim=5)</td>
<td>0.751</td>
<td>0.004</td>
<td>0.366</td>
<td>0.027</td>
</tr>
<tr>
<td>PCA(dim=10)</td>
<td>0.729</td>
<td>0.005</td>
<td>0.331</td>
<td>0.028</td>
</tr>
<tr>
<td>PCA(dim=25)</td>
<td>0.720</td>
<td>0.004</td>
<td>0.292</td>
<td>0.016</td>
</tr>
</tbody>
</table>

Table B.5.: Mean and standard deviation of the area under ROC curve. Effect of number of retained eigenvectors on discriminative performance.
Figure B.7.: ROC curves of flow dataset when discriminating \textit{http} from \textit{woodpecker}, \textit{http_tunnel} and \textit{vpn}. Effect of number of retained eigenvectors on discriminative performance.
Figure B.8.: ROC curves of trace dataset when discriminating http from woodpecker, http_tunnel and vpn. Effect of number of retained eigenvectors on discriminative performance.
Figure B.9.: Semilogarithmic plot of the eigenvalues using the flow and trace datasets.