TALLINN UNIVERSITY OF TECHNOLOGY

School of Information Technology

Department of Software Science

ITC70LT

Neslisah Celik 156340IVCM

ANOMALY DETECTION USING LOCKED SHIELDS LOGS

Master thesis

Prof. Dr. Olaf M. Maennel

Tallinn 2017

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Neslisah Celik

Abstract

Locked Shields is a world-known prestigious cyber security exercise which is hosted by NATO CCD COE every year in Estonia. The game aims to enhance security awareness of international participants in cyber domain. Participant teams should put effort into various categories such as media, legal aspects and forensics as well as protecting their systems against attacks which are vital part of the game. The winner of Locked Shields is determined based on various qualitative and quantitative observations during the game. Qualitative data is composed of general observations which were made by decision-making authority during the game while quantitative data consists of various logs and reports. However there is an accuracy problem in quantitative data which leads to score participant teams incorrectly and thus, causing misconceptions on determining the winner. My dissertation addresses the problem in quantitative data and proposes an anomaly detection inspired methodology on how to mitigate it.

This thesis is written in English and is 59 pages long, including 4 chapters, 16 figures and 4 tables.

Annotatsioon

Locked Shields on ülemaailmselt tuntud prestiižikas küberkaitseõppus, mida korraldab NATO CCD COE igal aastal Eestis. Õppuse eesmärk on rahvusvaheliste osalejate küberturbe alase teadvuse suurendamine. Osalevad meeskonnad rakendavad teadmisi mitmes valdkonnas, sealhulgas meedia, seadus, kriminalistika ja süsteemi kaitsmine erinevate rünnakute vastu, mis on mängu oluline osa. Locked Shields õppuse võitja valitakse erinevate mängu jooksul tehtud kvalitatiivsete ning kvantitatiivsete tähelepanekute alusel. Kvalitatiivne informatsioon koosneb üldistest tähelepanekutest, mis tehakse mängu ajal otsustava kohtuniku poolt. Kvantitatiivne teave koosneb mitmetest raportitest ja logidest, kuid ei ole piisavalt täpne, mis põhjustab eksitusi võitja väljaselgitamisel. Minu väitekiri käsitleb kvantitatiivse teabe täpsuse probleemi ja pakub välja anomaaliate tabamisest inspireeritud meetodid selle lahendamiseks.

See tees on kirjutatud inglise keeles ja on 59 lehekülge pikk, sealhulgas 4 peatükki, 16 arvud ja 4 tabelit.

Table of abbreviations and terms

LS	Locked Shields
CERT	Computer Emergency Response Team
ENISA	European Union Agency for Network and Information Security
NATO CCD COE	NATO Cooperative Cyber Defence Centre of Excellence
ISACA	Information Systems Audit and Control Association
ICSA	International Cyber Security Academy
IDES	Intrusion Detection Expert System

Table of contents

List of figures

List of tables

1. Introduction

Internet has arrived with obscurities and sometimes dangers in our world as well as its fascinating life-saver features. That is because; governments and enterprises are increasingly getting to share more information and services in cyberspace which is also available without any restrictions for anyone who is "smart enough". [1]

Considering examples such as so-called digital weapon STUXNET or global espionage network GhostNet, cyber warfare is becoming a big threat agent for countries. This increases the need for cyber-skilled professionals as well as strong defense infrastructure. Therefore, security staff requirement on national level is becoming a popular area of asset to be invested in. According to CERT inventory of ENISA as of July 2016, there are 35 registered national CERT teams in Europe. [2]

The way of having desired skills in cyber domain and not getting impaired in them is going through a decent and equipped practical training in first place. There are many institutions proving certified cyber security trainings on international level such as ISACA, ICSA and APMG.

One of the most important cyber security training exercises, Locked Shields, is hosted by NATO CCD COE every year in Estonia. It is the world's largest and most advanced international cyber defence exercise which aims at enhancing security awareness between NATO, NATO allies and partners in cyber defence. [3]

Participants from multiple NATO nations take place in different missions in Locked Shields. They are mostly consisted of experts and specialists from governmental organisations, military units, CERT teams and private sector companies. [3]

Each year a different attacking scenario is defined and there are mainly 5 different team categories to take different roles. They are blue, red, white, green, yellow and legal teams.

The winner of LS is decided upon 5 major evaluation criterias which are examined by White Team. These criterias are named as attack, availability, injects, reporting, usability and they form general LS scoring result. Overall team evaluation by White team during the game, a final report including the information of attacks that were performed by Red Team and scoring results are main inputs to White Team for decision making process.

Availability scoring results are constructed by aggregating network traffic packets that were sent by scoring bot servers to blue team servers, in order to check whether servers are available or not. So, if the scoreboard labels a team's server "down" , corresponding team loses points. However, availability scoring data (hereinafter referred to as scoring data) is not validated by any conforming data in order to assure whether the server is actually functioning or not when it is scored as down. Conforming data is meant to be network pcap logs of Locked Shields game.

Consistency between scoring data and network pcaps is a necessity because both data sets should be enhancing the qualities of each others in case of any inaccuracy or insufficiency in logs. However, there is not a correlation between these data sets in order to verify whether they conform to each other or they do not. As a result, there is an open-ended and unvalidated trust in scoring data which has a critical impact on the process of determining the winner of Locked Shields game. For example, if scoring data indicates that a server is not accessible by sending ICMP requests at a time, one can not be sure whether the server was really functioning or not at that time.

In this paper, I suggest using anomaly detection inspired methods to provide a scoring data validation. Since Locked Shields data sets are explicitly one of a kind comparing to other data sets that were used in anomaly detection methods in literature, it is an essential factor to review them and come up with a customized method for Locked Shields. Implementation results obtained during thesis work shows that the proposed method is able to detect anomalies as points to be improved in interpretation of scoring data. Because detected anomalies are suspicious timestamps where network pcaps and scoring data do not conform each other. In other words, what scoring data indicates is disputable at that detected anomalous point hence a further investigation is needed to break down the case and to identify countermeasures if needed.

2. Foundations

This chapter starts with the definition of Anomaly Detection and general literature overview. The following parts discuss various Anomaly Detection methods and their advantages.

2.1. Anomaly Detection

Organizations are becoming more vulnerable and insecure against various cyber threat agents as Internet becomes more accessible and more information is hosted in cyber domain. Also, new and different sort of attacks, which are also need to be covered in defense scope, are added in cyber domain almost each year. For instance, according to Kaspersky Security Bulletin of year 2015, there were 121,262,075 unique malicious objects including scripts, exploits, executable files etc. detected in 2015. [4] Therefore, having robust and trustful countermeasures on the detection process of cyber attacks has been increasingly going on.

Authentication and authorization controls are applied in first place in order to prevent unauthorized accesses and privilege escalations in systems. This type of countermeasures form the first level of the security in an organization's network. However, changing trends in cyber domain require additional security mechanisms on extra levels. Therefore; firewalls, vulnerability scanners and IDS/IPS softwares were constructed as second level countermeasures of security complementarily.

IDSs are capable of detecting malicious network traffic, host-based attacks including privilege escalation, unauthorized login and accesses, malwares and data-driven attacks on various applications which lead to privilege escalation most of time. [5] Generally, IDSs work like a typical network sniffer and they are expected to detect attacks being or after occurred despite all the countermeasures in the system and report back on findings if any rule violations occurred.

IDSs are classified in two main categories which are host-based and network based. Host-based IDSs can only analyze the network traffic of a machine on which IDS is hosted while network based IDSs can analyze the whole network traffic.

Most of IDSs are based on signature-based detection. However such methods can only detect previously known attacks and malwares that have a corresponding signature in

signature database. [6] This requires a cumbersome periodic updating process of signature database for system administrators. Another drawback of signature-based detection systems is not maintaining the state information of signatures if attacking activities are spreaded into discrete events. Because this means, corresponding signature is also dispersed among multiple packets. [5] These drawbacks have directed people's interest to data-mining based intrusion detection methods consequently.

Anomaly Detection is a data-mining based intrusion detection method. It refers to the process of finding patterns which do not conform to the expected profile. [7] Under this definition, the scope of anomaly detection includes anomalous behaviors coming from not only external threat agents but also from authorized users inside network. It has been mentioned in intrusion detection domain ever since it was proposed by Denning in 1987. [8] Proposition consisted of a model of real-time intrusion detection system which is capable of detecting external break-ins as well as misuses by internal users in a specified network. [8]

As the first step of anomaly detection, baseline profile or also known as normal profile of the network is created. Thenceforth, any network activity that do not conform the baseline profile is treated as an anomaly. [5] Anomaly detection systems bring out several benefits to intrusion detection. First, they are not only able to detect incoming external attacks but also internal attacks. For instance, if an internal user or someone who uses a privileged account starts performing actions that deviate from the baseline user-profile, anomaly detection system generates an alert and reports it. [5]

Second, an anomaly detection system is based on pre-learnt baseline profiles. Therefore it is very difficult for an attacker to know what malicious activity can be performed without generating an alert.[5] Lastly, an anomaly detection system is capable of detecting previously unknown or zero-day attacks. Because it does not use an intrusive activity profile or signature in order to detect attacks. Instead of this, any activity that differs from normal profile triggers the alert system. [5]

2.2. Anomaly Detection Techniques

The main assumption of anomaly detection is based on hypothesis that Denning set forth: "*Exploitation of a system's vulnerabilities involves abnormal use of the system; therefore, security violations could be detected from abnormal patterns of system usage*." [8] Therefore in ideal case, we expect a 100% match of detected anomalous

activities and malicious activities so that we would not have any false-positives or false-negatives.

However, anomalous activities are not always covered by malicious activities. This state of mismatching leads us to have different possibilities of anomaly detection results. There are four possibilities of having different results, each with a non-zero probability [9] :

- **Intrusive but not anomalous:** There might be cases where system fails detecting a malicious activity. These is cases are named as false-negatives. [5]
- **Not intrusive but anomalous:** There might be cases where system detects an anomalous activity which is in fact was not malicious. These cases are named as false-positives. [5]
- **Not intrusive and not anomalous:** This is the case of system not detecting a non-malicious activity as expected. These cases are named as true-negatives. [5]
- **Intrusive and anomalous:** This is the case of system detecting a malicious activity as expected and these cases are named as true-positive. [5]

An anomaly detection approach usually consists of three phases in each technique: a parameterization, training and a detection phases. On parameterization phase, observed system metrics are presented in a pre-configured form. Later, on training phase, the baseline traffic profile is defined according to the metrics. Finally on detection phase, baseline profile is applied to newly incoming data and in case of any deviations detected above threshold, an alert is triggered. [5], [10]

There are different classifications of anomaly detection techniques proposed in the literature. Lazarevic [11], classifies techniques based on employed anomaly detection algorithms and sets out 5 categories. These categories are statistical methods, ruled based methods, distanced based methods, profiling methods and model based approaches. [11] On the other hand, according to the type of data processing, Patcha and Mark [5], set forth 3 main groups. These groups are statistical-based, knowledge-based and machine learning-based anomaly detection techniques. [5] There are also different subclasses under each 3 main classes and each subclass involves one or multiple implementations of theirs. Since Patcha and Mark's 3-groups scheme is more comprehensible for me comparing to other classification and survey studies in the literature; I have preferred to gather my review of related work under their classification scheme.

2.1.1. Statistical Anomaly Detection

In statistical anomaly detection, baseline network profile is created by captured network traffic. This baseline profile is probabilistic and based on univariate or multivariate parameters ,which have been defined on parameterization phase, such as data volume per packet, packet types, packet loss, number of different source IP addresses, etc. In this context, a current network profile is also constructed based on incoming network traffic in accordance with the baseline profile structure. As network traffic occures, current network profile changes and an anomaly score is calculated by comparing current network profile and baseline profile. Anomaly score can be calculated for either each parameter or combination of multiple parameters. The score indicates the degree of irregularity for whatever parameter it was calculated for. [11]

IDES is the earliest application of statistical approach which was set forth by Denning and Neumann. It was designed to detect various types of intrusions including attempted break-ins, masquerading, penetration, access violations, denial of service attacks, data leakage, and side-effects of virus, worms and similar malicious programs. [12]

IDES was a system independent intrusion detection system which follows the basic approach of monitoring system activity as it is logged at the same time. It consisted of 6 main components which are subjects, objects, audit records, profiles, anomaly records and activity rules. Subjects and objects are activity initiators and resources which are affected by activities respectively. Audit records are generated activity logs which are triggered by any activity performed by a subject on an object. Profiles represent network activities performed by subjects. Anomaly records are generated information if an anomalous behaviour is detected. Lastly, activity rules are any actions taken in case of any pre-defined condition is satisfied in IDES. These actions to be taken are updating profiles, detecting anomalous behavior and reporting. [12]

IDES observes system activities to identify activity profiles per subjects. Observations are obtained by audit records. An activity profile to be identified represents the behaviour of a given subject with respect to a given object. Behaviour is a statistical model constructed by a statistical metric which is a random variable x that is accumulated over a period. [12]

Statistical model does not need to have any information related to distrubition of x beforehand. It gathers all needed information of x via observations, determines anomaly threshold by itself and use this information to determine whether a new observation x+1 is an anomaly or not. As an example, we can think of a baseline profile for User X daily

logging in his Windows domain account. Here User A is the subject and his Windows account is the object while Windows login is the activity. If we assume we want to see anomalies based on a daily period, In this case, x would represent daily login activity of User A in total which is performed via his Windows domain account and we expect activity x+1 to be detected if it is anomalous.

IDES has 3 types of metrics which are event (audit record) counter, interval timer and resource measure. Event counter represents number of events satisfying a condition during a period. Interval timer is the difference between respective audit records and resource measure is the quantity of resources consumed by an action during a period. Given one or multiple of the metrics, there are several statistical models that can be included in IDES. [5]

Haystack is also one of the earliest implementations of statistical based anomaly detection approach which targets intrusions in multi-user air force computer systems. Although its model is a modified version of IDES; Haystack is a custom-built IDS system for a Unisys mainframe running the OS/100 operating system unlike IDES. Eventually, it is designed to detect similar intrusion types with IDES. [13]

Conceptually, Haystack interacts with 3 external components which are Unisys operating system on execution, system security officer and a database management system on analysis phase. It has 2 operational clusters on Unisys and a IBM Zenith Z-248 PC which is in charge of system security officer. [13]

First step of detection starts on Unisys. Haystack preprocessor on Unisys extracts audit records from operating systems's audit trail logs, parses them in accordance with predefined abstract elements which would specify an audit trail event and converts them to the required CAT file format. On the other hand, Haystack maintains a database in order to store user groups which are based on user profiles. A session history record based on an user profile is created if any login event is encountered for the user as the CAT file is being read on Zenith. An user's session hisory record is updated each time an audit event by the user is encountered in the CAT file. Haystack analyzes newly created sessions by statistical and pattern-based approaches in order to detect predefined anomalous behaviors. [13] There are two types of anaysis held in Haystack. First one is detecting whether a user's new session history record is deviating dramatically from user's previous session history records or not. This detection is performed by trend analysis which refers to statistical approach. Second analysis is comparing user's new

session history record to a predefined anomalous behaviour -in this case this is an intrusion- which refers to pattern-based approach. [14]

Statistical approaches have a number of values. Firstly, they do not require any knowledge related to expected system activities or behaviour beforehand. They learn the normal behavior of system by observing network traffic. [11] They also do not need to know any prior knowledge about system breaches or attack information which makes it possible for them to detect zero day attacks [5]. Statistical approaches also can provide accurate information on attacks occur over large amount of time such as portscan activities. However, statistical approaches have also some disadvantages. First of all, threshold calculation might be difficult in order to balance likelihood of false-positives and false-negatives. In addition, since the technique is based on modelling behaviours, not all the system behaviors might be eligible to be modelled. [5] Therefore, either you might end up with inaccurate modelling or results that make no sense. Statistical systems might be also tricked to accept anomalous behaviour as normal by skilled adversaries [5]. This is a critical drawback for real time anomaly detection systems in particular, since adversary has a direct communication with system which gives him an opportunity to get feedback on his actions immediately.

2.1.2. Machine Learning Based Anomaly Detection

Machine learning is a computer science methodology in which a system or a program can train itself to improve on a certain task over time. Machine learning techniques in anomaly detection focuses on achieving better results and performance based on previous results. Systems using machine learning techniques are capable of learning and adapting new execution strategies based on learnt data. However since machine learning methods require excessive resources comparing to the other methods, they are rather expensive. Also, if we consider the large amount of data and traffic frequency of nowadays computer and network sytems, this method is not scalable for real time anomaly detection operations either. [5] There are different types of machine learning techniques in the literature:

System Calls Sequence analysis

User profiles are described as a whole set of related process elements such as login location, resources consumed, typing rate, counts of used commands in computer security aspect. However, Lane and Brodley [16] state that being dependent on only a set of elements for profile description ignores the fact that human/computer interaction is essentially a causal process. They note that if a user is interacting with the computer,

he does this for a certain aim. He types commands in order to achieve what is on his mind and this leads computer to act accordingly. According to their hypothesis; a user behaves similarly to similar events and this causes a sequence of similar actions which is called characteristic sequences. Therefore, they intend to distinguish a normal user from an adversary by differences in these characteristic sequences. [16]

This technique is called as sequence analysis based anomaly detection. It is essentially relies on the fact that every program produces a system call or a set of system calls for operating system in order to make operating system work as it needs and anomalies are detected if any deviation from a regular system call behavior occurs. [15]

Bayesian Networks

This technique benefits from the principle of Bayes theorem which is calculating the probability of an event based on given conditional probability distributions of other relevant events. Bayesian network approach is considered as a classification based anomaly detection technique [17]. Therefore this method can also be categorized as a classification based technique under data-mining approaches. However, since it uses the Bayes theorem which is a branch of machine learning in particular at core, it is discussed under machine learning based approaches.

Bayesian network approach has same 2-step - training and testing - fashion. On training step, algorithm learns a classifier (model) by using training data. On testing step, incoming test instance is classified whether it is normal or anomalous by using the classifier derived on training step. [17]

Bayesian networks are statistical networks in which variables are represented as nodes while probabilistic dependencies or relations between variables are represented as directional arrows [18]. Broadly, a Bayesian network consists of two parts. One part includes graphical representation of nodes and directional arrows which reflect relevant variables and their relations. Second part includes conditional probability distrubution views belonging to variables represented in first part.

In Bayesian learning approach, system is formulated probabilistically. System may consist of many classes. First, all qualitative facts known regarding each class is modelled. This facts might be distribution forms, independence assumptions, distribution confidences etc. This reveals unknown elements of classes. Then, a prior probability distribution of unknown class elements is specified by using probability distributions of other elements related to unknown elements. This step is mainly an assumption step performed before gathering the actual data in order to be able to

represent unknown elements in the model. After gathering data from system, a posterior probability distribution of unknown elements is calculated based on specified model. Later, posterior distribution data is used to predict new prior values more accurately. [17]

Kruegel et. al. [19] set forth a model which replaces threshold-based decision process with a Bayesian network. They implemented an intrusion detection system to detect attacks against deamon applications and setuid programs running on Linux or Solaris operating systems by analyzing operating system calls. [19]

As first step, system analyzes individual system calls and their arguments; transforms them into input events and represents them as feature vectors. These vectors are evaluated by 4 different anomaly detection models and an output value denoting the deviation of event features from baseline profile is produced accordingly. Anomaly detection models are used to characterize different argument types and each model has a confidence value represented by a node connected to its relevant model node. [19]

Model confidence is also important in event classification. Because if a model has a high confidence for its output, anomaly score derived by this model clearly will affect final decision more than a model with low confidence. Confidence levels are very high, high, medium, low and none. [19]

During the analysis of whether a system call is anomalous or not, the output of four models and their confidences are given as input to a Bayesian network. Later, they are aggregated by a simple event classifier and anomaly probabilities are calculated. If calculated probability exceeds threshold, an alarm is given. Kruegel et. al. also note that implementation of this Bayesian event classification scheme proposed by them decrease false-positive anomaly alarms significantly. [19]

One of prominent values of using Bayesian networks in anomaly detection is being able to handle situations where you miss data. Because Bayesian approach is capable of threading interdependencies between variables. In addition, since Bayesian networks can represent causal relations, result of represented event might be predictable. Also, the technique allows users to combine prior knowledge with data if it is needed. [5] Tying up models with confidence values in the implementation of Kruegel et. al. is a good example of benefiting from this advantage.

Principal Components Analysis (PCA)

Datasets that are used in anomaly detection are typically very large and multidimensional. Therefore, traditional statistical methods break down easily due to

having the problem of large dimensional datasets. Additionally, since data is represented as multiple variables in each dimension, variables that are important for anomaly detection can not be identified clearly in many cases. [21]

Many dimensionality reduction solutions have been developed in order to tackle this problem. PCA is noted to be the best linear dimension deduction technique in the sense of average of set of errors (mean-square errors). PCA is also named as singular value decomposition (SVD), Karhunen-Loeve transform, the Hotelling transform and the empirical orthogonal function (EOD) in various fields. [21]

PCA algorithm is based on expressing the dataset with its the principal directions which are obtained by calculating the most dominant eigenvectors of data. Y. Lee, et. al. [22] proposed a PCA based anomaly detection method where PCA was used in order to detect outliers of the data set. They claim that anomaly of the target data set can be determined according to the variation of the resulting dominant eigenvector. Their algorithm studies variation of principal directions when a new instance is added or removed from the data set and it observes that the principal direction is deviated when an outlier instance is added. Therefore algorithm is able to determine the outlierness of the target data. This calculated score of outlierness represents the threshold value in anomaly detection process. [22]

Microsoft also offers a PCA-based anomaly detection module in their Azure machine learning platform. Users are able to generate their untrained (baseline) anomaly detection model by specifying model parameters, number of parameters will be used in PCA and a desired PCA method on Azure UI. Generated model later can be trained via Trained Model module and anomalies can be predicted by Score Model module. [23]

Markov Models

Markov models are defined as stochastic models in probability theory. According to the theory of Markov model, the probability distribution of the state at a given future time does only depend on the states at current time. It does not related to the previous states leading to the state at current time. [24]

There are two main approaches of Markov models. They are named as Markov Chains and Hidden Markov Models. Markov Chains are fully observable stochastic processes in discrete-time. In other words, if a system state at t+1 time is only depending on the state at t time, system would be modelled as a Markov Chain. [24], [25]

Hidden Markov Models (HMM) are stochastic processes with another underlying stochastic process which is not observable. L. R. Rabiner et. Al [25] gives a simplified

example to explain HMM: assuming you are in a room with a curtain through which you can not see what is happening. On the other side of the curtain another person is flipping a coin for multiple times. Coin flipper will not tell you what he is doing exactly but he will only tell you the result of each coin flip. Thus a sequence of coin flipping events that would be representing the underlying stochastic events, occur but you only observe the results which would be representing stochastic events on shallow. [25] There are anomaly detection studies that implement both approaches in literature.

Ye [24] proposed an anomaly detection technique where Markov Chain approach is used to model a temporal profile of an expected behavior in a computer and network system. Ye states that whatever activity it is, both anomalous and normal events contain sequences of computer actions and this actions induce us to have corresponding audit events in a monitored system. Considering states of a system can be represented by the types of different audit events; temporal behavior of the system can also be represented as stochastic process in discrete-time. Therefore, Markov Chain model of temporal behavior is built by learning from previous audit events that were collected during the expected usage of the system. This model is the baseline profile of the system. After that, a temporal behavior of recent past in an observation windows of size N is created based upon continuous audit events. For each audit event in the window at a specific time, audit event types and corresponding sequence of states are obtained. The probability of baseline model supporting the sequence of states is calculated by obtained values. The higher the probability is calculated, the more likely that sequence of events belongs to a normal activity. Anomalous activities are expected to have low supporting probabilities since they would be deviating from baseline profile. [24]

Hidden Markov Model approach is proposed by Srivastava et. al. [26] in order to detect credit card frauds. Assuming a fraud detection system (FDS) is running in a credit card issuing bank, every incoming transaction is analyzed by FDS based on card details and the value of purchase. They begin modelling with identifying observation variables. They define 3 different price ranges: low, medium, high and sequence of transaction amounts based on the type of purchases. Because each individual transition amount depends on the type of purchase. The type of each purchase is linked to corresponding merchant's line of business. However, the card issuing bank running FDS doesnt have any information on the business lines. Therefore, the type of purchase is hidden from the FDS. All possible types of purchase and merchants' lines of business represent the hidden states of HMM while amount of purchase and price ranges represent shallow

states. A HMM is built and trained for each credit cardholder so that baseline profiles are constructed. After that, just as in Markov Chain approach, a new set of sequence of transaction amounts in a time amount of t is collected from an incoming set of cardholder transactions in accordance with observation varilables which are defined in the beginning. New sequence is given to HMM and probability of acceptance by HMM is calculated. The transaction is considered as fraud if the probability value is lower than threshold. [26]

2.1.3. Data Mining Based Anomaly Detection

Data mining is essentially finding the regularities and irregularities which may not be seen with naked eye in large data sets. [5], [27] Considering the increasing volume of nowadays network traffic, intensity as well as attacking methods, collected data sets for anomaly detection has been questioned in a sense of adequacy and accuracy. This situation has led scientists to look for different solutions. Using either standalone data mining approaches or combining with other approaches for anomaly detection is one of these solutions. [5], [27], [28]

Data mining approaches can contribute followings to an anomaly detection system:

- Eliminating false-positives to allow specialists to focus on real anomalies
- Identifying false-positive generators
- Identifying long, ongoing and unknown anomalous activities
- Eliminating manual and ad-hoc elements from the process of building an anomaly detection system. [5], [27], [28]

There are many different data mining techniques that have been proposed in the literature.

Classification-Based Intrusion Detection

An anomaly detection system which classifies data based on a specific set of rules or configuration whether the data is anomalous or not, can be named as a classification-based intrusion detection system. Once system identifies class attributes and classes and then learns a baseline profile, it later classifies the unknown data samples according to previously learned profile. There many techniques derived in previous related works. These techniques include inductive rule generation, fuzzy logic, genetic algorithms and neural networks. [5], [29]

Inductive rule generation algorithms, which is the first among classification-based techniques, generally consist of application of a set of association rules and frequent patterns to classify monitored data. As an example of association rules it can be stated

that if event X occurs at t time, the most likely event Y will occur also. In this situation X and Y are described as {variable:value} pairs and the aim is to find data sets Xs and Ys by value classification where X and Y have this variable:value association. [5], [29]

Some inductive rule generation algorithms first construct a decision tree to extract rules out of it while some of them directly extract rules from data set via a divide-and-conquer approach. Although rules are supposed to be as simple and nonrigid as they can; this also might cause them to be difficult to manage since they tend to be unstructured. [29]

Another classification technique is using fuzzy logic algorithms. Fuzzy logic algorithms are based on larger levels of uncertainity. In this context, if A is a fuzzy data set and x is a relevant object; statement of "*x is a member of A*" is not necessarily true or false. It might be true only to some degree and the degree might be enough to indicate that x is actually a member of A. [30] The main benefit that fuzzy logic contributes to anomaly detection is providing a normalization of quantitative measures. Because since many security related features are quantitative, normal behavior patterns that have been constructed based on these values might give false-positive anomaly alarms even if incoming activity is not in pattern's direction but conforms expected behavior at some point. Normalization would eliminate such issues. [31]

Fuzzy Intrusion Recognition Engine (FIRE) have been proposed by Dickerson et. al. [32] as an anomaly detection system that uses fuzzy logic. FIRE combines simple network traffic metrics with fuzzy rules. It collects raw network data with its own network data collector (NDC), merges and mines data with its own network data processor (NDP). After completing data mining process, NDP produces fuzy input sets based on historic input data of each data element. All fuzzy input sets are combined by Fuzzy Threat Analyzer (FTA) and finally fuzzy anomaly alerts are given in accordance with fuzzy rules that are needed to be defined by security administrator after fuzzy input sets are created. There is no baseline profile created in FIRE. Instead, fuzzy logic rules applied to data to classify whether data is anomalous or not. [32]

Another method of classification technique; genetic algorithms (GA), employs the metaphor of biology in order to evolve a population of initial individuals iteratively to a population of high quality individuals. A typical genetic algorithm begins with selecting a number of best individuals based on a defined fitness function. Rest of individuals are discarded. Next, a number of individuals are selected and paired with each other to produce one offspring by partially exchanging their genes. Selection, pairing and

producing offspring phases are repeated iteratively and in the end, the mutation operations are applied to a number of individuals selected. This mutation operations usually mean individuals changing their values abruptly. [33], [34]

The major reason of GA being used in anomaly detection is that data search process becomes faster and more accurate due to GA being a flexible and a robust searching method. Additionaly, a genetic algorithm search approaches to a solution from multiple directions based on its probabilistic rules instead of deterministic ones. This feature helps eliminating false-positives and false-negatives in anomaly detection. [5], [34]

There is also another variation of GA that is named as genetic programming (GP). The difference between GA and GP is the way of encoding individuals. GA uses fixed length vectors to encode individuals while GP uses parsing trees where leaf nodes are genes and non-leaf nodes are primitive functions such as AND, OR etc. Therefore, GP can derive more complex and accurate rules than GA can. [34]

The first application of genetic algorithm approach in anomaly detection has been proposed by Crosbie et. Al. [35] where GP is used to derive new anomaly scenarios from prespecified scenarios. They aim to create an IDS which is capable of changing over time to hold new threat patterns and customize itself accordingly. In this context, they divide an IDS into multiple functional entities which they call "multiple autonomous agents" to have multiple IDSs working simultaneously. Next, they use GP to evolve and replace agents continuously in accordance with new threat patterns in real time. [35]

Final classification based technique to cover is neural networks. A neural network consists of simple processing nodes and connections between them. The connection between any two nodes has a weight. The weight indicates that how much nodes will affect each other. Nodes are divided into two sets: input and output nodes. Network performs a mapping operation between input and output nodes once a value is assigned to input nodes. This mapping is stored in the weights of the network. [36]

The goal to use neural networks in anomaly detection is mostly to be able to classify real time data as being anomalous or not. It also provides deduction from an incomplete data set. Gnosh et. al. [36] describes a study in using neural networks for anomaly detection. They implement a backpropagation network where input value is submitted to the network and the values for each level of nodes are cascaded forward. A normal data set is located in input nodes of the network on training phase. On performance phase, a randomly generated data as input value is given to the trained network and

network generalizes all anomalous data by default under favour of prelocated normal data it stores. [36]

Another reason of why neural networks is preferred to use in anomaly detection is to reduce the high rate of false-positives. As an example, the work proposed by Ryan et. al. [36] aims to develop a neural network based anomaly detection model that would accept newly incoming legitimate behaviors instead giving a false-positive alert for them. They presented neural network intrusion detector (NNID) which is a trained neural network to identify users based on the commands they use during the day. NNID is run by a security specialist to see which users have deviated from their normal behavior at the end of the day. First, audit logs of each user are collected for several days and a vector specifying how often a user executed each command is formed accordingly. This phase outputs training data for the system hence as second step, neural network is trained based on it. On the final step, trained network identifies users for each newly incoming command vector. If network suggests another user different from the actual user in previous vectors or it doesnt even suggest any user; an anomaly is given. [5], [36], [37]

Although neural networks do a good job in predicting; they can be slow and expensive in sense of training. Because network has to recalculate weights each time for a new random data set. Moreover, the network might not find a proper result due to insufficient input data. [5], [36], [37]

Clustering and Outlier Detection

Clustering is a method of aggregating data instances and has been used in anomaly detection before. Portnoy et. al. [38] proposed a clustering method based on distance between data instances and clusters. According to the method, essentially, if there is no cluster in the set, a new cluster is created and data instance is assigned to it. If there are present clusters, data instance is assigned to the cluster in minimum distance. Once there is no data instance left in the set, clusters are labeled as being anomalous or not. Labelling is based on the assumption that clusters contain the largest amount of data instances are considered as normal data so that rest of clusters are anomalous. On testing phase, a randomly given data instance d is clustered based on its distance to the closest cluster and is classified according to the cluster label. [38]

Outlier detection is the process of detecting data elements which dramatically inconsistent or different comparing to a regular data set. Generally, outliers are supposed to be single points. However, a group of anomalous elements migh form small

clusters that are also needed to be pointed as outliers. [39] Clustering is very related to outlier detection, since every anomalous cluster in a data set would be a possible outlier. As a matter of fact, Duan et. al. [39] presented a cluster-based outlier detection method where outliers are detected by clustering algorithm LDBSCAN [39]. The algorithm finds clusters based on local outlier factor (LOF) of each data instance. LOF value indicates a degree of an instance being an outlier. [40]

There is also another related study described by Ertoz et. al. based on outlier detection. They introduce an anomaly detection system named Minnesota Intrusion Detection System (MINDS) where anomaly detection module of MINDS assigns a LOF value to each data point. LOF value considers the density of neighborhood around the data point in order to determine its outlierness. In this context, outliers tend to have high LOF values. [41]

Association Rule Discovery

Association rules are statements of events related to each other. For instance "*X% of customers buying Y product also buy Z product*" statement can be considered as a proper association rule template. In contrast to inductive rules that are mentioned in classification based techniques, association rules do not deduce anything from a present value. They just indicate a present association. Association rules are derived from large data sets by using different mining algoritms. The goal is to obtain multi-attribute correlations from a large data set [28]. Agrawal et. Al [42] proposed an algorithm which decides on which data items to be taken into account to create rules and pruning unused items. Additionally, Park. et. al. [43] described another association rule mining algorithm that is based on hashing. Algorithm utilizes a hash method in order to generate a subset in a large data set iteratively and prune unused items in subsets. [43]

Association rules are important in anomaly detection domain since they can be used to provide a summary of anomalous connections detected. Because program executions and user activities have consistent correlations between the system. Therefore, these correlations can be represented by association rules [5], [16]

Lee et. al. [28] implemented an assocation rule algorithm in order to collect training data set and to create a baseline profile in an anomaly detection system. Algorithm computes a new rule set and updates the existing rule sets each time from the audit trails of the monitored program that was run under different settings. If it finds an exact match of new rule set in existing ones, it increments a match_count value. If it does not find any match, it adds a new rule and sets match_count value to 1. When all rule sets

become stable, training data is generated. Training data is pruned by eliminating the rules with low match_count values comparing to a threshold value that is predefined by the user, and thus final baseline profile is created. [28]

3. Contribution

This chapter includes the deduction of the method that is made based on problem statement&related work and implementation details that were conducted based on deductions.

3.1. Deduction

Scoring data consists of different types of network traffic logs including ping, imap, smtp, http, dig etc. for each server. Each traffic type has a different way of acting and has different metrics to indicate the result of their act. For example, while pinging, several echo requests are being sent and echo replies are being received in return; but the result is given as percentage of packet loss in total. On the other hand, for HTTP traffic, a simple HTTP GET request is sent and the result is shown as the corresponding HTTP status code if the request gets replied.

Having such variety in scoring data inhibits defining a single baseline profile and using a single anomaly detection method for all of it. Therefore, each traffic type has to be examined in a separate way at first to see if the base traffic data is accurate or not. Next, more complex scenarios including multiple traffic types can be considered.

Considering some of scoring data logs are not the actual network logs but an altered or aggregated form of them in order to get more compact and user-friendly results; baseline profiles of network traffics should be defined in a probabilistic approach. Because for example, maybe we know that only one HTTP request was sent to check if HTTP service is available but we do not know how many ICMP requests were sent to get 57% of packet loss for checking server availability. In this context, if the scoring log is the original network log as HTTP example in previous statement, probabilistic modeling will not wreck its accuracy; but if it is not the original as ICMP requests, probabilistic modeling will help us to get a better assumption of its accuracy since we approach it in the same way it was constructed.

On the basis of the anomaly detection, a normal profile is constructed first and then anomalies are detected accordingly. However, since Locked Shields data is already anomalous due to including attacking logs that were made by Red Team hence are expected to have; it is a need to customize traditional anomaly detection approach. In

this context, after normal cases per traffic types are created; anomaly cases based on normal cases should be also created based on normal cases. This would eliminate false-positives which would be an expected anomalous behavior such as a Red Team attack. After anomaly cases are defined, numerical metrics that will form the anomaly in each case are identified out of scoring data. If there is not an extractable data to be used as a metric, a new metric that is indicating the anomaly for that specific case can be built. For example in HTTP example, logs with 200 status code can be represented as just 200 or it can also be composed with another value such as time gap between sent and received packets. It depends on users' way of thinking.

Once scoring data metrics are defined, metrics of network pcap should also be defined as binders. Because we need correlating factors between two separate data sets to check on them by each other. Metrics should be clear, understandable and state the anomaly precisely since anomaly detection holds a large rate of false-positivity in its nature already and thus, we want to have false alarms as less as possible.

Defining anomaly cases beforehand may not make any difference for simple cases however; for complex cases including various traffic types and logs; this is an essential step to get more precise results without having false-positives. Once anomaly cases and metrics are created, one can start looking for anomalies through the network pcap files based on determined metrics.

As given in related work section, after a baseline profile is constructed; anomaly detection is either a real-time process or it requires to have enough of new traffic to be able to detect anomalies over them offline. However, for Locked Shields, anomaly detection is a matter of searching for defined anomaly cases in pre-collected real network logs to see if they occurred or not. This is done so for two reasons.

First, scoring data format is not convenient to create one common normal profile as it has different traffic logs and we need to check on each of them to see if they are reliable. Moreover, there is not enough scoring data to construct an entire, robust normal profile to pinpoint anomalies in such a short time (two days) either when Locked Shields takes place. Because, for several anomaly detection studies [46], [47], [48], [49], sample data sets that were distributed by 1998 DARPA evaluation which was performed by MIT Lincoln Lab, were used to construct a normal profile. These data sets were taken from a military network with a various types of injections into the network over 7 weeks and yet, results were not successful in detecting anomalies at a high rate. [45] Therefore, although a normal profile was constructed based on historic data

perfectly somehow and was tried to be validated during LS 2017 on live; two days of network traffic still would have not been enough for a reliable validation.

In following sections, demonstration details of the proposed method are described over a case study. The first step of the methodology begins with identifying baseline cases which are present and extractable from the scoring data. These cases define the regular and expected behavior, which also refers to the first step of anomaly detection: defining baseline profiles. We have several expected cases depending on the intended purpose of servers and traffic types of logs in scoring data set. Therefore, we have multiple baseline cases accordingly as it is stated before.

After defining baseline profiles, anomaly cases are defined and next, anomaly metrics in both scoring data and network pcaps are identified. Data sets are correlated by identified metrics and once the algorithm that will search by the metrics is decided, data sets are pre-processed in order to prepare them for analysis phase. Once pre-processing is done, both data sets are tied by timestamp and the correlation value which represents the traffic intensity at that timestamp for that specific case. Anomalous time points are identified according to the final data set which consists of timestamp and correlation values.

During implementation, I worked on different servers which are mail, web and dns. However, methodology has been demonstrated over a *mail server and its ping traffic* for the anonymous *blue team XX* since I have gotten the best visual results out of it.

3.1.1. Mail Server and Ping (ICMP) Traffic

Ping is a fundemental software command in every operating system to check whether a server is reachable or not and it is used by scoring bots in Locked Shields game to fit its purpose as well. Thus, in scoring data (Figure 1), there are records showing the aggregated information of received packets per ping session.

```
"mail.blue  .ex"|"ping"|"OK"|"PING OK - Packet loss = 75%, RTA = 26.09 ms"|"1461155051"|"1461155062"
"mail.blue  .ex"|"ping"|"CRITICAL"|"PING CRITICAL - Packet loss = 100%"|"1461155087"|"1461155102"
"mail.blue  .ex"|"ping"|"CRITICAL"|"PING CRITICAL - Packet loss = 100%"|"1461155123"|"1461155138"
"mail.blue  .ex"|"ping"|"CRITICAL"|"PING CRITICAL - Packet loss = 100%"|"1461155159"|"1461155174"
"mail.blue  .ex"|"ping"|"OK"|"PING OK - Packet loss = 86%, RTA = 16.11 ms"|"1461155195"|"1461155210"
"mail.blue  .ex"|"ping"|"OK"|"PING OK - Packet loss = 86%, RTA = 15.11 ms"|"1461155231"|"1461155246"
"mail.blue  .ex"|"ping"|"OK"|"PING OK - Packet loss = 66%, RTA = 10.23 ms"|"1461155267"|"1461155275"
```

Figure 1. Sample of ping records in scoring data

There are two main indicatives of a mail server being available and functioning in a network in scoring data. First is total packet loss rate in ping sessions and second is whether there is a mail traffic being occurred or not. Packet loss rate is important because it evinces the accessibility of machine. Existence of mail traffic is also

important because it shows that mail server is actually functioning, serving as intended. Mail traffic type consists of smtp, smtps, imap and pop3 in Locked Shields case. These two indicatives jointly shall indicate the actual availability of a mail server in scoring data of Locked Shields rather than individually.

On this case study, ping traffic of mail.blueXX.ex machine belonging to an anonymous blue team is examined. There is a possibility of encountering 3 different cases of ICMP packet loss rate in scoring data regarding the aforementioned mail server:

- Packet loss rate is 0 (PLR = 0)
- Packet loss rate is in between 0 and 100 (0 < PLR < 100)
- Packet loss rate is 100 (PLR = 100)

Expected cases are detailed and anomaly cases are identified in following section.

3.1.2. Defining Expected and Anomaly Cases

There are 3 expected packet loss rate related cases that are described in 3.1.1. According to the cases, we can put forth possible anomaly cases. Expected and anomaly cases are enumerated and explained in below.

Expected Case 1: Packet loss rate is 0 (PLR = 0) → Server is accessible and we expect to see network traffic from/to mail.blueXX.ex at corresponding times in scoring data where there is no packet loss.

Anomaly Case 1: Assuming server is accessible since there is no packet loss, we do not expect not seeing network traffic data from/to mail.blueXX.ex. Because if there is a sent packet by scoring server, there should be at least one received packet in mail.blueXX.ex since scoring server was replied and claimed there is 0% packet loss. If we detect such a case in data frames, we claim an anomaly and start further investigation.

Expected Case 2: Packet loss rate is in between 0 and 100 (0 < PLR < 100) → Server is partly accessible and thus, we expect to see traffic from/to mail.blueXX.ex.

Anomaly Case 2: Anomaly case is the same as case 1 here since the server is not entirely out of access. Because it replied to some ICMP requests and scoring data claimed there is X% amount of packet loss where X is not equal to 0 and 100.

Expected Case 3: Packet loss rate is 100 (PLR = 100) → Server is not accessible and we do not expect to see any traffic from/to mail.blueXX.ex as scoring server claims.

Anomaly Case 3: If there is 100% packet loss in a network traffic, we do not expect to see any traffic from/tomail.blueXX.ex. If there is any traffic in pcap files at the time where there is no reply from mail server, we claim an anomaly and start looking into the

case. According to the traffic type found in pcap files, case might break down in such cases:

- If found traffic type is either smtp, imap, smtps or pop3; this subcase shows that although there is no sent and received ICMP packets from/to mail server, it is functioning properly. This induces us two possible reasons within given information of Locked Shields:
  - **Misconfigured reverse path filtering:** Mail server is running an Ubuntu operating system and thus, it has built-in advanced routing and traffic control features including reverse path filtering. It is a kernel feature that ignores the packets which are not replied through the interface they came in. [44] Blue team might have enabled reverse path filtering for wrong network interface and treated scoring server requests as bogus packets.
  - **Misconfigured firewall:** Blue team might have set up a firewall and configured a poor blocking rule or configured a correct rule for either wrong source ip or source interface.
  - **Miscellaneous:** There might be a problem with the router in between or scoring server is just dropping replies for some reason. Further investigation is needed to detail it but whatever the reason is this subcase points an anomaly since there is an inconsistency between pcaps and scoring.
- If found traffic type contains both sent and received ICMP packets; case pinpoints that although reply packets are generated and sent successfully, there may be either a problem with the router in between or scoring server is dropping replies anyhow.
- If found traffic type is only received ICMP requests and there is no echo reply back; this case also indicates an anomaly may be due to either misconfigured path filtering, firewall or routing tables on mail.

3.1.3. Identifying Anomaly Metrics

Metrics should be defined as simple and clear. They should not contain ambiguous values and they should clearly state the anomaly to eliminate false-positives as much as possible. On mail server case study, anomaly metric in scoring data is determined to be the packet loss rate. Because it clearly denotes the availability of the server numerically. Packet loss rate in scoring data, is supposed to have a binder metric in network pcaps too. Considering we already can say if there should be a traffic going on between the

mail server and scoring server by looking at the packet loss rate, we expect to see corresponding network packets or to not seeing network packets in pcaps as well. Therefore, anomaly metric in network pcaps is identified as total number of packets in pcaps that were sent and received at that specific timestamp in scoring data. However, considering sent and receive operations also consume time and this might cause some of packets deviating from the exact timestamp in scoring data; time on only hour and minute basis should also be taken into account separately. All in all, we end up having 2 different time formats to look for in pcap files: one is the full time format while other is on hour and minute basis time format in order to achieve nearest results.

3.1.4. Data Correlation

In the scope of discussion in deduction section, instead of creating a baseline profile and to validate it; anomaly cases are created out of expected ones for mail server case study and are looked for in pcap data. Therefore, scoring data is validated by pcap files and an initial consistency between scoring data and pcap files is provided.

Having anomaly scenarios predefined has an advantage of allowing one to choose which searching algorithm should be used for which anomaly as well. However, looking through defined anomaly metrics in previous section, we obviously have limit on them and this narrows the choices.

As it can be seen when scoring data is examined, ping logs of scoring data has the granularity of 2 log records per minute nearly. Therefore it can be noted that there is a nearly regular period per time unit -time unit is minute in this case- and this period is 2. Also for each period, we have a corresponding metric which is packet loss rate. However although we have anomaly metrics in both data sets, we still do not have correlation. Data sets should be connected on the binder metrics.

Considering anomaly scenarios in section 3.1.2, packet loss rate should be inversely proportional to the number of packets hence multiplication of them is an inverse proportion constant that indicates their relation. So, the multiplication is eligible tobe used as a variable that corralates scoring and pcap data.

Following section describes an univariant anomaly detection algorithm to be used for implementation.

3.1.5. Anomaly Detection

AnomalyDetection is an open source R package, developed by Twitter in order to detect anomalies from a statistical point. It is a defacto package that can be used for detecting anomalies in data sets related to computer science to politics. Therefore, it is a handy

package that would also fit underlying anomalies of Locked Shields. The base algorithm used in package is named as Seasonal Hybrid ESD (S-H-ESD) by Twitter. S-H-ESD is is based on outlier identification by using generalized ESD test. [50]

Generalized ESD test is a method of identifying one or more outliers in a univariate data set even though there is a masking. Masking means discordant data points cancelling the effect of more extreme ones. In other words, it is not being able to see the big picture due to disorder of data points. [51], [52]

The test only needs an upper bound for the suspected number of outliers to be detected. According to the generalized ESD algorithm, a normal plot of given data set is created in first step. After that, a significance level which indicates threshold, and an upper bound of suspected amount of anomalies are manually set. Next, statistical test and critical values are calculated according to the formula that is described in [52]. In the end, test statistics that are greater than their critical values are pointed as anomalies. [51], [52]

What the package brings up as new is that it is also able to detect local and global anomalies by using time series decomposition with generalized ESD algorithm. Local anomalies denote the anomalies inside seasonal patterns while global anomalies denote anomalies that cannot be explained with seasonal patterns. A seasonal pattern means a data set influences by a fixed and know time period that is called as "season". A season can be a month, a day, a week of a month etc. [50]

Twitter's AnomalyDetection package has 2 main functions for detection.

AnomalyDetectionTs() function is used to detect one or more statistical anomalies in the data set with input time series. However since LS data granularity is nearest to 2 log records per minute -meaning it is on seconds basis- and AnomalyDetectionTs() demands to have either minutely, hourly or daily data, this function can not be employed. [50]

```
AnomalyDetectionTs(dataset, max_anoms, direction, only_last, plot)
```

Figure 5. AnomalyDetectionTs() function for data with time series

On the other hand AnomalyDetectionVec() function is used to detect statistical anomalies in a vector of data set. Data is not required to have regular time series. [50] Therefore AnomalyDetectionVec() function is more applicable for LS. Once, anomalies are detected with AnomalyDetectionVec(), timestamps can be still pointed by their indexes although time was not given as a parameter to the function.

```
AnomalyDetectionVec(dataset[,2], max_anoms, period, direction,
only_last, plot)
```

Figure 6. AnomalyDetectionVec() function for data without time series

The function takes 6 arguments.

dataset[,2]; is the column of a data frame consists of number of observations.

max_anoms; is the upper bound of suspected amount of anomalies in data set, as a percentage of data set. This means assuming data set has X number of data points then there would be max_anoms% of them that are expected to be anomalous.

period; is the number of observations in a single season or period

only_last; indicates reporting anomalies found only in last season if it is set TRUE.

direction; means the direction of anomalies to be detected. If it is set 'both', algorithm detects both negative and positive anomalies.

plot; means when an anomaly is detected, it will be displayed as a plot on graph. [50]

After metrics and anomaly detection algorithm are determined, data sets are pre-processed and composed to construct a final dataset to be used in AnomalyDetectionVs() in following steps.

3.1.6. Scoring Data Pre-Processing

Data on a particular topic might be gathered in many ways so that results in having an intricate data set to analyze. The source of these data varies from observations, interviews to sensors, logs etc with different levels of complexity and reliability. Every operation that is performed over raw data to use it efficiently in later processes is considered as data-preprocessing and it is an essential step of data mining in order to get a better understanding of data and being able to deduct more reliable results. [20]

Scoring data in Locked Shields comes in a text format including information of received packets belonging to different traffic types. Each record has a server host name, traffic type, a status indicator (has 3 options: OK, CRITICAL and WARNING), description, sent and received times respectively and each field is separated by pipe operand ("|") (Figure 1).

There are 2 obstacles on data in this phase. First is to pull out ping traffic logs belonging to mail.blueXX.ex and second is to convert time values which are in Unix format to a human readable format (HH:MM:SS). I created a python script for the extracting ping traffic logs. Script can be executed by passing team name, an identifier of host name and traffic type through command line. (Appendix 1) Essentially, it can be executed by following command line statement in order to extract ping logs of mail.blueXX.ex:

```
python script.py blueXX mail ping
```

Figure 2. Executive command to extract ping logs of mail.blueXX.ex

I also created a mini R script in order to convert Unix formatted timestamps to a readable date format. (Appendix 2) The reason I use R for time conversion is that I want to be able to convert times with minimum function arguments and to juggle with values in an easier and flexible way. This flexibility is needed for handling unexpected cases. For instance, since we have switched to daylight saving time on 26th March, pcap files were tcpdumped in accordance with new time as of that date. This situation caused to have inconsistency between timestamps in scoring data, in the names of pcap files and in pcap content. Therefore timestamps in scoring data require additional manipulations after being converted in order to proceed to the next step of analysis. Since R has quite developer-friendly and effective solutions to overcome such problems, I preferred to move on with it. The R script is called as a subprocess in extracting python script.

3.1.7. Identifying Pcap Files To Be Tcpdumped

Locked Shields lasts two days in total. There are 2075 pcap files of blue team XX for only the first day of game. Considering sizes of pcap files varying between 24B to 6.2GB and only traffic sent to/received from mail.blueXX.ex server at specific times is needed; it is very cumbersome and redundant workload to go through all pcap files and tcpdump them. Therefore I decided to map scoring data and pcap files by time values in order to identify which pcap files are needed to be tcpdumped to extract traffic belonging to the mail server.

I created a python script to perform data set mapping at first and to tcpdump mapped pcap files secondly. (Appendix 3) Script can be executed by passing file path in where you have pulled out ping traffic logs belonging to the mail server and a desired suffix string in order to be used for naming the folder in where dumped files will be stored. As an example, to find correct pcap files to be dumped and tcpdump them afterwards; script shall be executed as:

```
python script.py /pathtoextractedpingtrafficlogs/logs.mail.ping
mail.ping.txt
```

Figure 3. Executive command to map and tcpdump correct pcaps

Algorithm begins with parsing times in scoring data as HH:MM:SS format and add parsed values into an array. Next, the array is iterated by indexes to convert HH:MM:SS format to 20160420HHMMSS format in order to obtain the identifier string which will

be used for mapping. Pcap filenames are formatted as *teamname-ip4-20160420HHMMSS.pcap* and that is why this conversion is needed. In addition to this, 3 hours are subtracted from time value after conversion. Because each pcap file have been named according to UTC time zone while scoring data logs are in EEST time zone. To give a concrete example, converted scoring data identifier 20160420155555 would become 20160420125555 to identify correct pcap file which is blueXX-ip4-20160420125555.pcap. Mapping starts after all correct mapping identifiers are constructed in same way.

Mapping algorithm is based on the fact that there shall be at least one pcap file which corresponds to a scoring data identifier. In other words, we look for an entire match between 20160420HHMMSS values. If there is not any, then there shall be at least one pcap file which includes 20160420HHMM and this pcap file shall be second to the last one among ascendingly listed files. Assuming, if there is not an entire match for 20160420131512; then algorithm lists pcap files including 201604201315 in ascending order. In this context, if we had blueXX-ip4-20160420131501.pcap, blueXX-ip4-20160420131508.pcap and blueXX-ip4-20160420131513.pcap listed, 20160420131508 would have been picked as correct file since it contains logs starting from a time behind the time we are looking for and hence including the logs we want.

If there is not any listed pcap files again, this time algorithm looks for a match of 20160420HH. If there is still no pcap files listed, it returns a message saying that there is no pcap file found for identifier 20160420HHMMSS.

Picked pcap files are added in an array, they are tcpdumped one by one and saved in another directory by iterating over the array in the end of script. Picked pcap files are also stored in a dictionary as values with having corresponding converted scoring data identifiers as keys. This dictionary is needed for next step.

As mentioned above, there were 2075 pcap files for blue team XX. However, this amount is now reduced to 754 files by using this algorithm for mail server case study.

3.1.8. Counting Network Packets

As it is mentioned in previous sections, it is expected to have traffic logs in pcap files that correspond to scoring data logs and thus anomaly metric in network pcaps is number of total packets per timestamp. Because in the fundemental context of networking; if there is a sending packet to an endpoint, there must be at least one received packet by the endpoint to claim that both sender and receiver endpoints are connected.

Number of total network packets per timestamp is the anomaly binder of network pcaps. We aim to create a data frame consists of number of packets and corresponding timestamps in order to correlate it with scoring data frame later.

Therefore, I created a python script in order to pinpoint ingoing/outgoing traffic to the mail server at HH:MM:SS time of scoring data logs. (Appendix 4) I look for a full time value match at first and if there is not an exact match, I look for a HH:MM match. Therefore I am looking at scoring data in an observation time window of 1 minute eventually.

Script is executed on command line by passing file path where you extracted ping logs from scoring data, file path where you dumped relevant pcap files, the ip of the server you look for and a desired file name to use naming the file that will store number of counted network packets in the end. For mail server case study, script can be executed as following:

```
python script.py /pathtoextractedpingtrafficlogs/logs.mail.ping
/pathtodumpedpcaps/pcaps.mail.ping SERVER_IP mail.ping_counted.packets
```

Figure 4. Executive command to count network traffic logs in pcaps

Algorithm begins with parsing datetimes in scoring data as HH:MM:SS format as it does in previous step. In this way, we obtain outgoing traffic timestamps which were sent by scoring server and we will be looking for in pcap logs.

Algorithm also uses the same mapping algorithm that is used in identifying pcap files. The only difference is, it takes the returning dictionary with {datetime:pcap file names} key-value pairs. I used a dictionary data structure because since I extract scoring time values to use as keys; it is easy to look through a dictionary by them and to find out which corresponding pcap files I should search the traffic in. Once algorithm finishes counting relevant traffic logs, it opens a new file and save in them as [time]|[number of counted logs at the time] format to ease file parsing in R later.

3.1.9. Pre-processing Final Data In R

After getting data ready to process in R, relevant files are imported in R environment. We need ping logs extracted from scoring data and number of packets per time in pcap file. First, previously extracted scoring data regarding ping traffic of mail server is imported into a data frame and parsed by read.table function. If we name data frame *mailping*; an instance of it would look like following table.

Table 1. Data frame instance of extracted ping data

V1	V2	V3	V4	V5	V6
mail.blueXX.ex	ping	OK	PING OK - Packet loss = 57%, RTA = 15.76 ms	2016-04-20 10:53:17	1461135203

Since we need packet loss rate and sending time only out of scoring data, we create a subset data frame *mailpingpacketloss* which is only composed of V4 and V5. Only those two is needed because packet loss rate is the anomaly binder of scoring data and sending timestamp is the mapping variable with network pcaps. Next, we use str_extract() function of R to extract only the numeric percentage value from V4. In this way, *mailpingpacketloss* shaped in its final form (Table 2)and scoring data frame has been created.

Table 2. Data frame instance of extracted ping data in final form

V4	V5
57	2016-04-20 10:53:17

Second, previously created file including [time]|[number of counted logs at the time] information is also imported and parsed as scoring data. It is parsed in order to get rid of noisy non digit characters and is sorted by its time column and network pcaps data frame has been created. (Table 3)

Table 3. Data frame instance of counted network packets in final form

V1	V2
10:25:24	122
10:25:56	0

As it is stated in section 3.1.4., both data frames should be correlated over binders to obtain an univariant data set which indicates the connection both scoring and network pcaps. Multiplying number of packets in Table 3 and packet loss rate in Table 2 and then merging them by timestamps, gives us the single-variant data set we want. Therefore mailpacketloss$V4 (Table 2) and mail_totalpackets$V2 (Table 3) are

multiplied and resulted column is added in a new data frame named *mail_analysisdata* along with corresponding time values.

The first created *mail_analysisdata* data frame is used to detect anomalies in anomaly case 3. Since each case has different anomaly condition, *mail_analysisdata* set is needed to be modified for each case in following section.

3.1.10. Applying AnomalyDetection to Locked Shields

After constructing *mail_analysisdata* in section 3.1.9, its second column (mail_analysisdata[,2]) where total number of packets per timestamp are stored is passed to AnomalyDetectionVec() function as following for mail server case study:

```
AnomalyDetectionVec(mail_analysisdata[,2], max_anoms=0.05, period=2,
direction='pos', only_last=FALSE, plot=TRUE)
```

Figure 7. Calling AnomalyDetectionVec() for Locked Shields data

Values that are passed to function are explained:

max_anoms; this indicates the upper bound value as percentage. It is a challenge is to set a proper upper bound of suspected anomalies in final data set. Because by definition of Generalized ESD algorithm, there is not any specific way to calculate an upper bound value to start with. So it is a matter of determining how much percentage of your data is expected to be anomalous and it requires an iterative sensivity analysis to determine the nearest percentage. There are 887 rows in mail_analysis data frame. I started with 0.05 meaning 5% of data is expected to be anomalous as this is defined by default in AnomalyDetection package. There were 30 anomalous time points detected as cyan dots on the graph. (Figure 8)

period; is the granularity of final data set which is 2 since we have nearly 2 log records per one minute in scoring data.

only_last; this does not have any affect on our case, so is set as default.

direction; is set as positive since we dont have a negative correlation value for this anomaly case 3. This will set to negative in order to detect case 2 which is explained in further sections.

plot; is set to true to display anomalous points as dots on the graph.

```
> AnomalyDetectionVec(mail_analysisdata[,2], max_anoms=0.05, period=2, direction='pos', only_last=FALSE, plot=TRUE)
$anoms
   index  anoms
1     18   8300
2     19 107500
3     29    400
4     51    600
5     52   4100
6     53  13300
7    196    375
8    207    575
9    239   1025
10   249   1000
11   256    875
12   284   3320
13   289   1150
14   309    775
15   312    250
16   378    850
17   385    150
18   386   2975
19   389  37650
20   404   2425
21   408    225
22   409    400
23   434   2425
24   478   1200
25   525   2200
26   556   2000
27   562    575
28   605   1325
29   632   2875
30   683   2475
```

Figure 8. Calculated anomalies for ping traffic of mail.blueXX.ex

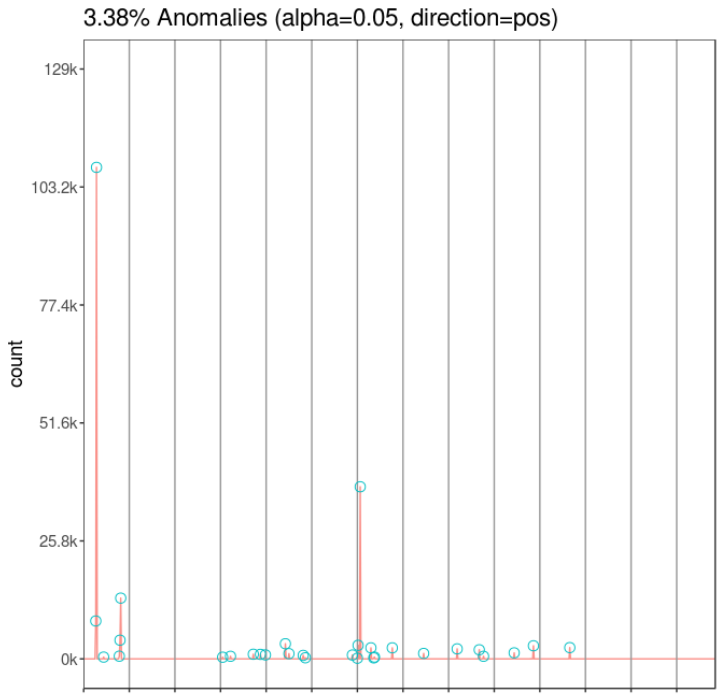

Figure 9. Plot representation of the calculated anomalies for case 3

Figure 9 represents the anomalous visual of *mail_analysisdata* data frame. On y-axis, there are multiplication results also known as correlation values which can be seen as *anom* column in Figure 8 while on x-axis there are timestamps where we can find in *mail_analysisdata* by indexing based on *index* column in Figure 8. Each figure in further pages denotes same variables on both x-axis and y-axis. Cyan points stand for detected time points where there are anomalous correlation values. In other words, they

point out the times where scoring data and network pcaps do not conform each other. The root cause of data sets not conforming each other should be investigated for each time point. As I manually went through some anomalies, there were cases such as pcap files were not tcpdumped due to them being corrupted. Further investigation is needed to break down these cases.

A detected anomaly is considered as a false-positive for this anomaly case 3, if packet loss rate in scoring data is less than 100. Therefore, I looked into detected time points in *mailpingpacketloss* (Table 2) by timestamps to check whether there were false-positives or not. As result, 24 of them were found false-positive and to decrease those, max_anom was set to 0.03 for second iteration.

26 anomalies were found for upper bound value 0.03 and 18 of them were false-positive this time. I did not continue to decrease upper bound value since decreasing it also started bringing false-negatives at the same time.

In the end, there were 6 true-positive data points that were pointed as anomalies indicating times where there were ongoing traffic although the mail server was being scored as down. This shows that anomaly case 3 defined in section 3.1.6 is validated by the proposed methodology. (Table 4).

Table 4. True-positive anomalous data points

index	anoms
18	8300
19	107500
29	400
51	600
52	4100
53	13300

In anomaly case 1 defined in section 3.1.2; we expect to see at least some traffic since packet loss rate is 0. Considering anomaly that we want to detect consists of cases where we have both packet loss rate and number of counted packets are 0, *mail_analysisdata* data frame is needed to be modified in order to calculate the case from a statistical perspective as stated before. Because algorithm can not calculate statistical representation of such cases due to 0 being the absorbing and unsigned integer.

As first step, column values where number of counted packets are stored (mail_totalpackets$V2) are set to -100 where mail_totalpackets$V2 equals to 0. This will negatively point out the records we want to identify. Next, mail_totalpackets$V2 values are set to 0 where mail_totalpackets$V2 does not equal to -100. This will reduce the number of excessive correlation values coming from other cases in final data set and thus anomalies will be pinpointed on the graph in a clearer way. After that, column values where we keep packet loss rates (mailpacketloss$V4) are set to 200 where mailpacketloss$V4 equals to 0. That's because we want to mask cases where packet loss rate is 100 and counted packets are 0 (-100 in updated data frame). Finally, we perform multiplication and generate *mail_analysisdata* data frame for case 1.

As in anomaly case 3, recreated *mail_analysisdata* is passed to AnomalyDetectionVec() with same arguments except direction (Figure 7). Direction argument is set as negative instead of positive this time since multiplication values to be calculated are negative. There were 44 anomalies found with no false-positives. (Figure 10).

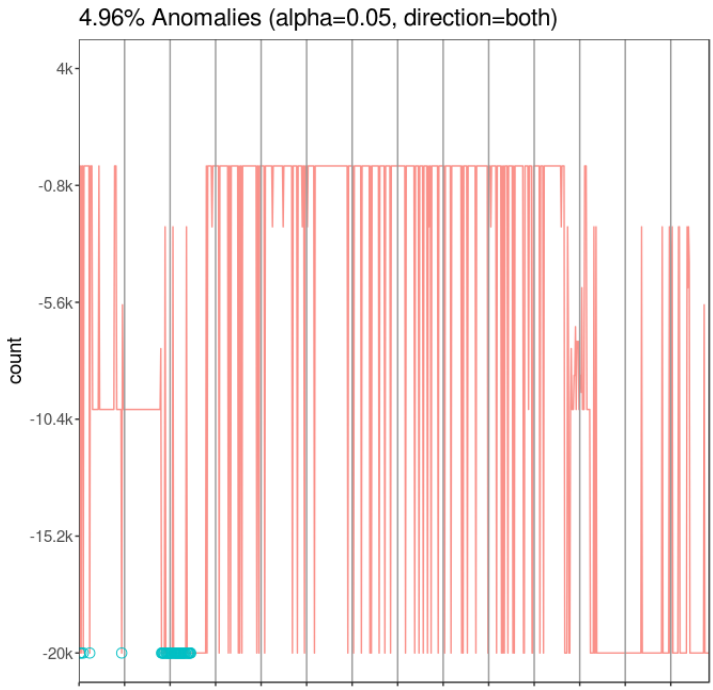

Figure 10. Plot representation of the calculated anomalies for case 1

When looking at the plotted anomalies on the graph, it is obvious that there are other data points that peak the same count value by the cyan points along the right edge of the graph. In other words, upper bound of expected anomalies should be expanded to cover all anomalous points on the graph. Thus, max_anom value is doubled and set to 0.1. In

this case, there were 88 anomalous timestamps detected (Figure 11) with no false-negatives and again, as it is seen on graph max_bound should be expanded since there are still uncovered anomalies.

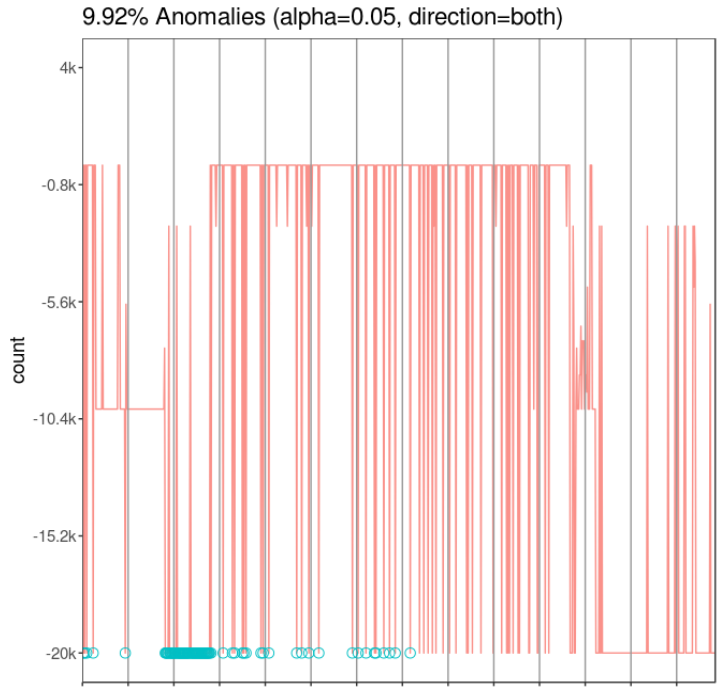

Figure 11. Plot representation of the calculated anomalies for case 1 (ma = 0.1)

There were 177 anomalies detected again with no false-positive where max_anom value was redoubled and set to 0.2 (Figure 12) .

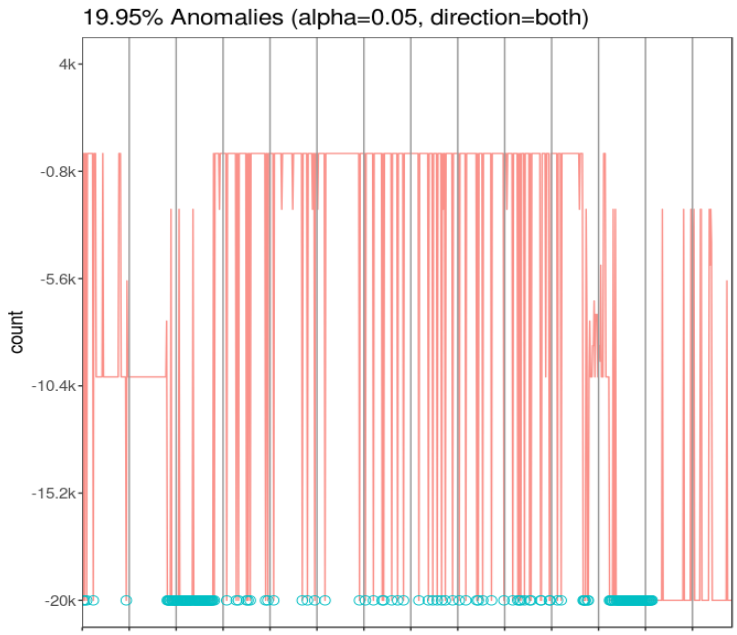

Figure 12. Plot representation of the calculated anomalies for case 1 (ma = 0.2)

In order to detect uncovered anomalies on graph, max_anom value is set to 0.4 for this time. However this time there were 81 false-positive data points were also detected (Figure 13). False-positives are actually expected cases where packet loss rate is 100 and number of packets are 0 for case 1. I manually looked up them in *mailpingpacketloss* (Table 2) and *mail_totalpackets* (Table 3) by timestamps as I did in case 3.

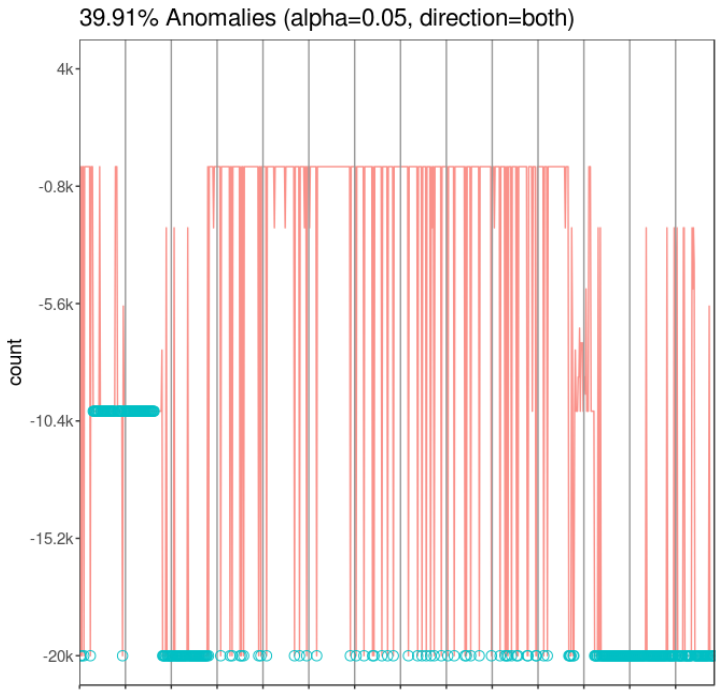

Figure 13. Plot representation of the calculated anomalies for case 1 (ma = 0.4)

To exclude false-positives, max_anom is reduced to 0.3 for next iteration and finally, there were 266 anomalies detected with no false-positives (Figure 14) and there is no other anomalous data points that were uncovered.

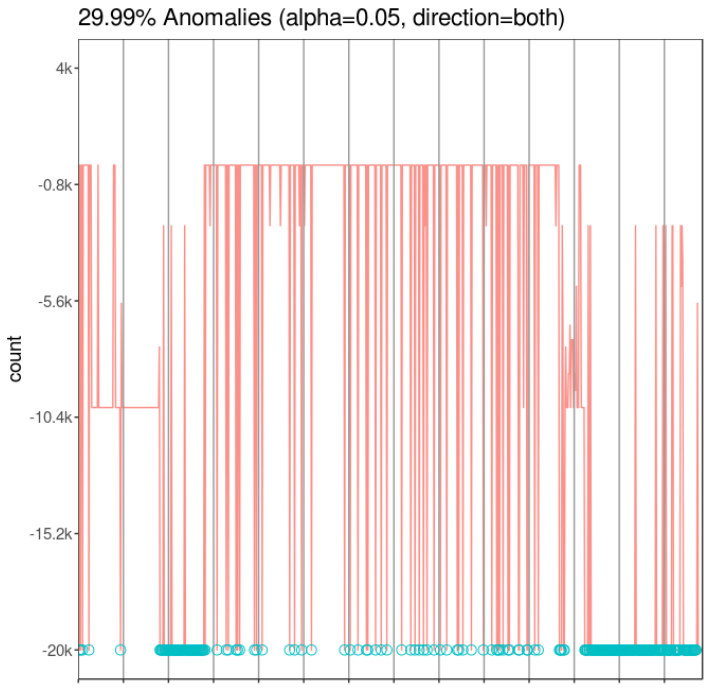

Figure 14. Plot representation of the calculated anomalies for case 1 (ma = 0.3)

Anomaly cases 1 and 3 have been validated by the proposed method so far. Similar ways used to validate both can be used to validate case 2 as well. However, since it is stated in case 2 that packet loss rate is between 0 and 100, meaning server is partly available so that we expect to see at least some traffic; every data row which has chance to show up as anomaly belonging to either case 1 or case 3, should be clearly dismissed. In order to do this, column values where we keep packet loss rates (mailpacketloss$V4) are set to 0 where mailpacketloss$V4 equals to 100. This will keep anomalies of case 3 away from popping up. After that, column values where number of counted packets are stored (mail_totalpackets$V2) are set to 1000 where mail_totalpackets$V2 equals to 0. This will skyrocket multiplication results hence it will pinpoint anomalies without false-positives. Because for example a case where we have 40% packet loss rate and 200 packet is an expected case so that we set values high to not have expected cases as anomalies.

There is no need to do any changes to data frame for dismissing case 1. Because, in case 1, packet loss rate is 0 and thus, they will not even be calculated in algorithm due to having 0 in multiplication.

In the end, multiplication is done and final data set for case 2 is created. As in anomaly case 3, recreated *mail_analysisdata* is passed to AnomalyDetectionVec() with same arguments and with max_anom value being 0.05 for first iteration. There are 44 anomalies were detected and 3 of them were false-positive as it is seen on the graph (they are the lowest 3) and also by manually checking as in case 1 and 3. (Figure 15)

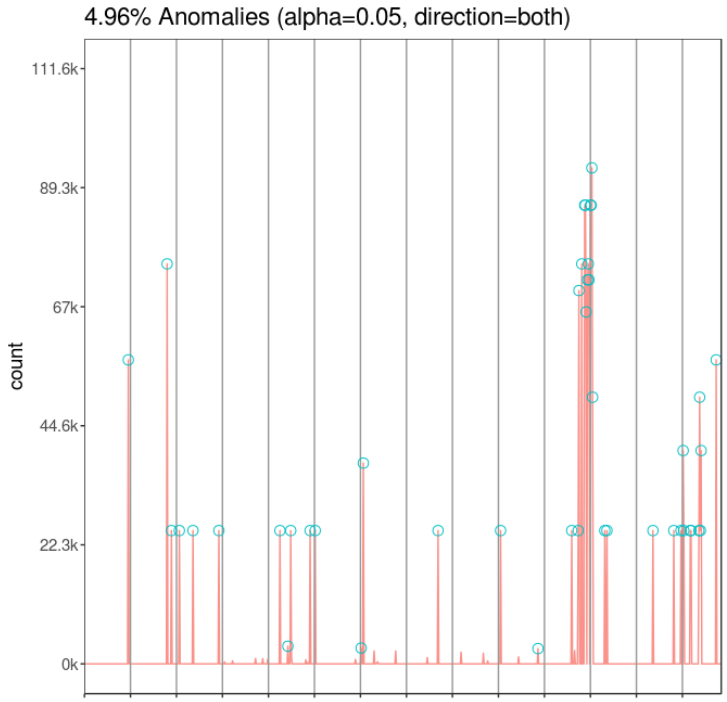

Figure 15. Plot representation of the calculated anomalies for case 2 (ma = 0.05)

In order to eliminate 3 false-positives, max_anom value should set to an x where 0.04 < x < 0.05. Thus, max_anom is reduced by 0.005 percent. However since this causes a loss of 2 anomalous points; after several iterations where 0.045 < max_anom < 0.05; 0.047 is identified as the optimum value for max_anom where we have all anomaly points and no false-positives detected. (Figure 16)

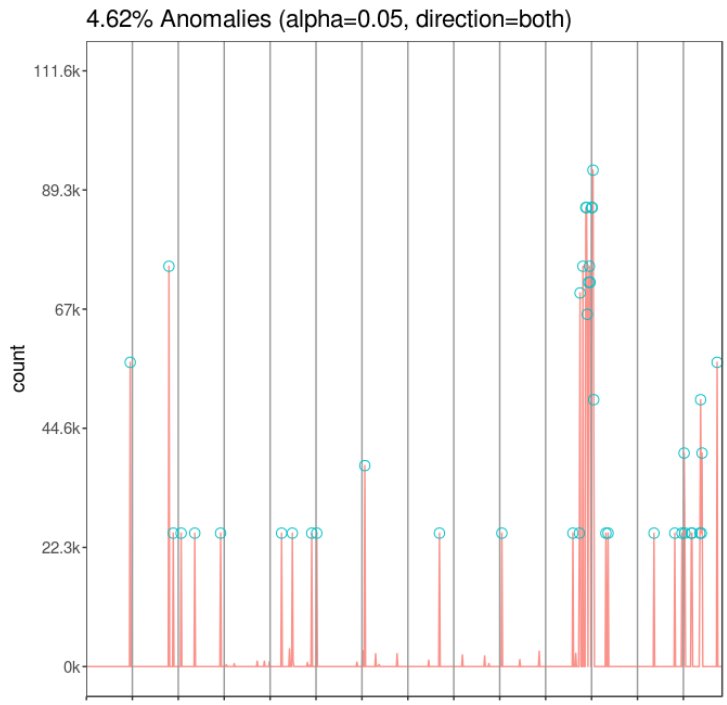

Figure 16. Plot representation of the calculated anomalies for case 2 (ma = 0.047)

All anomaly cases that are defined in section 3.1.2 have been validated by applying proposed steps and have been visualized accordingly. Although cases are needed to be broken down to obtain details, anomalous points that are shown as cyan dots on graphs substantially indicate unconformations between scoring data and network pcaps and thus, warns authorities about the fact that scoring data needs to be improved.

4. Conclusions

Several anomaly detection methods are discussed in this paper. Locked Shields data sets are examined in order to find out whether a discussed anomaly detection method is applicable or not. To support the claim that Locked Shields scoring data can be validated by anomaly detection methodologies, a new method inspired by statistical modelling over scoring data and network pcap files was proposed and implemented in scoring data.

Experimental results on case study data indicate that anomaly detection based techniques are able to provide a fast data correlation, accurate results and a more user-friendly way of data representation. Predefined anomaly cases are covered and anomalous timestamps are pinpointed on R graphs. Detected anomalies can be clearly degraded into the points where scoring data process needs to be strengthened. However, it should be noted that all of detected anomalies are required to be investigated to the root cause. Because as it is stated before in related work section, anomaly detection does not provide root causes of anomalous events, and thus there is always a possibility of either crystal clear or underlying false-positives among anomalies. Therefore, a manual interpretation or review is necessary in the end.

Anomaly cases were constructed based on networking knowledge and scoring data analysis in a limited amount of time and knowledge. My long-term goal would be to construct more complex and branching cases that include more than one traffic type. If there are more complex cases with multiple traffics in future, multiple baseline profile models can be created and AnomalyDetection R package can be modified particularly for Locked Shields baseline models. As result, more reliable, complex and significant anomalous points can be detected.

References

[1] S. E. Goodman, J. C. Kirk, and M. H. Kirk, "Cyberspace as a medium for terrorists,"*Technological Forecasting and Social Change*, vol. 74, no. 2, pp. 193–210, Feb. 2007.

[2] ENISA, "CSIRTs by country - interactive map," 2016. [Online]. Available: https://www.enisa.europa.eu/topics/national-csirt-network/csirt-inventory/certs-by-country-interactive-map. Accessed: Feb. 8, 2017.

[3] NATO CCDCOE, "Slovakia wins locked shields 2016 ahead of NATO and Finland," CCDCOE, 2016. [Online]. Available: https://ccdcoe.org/slovakia-wins-locked-shields-2016-ahead-nato-and-finland.html. Accessed: Feb. 9, 2017.

[4] Kaspersky, "Kaspersky Security Bulletin 2015," in *https://securelist.com/*, 2015. [Online].Available:https://securelist.com/files/2015/12/KSB_2015_Statistics_FINAL_EN.pdf. Accessed: Feb. 16, 2017.

[5] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends,"*Computer Networks*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007.

[6] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, "A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection,"*SIAM 2003*, pp. 25–36, 2003.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection,"*ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[8] D. E. Denning, "An intrusion-detection model,"*IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[9] S. Kumar, E. H. Spafford, and Purdue, "'An application of pattern matching in intrusion detection' by Sandeep Kumar and Eugene H. Spafford," 1994. [Online]. Available: http://docs.lib.purdue.edu/cstech/1116/. Accessed: Mar. 4, 2017.

[10] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges,"*Computers & Security*, vol. 28, no. 1-2, pp. 18–28, Feb. 2009.

[11] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," in *Managing Cyber Threats*. Springer Nature, pp. 19–78.

[12] D.E.Denning, P. G. Neumann, "Requirements and model for IDES-A real-time intrusion detection system,"Comput.Sci. Lab, SRI International, Menlo Park, CA, Tech. Rep., 1985.

[13] S. E. Smaha, "Haystack: An intrusion detection system,"*[Proceedings 1988] Fourth Aerospace Computer Security Applications*, pp. 37–44, 1988.

[14] E. A. Fisch and G. B. White, *Secure computers and networks: analysis, design, and implementation*. Boca Raton, FL: CRC Press, pp. 100-101, 1999.

[15] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, “A sense of self for Unix processes,” *Proceedings 1996 IEEE Symposium on Security and Privacy*, pp. 120–128, May 1996.

[16] T. Lane and C. E. Brodley. "An application of machine learning to anomaly detection."*Proceedings of the 20th National Information Systems Security Conference*, vol. 377, pp. 1–12, 1997.

[17] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A Survey,” *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jan. 2009.

[18] D. Heckerman, D. Geiger, and D. M. Chickering, “Learning Bayesian networks: The combination of knowledge and statistical data,” *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.

[19] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, “Bayesian event classification for intrusion detection,” *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, Dec. 2003.

[20] A. Famili, W. Shen, R. Weber and E. Simoudis, "Data preprocessing and intelligent data analysis", *Intelligent Data Analysis*, vol. 1, no. 1-4, pp. 3-23, 1997.

[21] I.K. Fodor, "A Survey of Dimension Reduction Techniques", *Lawrence Livermore National Library*, pp. 1-24, 2002.

[22] Y. Lee, Y. Yeh and Y. Wang, "Anomaly Detection via Online Oversampling Principal Component Analysis", *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1460-1470, 2013.

[23] "PCA-Based Anomaly Detection", *Msdn.microsoft.com*, 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn913102.aspx. [Accessed: 13- Apr- 2017].

[24] N. Ye, "A markov chain model of temporal behavior for anomaly detection"*Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop*, vol. 166, pp. 171-174, June. 2000.

[25] L. Rabiner and B. Juang, "An introduction to hidden Markov models", *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4-16, 1986.

[26] A. Srivastava, A. Kundu, S. Sural and A. Majumdar, "Credit Card Fraud Detection Using Hidden Markov Model", *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 1, pp. 37-48, 2008.

[27] E. Bloedorn, A. D. Christiansen, W. Hill, C. Skorupka, L. M. Talbot, J. Tivel, "Data Mining for Network Intrusion Detection: How to Get Started", *MITRE Technical Report*, 2001.

[28] W. Lee, S. J. Stolfo, "Data Mining Approaches for Intrusion Detection." *Usenix security*, pp. 79–94, 1998.

[29] W. Zhang, Q. Yang and Y. Geng, "A Survey of Anomaly Detection Methods in Networks", *2009 International Symposium on Computer Network and Multimedia Technology*, 2009.

[30] G. Klir, B. Yuan, "Fuzzy sets and fuzzy logic", New Jersey: Prentice hall, vol. 4, 1995.

[31] S.M. Bridges, R.B. Vaughn, "Fuzzy data mining and genetic algorithms applied to intrusion detection." *Proceedings of the National Information Systems Security Conference*, Baltimore, MD, 2000.

[32] J. Dickerson and J. Dickerson, "Fuzzy network profiling for intrusion detection", *PeachFuzz 2000. 19th International Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.00TH8500)*, pp. 301-306, 2000.

[33] W. Li, "Using genetic algorithm for network intrusion detection."*Proceedings of the United States Department of Energy Cyber Security Group* 1, pp. 1-8, 2004.

[34] R. H. Gong, M. Zulkernine and P. Abolmaesumi, "A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection", *Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05)*. 2009.

[35] M. Crosbie and E. Spafford, "Applying Genetic Programming to Intrusion Detection", *Proceedings of the AAAI Fall Symposium*, pp. 1-8, 1995.

[36] A.K. Ghosh and A. Schwartzbard, A Study in Using Neural Networks for Anomaly and Misuse Detection.", *Proceedings of the 8th USENIX Security Symposium*, pp. 141-152, Aug. 1999.

[37] J. Ryan, M.J. Lin and R. Miikkulainen, "Intrusion detection with neural networks."*Advances in neural information processing systems*, pp. 943-949, 1998.

[38] L. Portnoy, E. Eskin and S. Stolfo, "Intrusion detection with unlabeled data using clustering."*In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*, pp. 1-14, 2001.

[39] L. Duan, L. Xu, Y. Liu and J. Lee, "Cluster-based outlier detection", *Annals of Operations Research*, vol. 168, no. 1, pp. 151-168, 2008.

[40] M. Breunig, H. Kriegel, R. Ng and J. Sander, "LOF", *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD '00*, 2000.

[41] L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, V. Kumar, J. Srivastava, and P. Dokas. "Minds-minnesota intrusion detection system."*Next generation data mining*, pp. 199-218, 2004.

[42] R. Agrawal, T. Imieliński and A. Swami, "Mining association rules between sets of items in large databases", *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*, 1993.

[43] J. Park, M. Chen and P. Yu, "An effective hash-based algorithm for mining association rules", *ACM SIGMOD Record*, vol. 24, no. 2, pp. 175-186, 1995.

[44] "Reverse Path Filtering", *Tldp.org*, 2017. [Online]. Available: http://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.kernel.rpf.html. [Accessed: 16- Mar- 2017].

[45] W. Fan, W. Lee, S. Stolfo and M. Miller, "A Multiple Model Cost-Sensitive Approach for Intrusion Detection", *Machine Learning: ECML 2000*, pp. 142-154, 2000.

[46] Wenke Lee, S. Stolfo, P. Chan, E. Eskin, Wei Fan, M. Miller, S. Hershkop and Junxin Zhang, "Real time data mining-based intrusion detection", *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, 2001.

[47] M. Mahoney, "Network traffic anomaly detection based on packet bytes", *Proceedings of the 2003 ACM symposium on Applied computing - SAC '03*, 2003.

[48] P. K. Chan, M. V. Mahoney, "PHAD: packet header anomaly detection for identifying hostile network traffic", *Florida Institute of Technology*, pp. 1-17, 2001.

[49] F. González and D. Dasgupta, *Genetic Programming and Evolvable Machines*, vol. 4, no. 4, pp. 383-403, 2003.

[50] Twitter, "twitter/AnomalyDetection", *GitHub*, 2015. [Online]. Available: https://github.com/twitter/AnomalyDetection. [Accessed: 3- Apr- 2017].

[51] NIST, "1.3.5.17.3. Generalized Extreme Studentized Deviate Test for Outliers", *Itl.nist.gov* ,2003. [Online]. Available: http://www.itl.nist.gov/div898/handbook/eda/section3/eda35h3.htm. [Accessed: 7- Apr- 2017].

[52] B. Rosner, "Percentage Points for a Generalized ESD Many-Outlier Procedure",*Technometrics*, pp. 165-172, May. 1983.

[53] R. Hyndman, "Cyclic and seasonal time series", *http://robjhyndman.com/hyndsight/*, 2011. [Online]. Available: http://robjhyndman.com/hyndsight/cyclicts/. [Accessed: 18- Apr- 2017].

Appendix 1 – Script extracting given type of logs for given server and team

```
import os
import sys
import subprocess
from datetime import datetime
import tzlocal  # $ pip install tzlocal
# python script.py blueXX mail ping
path  =  '/export2/ls16_study/scoring/LS16execution-score2-
day1.log.ipv4'
if sys.argv[3] == 'imapsmtppop3':
checkList= ['imap','smtp','smtps','pop3']
else:
checkList = [sys.argv[3]]
print checkList
trimmedlogfilename                                          =
path+'.'+sys.argv[1]+'.'+sys.argv[2]+'.'+sys.argv[3]

with  open(path)  as  logfile,  open("%s.%s.%s.%s"  %  (path,
sys.argv[1],     sys.argv[2],sys.argv[3])     ,'w')     as
trimmedlogfile:
for line in logfile:
if sys.argv[2]+'.'+sys.argv[1] in line:
if any(i in line for i in checkList):
trimmedlogfile.write(line)

subprocess.call(['Rscript',
'/export2/ls16_study/scripts/latest/convertTimes.R',
trimmedlogfilename])
```

Appendix 2 – Script converting scoring time value

```
args<-commandArgs(TRUE)

library(anytime)
df <- read.table(args[1], sep="|")
df$V5 <- as.numeric(df$V5)
df$V5 <- anytime(df$V5,tz='EET')
df$V5 <- df$V5 + 3600
write.table(df,args[1], col.names=FALSE, row.names=FALSE)
```

Appendix 3 – Script mapping files and tcpdumping

```
import subprocess
import sys
import os
import re
import collections

#python                                                script.py
/export2/ls16_study/scoring/LS16execution-score2-
day1.log.ipv4.blueXX.www.ping www.ping



path = sys.argv[1]
pcapPath = '/export2/ls16_study/team-XX/pcaps/'
timeList = []
filesToDump = []
pattern = '([0-9]+):([0-9]+):([0-9]+)'
pickedPcap = []
pcapPairs = {}



with open(path) as logfile:
     for line in logfile:
     match = re.search(pattern,line)
if (match):
     timeList.append(match.group(0))

for index in range(len(timeList)):
timeList[index]                                                =
str(int('20160420'+timeList[index].replace(':',""))       -
30000);
```

```
     tmpcap=''
for timeRange in timeList:
files = [f for f in os.listdir(pcapPath) if timeRange in f]
if files:
filesToDump.append(files[len(files) - 1])
          pcapPairs[timeRange] = files[len(files) - 1]
     else:
          files = [f for f in os.listdir(pcapPath) if
timeRange[:12] in f]
          tempcap = ''
          for filename in files:
                    if timeRange[12:] >= filename[22:-5]:
                         tempcap = filename
          if tempcap != '':
               filesToDump.append(tempcap)
               pcapPairs[timeRange] = tempcap

          else:
               files = [f for f in os.listdir(pcapPath) if
timeRange[:10] in f]
               tempcap = ''
               for filename in files:
                    if timeRange[10:-2] >= filename[20:-7]:
                         tempcap = filename
               if tempcap != '':
                    filesToDump.append(tempcap)
                    pcapPairs[timeRange] = tempcap
               else:
                    files = [f for f in
os.listdir(pcapPath) if timeRange[:8] in f]
                    tempcap = ''
                    for filename in files:
                         if timeRange[8:-4] >=
filename[18:-9]:
```

```
                    tempcap = filename
            if tempcap != '':
                filesToDump.append(tempcap)
                pcapPairs[timeRange] = tempcap
            else:
                print("There is no log file found
for %s" % f)

for filename in filesToDump:
my_cmd = ['sudo'] + ['tcpdump'] + ['-r'] +
[pcapPath+filename]
with open('/export2/ls16_study/team-
XX/latest_pcapTxts_%s_1/%s.txt' % (sys.argv[2], filename),
'w') as outfile:
subprocess.call(my_cmd, stdout=outfile)
```

Appendix 4 – Script counting packets

```
import subprocess
import sys
import os
import re
import mapAndDump #this is a little modified version of Appendix 3 we import. Difference is the algorithm is in main function and pcaPairs{} is returned in the end.

path = '/export2/ls16_study/practice/teamXX/logs/'
textPath = '/export2/ls16_study/team-XX/latest_pcapTxts_mail/'
timeList = []
pattern = '([0-9]+):([0-9]+):([0-9]+)'
countedPackets={}
textList={}
tmpTime = ''
counter = 0
ip = #SERVER IP
for logfilename in os.listdir(path):
fullpath = path + logfilename
with open(fullpath) as logfile:
for line in logfile:
match = re.search(pattern,line)
if (match):
timeList.append(match.group(0))
textList = mapAndDump.main()

for index in range(len(timeList)):
tmpTime = '20160420'+timeList[index].replace(':',"")
tmpTime = str(int(tmpTime)-30000)
```

```
if textList[tmpTime]:
counter = 0
          fulltextpath = textPath + str(textList[tmpTime])
+ '.txt'
with open(fulltextpath) as textfile:
     for tline in textfile:
     if timeList[index] in tline:
                          if ip in tline:
               counter = counter + 1
     countedPackets.setdefault(timeList[index],[]).append(c
ounter)

with                          open('/export2/ls16_study/team-
XX/countedPackets/mail_allpackets.txt', 'w') as file:
for time, packets in countedPackets.items():
file.write(time + '|')
file.write(str(packets))
file.write("\n")
file.close()
```