ANOMALY-BASED CYBER INTRUSION DETECTION SYSTEM WITH ENSEMBLE CLASSIFIER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

ALPER SARIKAYA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

SEPTEMBER 2018

ANOMALY-BASED CYBER INTRUSION DETECTION SYSTEM WITH ENSEMBLE CLASSIFIER

Submitted by ALPER SARIKAYA in partial fulfillment of the requirements for the degree of **Master of Science in Information Systems Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** ____________________

Prof. Dr. Yasemin Yardımcı Çetin
Head of Department, **Information Systems** ____________________

Assoc. Prof. Dr. Banu Günel Kılıç
Supervisor, **Information Systems Dept., METU** ____________________

Examining Committee Members:

Assoc. Prof. Dr. Altan Koçyiğit
Information Systems Dept., METU ____________________

Assoc. Prof. Dr. Banu Günel Kılıç
Information Systems Dept., METU ____________________

Assoc. Prof. Dr. Cengiz Acartürk
Cognitive Science Dept., METU ____________________

Asst. Prof. Dr. Pelin Angın
Computer Engineering Dept., METU ____________________

Assoc. Prof. Dr. Sevil Şen
Computer Engineering Dept., Hacettepe University ____________________

Date: 6 September 2018

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Alper SARIKAYA

Signature : __________________

ABSTRACT

ANOMALY-BASED CYBER INTRUSION DETECTION SYSTEM WITH ENSEMBLE CLASSIFIER

SARIKAYA, ALPER

MSc., Department of Information Systems

Supervisor: Assoc. Prof. Dr. Banu Günel Kılıç

September 2018, 61 pages

Nowadays, cyberattacks are occurring progressively. Along with this, diversity, size and density of the cyberattacks are increasing. When the logs of security devices are analyzed, massive amounts of attack signs are detained. Besides, it is also difficult for humans to evaluate the logs accurately. Therefore, the identification of key data, which can be used to distinguish an attack from this very large data set, is important for both rapid detection of attacks and rapid response of security devices. This study focuses on selection of appropriate features from logs via machine learning and determining the distinctive attributes specific to an attack in the selection of these data. Based on the selected features, a classification methodology is proposed. As a result, 80.20% overall accuracy has been achieved using the proposed model with 19 features. Moreover, a better detection rate on DoS and Exploit classes has been obtained.

Keywords: Cyberattack, Machine Learning, Intrusion Detection System

ÖZ

TOPLULUK ÖĞRENMESİYLE ANOMALİ TABANLI SİBER İHLAL TESPİT SİSTEMİ

SARIKAYA, ALPER
Yüksek Lisans, Bilişim Sistemleri Bölümü
Tez Yöneticisi: Doç. Dr. Banu Günel Kılıç

Eylül 2018, 61 sayfa

Günümüzde, siber saldırılar giderek artan bir şekilde meydana gelmektedir. Bununla birlikte, siber saldırıların çeşitliliği, büyüklüğü ve yoğunluğu artmaktadır. Güvenlik cihazlarının logları incelendiğinde, büyük miktarda saldırı izi elde edilmektedir. Ayrıca, insanlar için logların doğru olarak değerlendirmesi de zordur. Bu nedenle, bu çok büyük veri setinden bir saldırıyı ayırt etmek için kullanılabilecek anahtar verilerin tanımlanması hem saldırıların hızlı tespiti hem de güvenlik cihazlarının hızlı bir şekilde tepki göstermesi açısından önemlidir. Bu çalışma, makine öğrenmesi yoluyla loglardan uygun verilerin seçimine ve bu verilerin seçiminde bir saldırıya özgü ayırt edici özelliklerin belirlenmesine odaklanmaktadır. Seçilen özellikler kullanılarak, bir sınıflandırma metodolojisi önerilmiştir. Sonuç olarak, 19 özellik ile önerilen model kullanılarak %80,20 ortalama doğruluk başarılmıştır. Ayrıca, DoS ve Exploit sınıflarında daha iyi bir tespit oranı elde edilmiştir.

Anahtar Sözcükler: Siber Saldırı, Makine Öğrenmesi, Saldırı Tespit Sistemi

"Life is measured in achievement, not in years alone."
Bruce McLaren

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude and my respect to my thesis advisor Assoc.Prof.Dr. Banu Günel Kılıç for her priceless guidance, encouragement and continuous support to make this research possible.

I also thank my thesis jury members, Assoc. Prof. Dr. Altan Koçyiğit, Assoc. Prof. Dr. Cengiz Acartürk, Assoc. Prof. Dr. Sevil Şen, Asst. Prof. Dr. Pelin Angın for their suggestions and reviewing my work.

Special thanks to The Scientific and Technological Research Council of Turkey (TÜBİTAK) for 2210-A scholarship.

Finally, I would like to express my deepest gratitude to my wife, Özlem, for her love, encouragement and support in all my life. This thesis would not have been written without her.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACCS	Australian Centre for Cyber Security
ANOVA	Analysis of Variance
APT	Advanced Persistent Threat
ARFF	Attribute-Relation File Format
CATSUB	Clustering Categorical Data Based on Subspace
CVE	Common Vulnerabilities and Exposures
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DoS	Denial of Service
ENISA	European Union Agency for Network and Information Security
FN	False Negative
FP	False Positive
GA-LR	Genetic Algorithm–Logistic Regression
GB	Gigabyte
ID3	Iterative Dichotomizer 3
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
KDD	Knowledge Discovery and Data Mining
kNN	k-Nearest Neighbor
NP	Nondeterministic Polynomial Time
PCAP	Packet Capture
R2L	Remote to Local
RDP	Remote Desktop Protocol
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
SIEM	Security Information and Event Management
SQL	Structured Query Language
SVM	Support Vector Machine
TCP/IP	Transmission Control Protocol/Internet Protocol
TN	True Negative
TP	True Positive
TTL	Time to Live
U2R	User to Root

UNSW	University of New South Wales
VPN	Virtual Private Network
WAF	Web Application Firewall
XML	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1. Motivation

In today's world, the Internet is an indispensable need for humanity. In everyday life, people share their credit card information, bank accounts and many other sensitive private information through Internet. Besides, many commercial organizations and state agencies rely on the Internet. The networks are deeper and bigger than ever. For these reasons, keeping safe our resources, data, information and reputation are critical at the moment.

According to the European Union Agency for Network and Information Security (ENISA, 2017), the complexity of the attacks and sophistication of malicious actions continue to increase (Table 1). As of 2017, Malware, Web-based attack, Web application attack, Phishing, Spam and Denial of Service attacks were the main threats to our networks according to ENISA 2017 Cyber Threat Landscape Report. This report also emphasizes that the critical infrastructure is the main target for hackers. Therefore, cyberattacks which have evolved rapidly are instruments that target our networks and the information systems.

There are millions of different malwares, SQL Injection attacks and emerging XML injections. Each cyberattack type, such as Ransomware, DDoS, Web Application attack, have different characteristics. Therefore, writing a simple program to detect all cyberattack types is not possible. Since these attacks change continuously, it is essential to use adaptive technologies for cyberattack detection.

Cyber security is a set of technologies and functions designed to protect computers, networks, programs, resources and data from out of service attacks, unauthorized access, eavesdropping, change, or destruction. Designing an effective cyberattack detection mechanism requires many skilled tasks. High detection capability, quick response, preventive measures are important properties of these detection mechanisms.

There are many defense measures to protect a network and an information system, such as intrusion detection system (IDS), firewalls, anti-virus, and security information and event management (SIEM) (Zhong et al., 2018). Intrusion Detection/Prevention Systems (IDS/IPS) are important parts of network security to confront cyberattacks.

Table 1 2017 Cyber Threats and Status Change according to ENISA Threat Landscape Report 2017 (ENISA, 2017)

Threats in 2016	Threats in 2017	Status Change
1. Malware	1. Malware	↔
2. Web based attacks	2. Web based attacks	↔
3. Web application attacks	3. Web application attacks	↔
4. Denial of service	4. Phising	↑
5. Botnets	5. Spam	↑
6. Phising	6. Denial of service	↓
7. Spam	7. Ransomware	↑
8. Ransomware	8. Botnets	↓
9. Insider threat	9. Insider threat	↔
10. Physical manipulation / damage / theft / loss	10. Physical manipulation / damage / theft / loss	↔
11. Exploit kits	11. Data breaches	↑
12. Data breaches	12. Identity theft	↑
13. Identity theft	13. Information leakage	↑
14. Information leakage	14. Exploit kits	↓
15. Cyber espionage	15. Cyber espionage	↔

Machine learning based IDS models provide skillful techniques for network security and cyberattack detection. Generally, there are three IDS models based on machine learning: Signature-based, anomaly-based and hybrid (Buczak and Guven, 2016).

Signature-based (Misuse-based) IDS is designed to detect known attacks by using signatures of attacks. If attacks are known, they produce good detection rates and low false positive rates. In order to achieve better detection, frequent signature update is essential. Its main weak point is that signature-based techniques cannot detect zero-day attacks (Buczak and Guven, 2015). These are due to new vulnerabilities that emerge every day which hackers exploit quickly (Iglesias and Zseby, 2014).

Anomaly-based IDS analyzes the network traffic and the system behavior. It is simple to monitor normal network behavior. When it detects suspicious behavior, which is different from the normal network traffic, it alerts. They are attractive because of the ability to detect zero-day attacks (Gogoi et al., 2014). Also, abnormal activities detected from anomaly-based systems can be used to create an attack signature (Buczak and Guven, 2016) . Therefore, anomaly-based IDS is a much more preferred choice.

Hybrid IDS combines strong point of the signature-based and the anomaly-based IDS. Hybrid systems' main purposes are increasing the detection rates of known cyberattacks and decreasing false positive (FP) rates of unknown attacks (Buczak and Guven, 2016).

Another division of IDS is network-based or host-based. Simply, network-based IDS monitors all network traffic, while host-based IDS works on a specific host (Buczak and Guven, 2016).

However, in machine learning-based IDS, the critical point is that the network data or attack data set which are used for model training should be present for the real network behavior of today's attack types.

Even though IDS/IPS provide security against attacks, some attacks are hard to detect. Anomaly-based IDS may suffer from realizing attacks, which are carried out by highly skilled groups through various tactics and techniques. Especially, Advanced Persistent Threat (APT) is usually hard to detect by IDS/IPS.

1.2. Aim of the Study

This study aims to increase the detection rate of attack classes by an anomaly-based intrusion detection system.

The objectives of the study are:

- To analyze the effectiveness of a machine learning based IDS model against current attack types on a new cyberattack data set, University of New South Wales (UNSW)-NB15,
- To show the ability of hierarchical machine learning model for increasing the detection/accuracy rate,
- To reveal the difficulty of detecting some attack types,
- To understand the reasons of low detection rates for some attack types.

1.3. Scope of the Thesis

Within the scope of this thesis, cyberattack detection by a machine learning based IDS model was targeted. Only, supervised machine learning approach was considered. Random Forest Classifier was used for developing the model. For feature selection process, wrapper-based feature selection was selected.

The proposed model is an example of anomaly-based IDS. Signature-based IDS and hybrid-based IDS models are outside the scope of this thesis. In view of the data set, UNSW-NB15 cyberattack dataset was chosen for training and testing purposes.

Knowledge Discovery and Data Mining (KDD)-CUP 99 and NSL-KDD are well known data sets; however, they were not used in this thesis.

1.4. Outline of the Thesis

This document is organized as six chapters. The aim and scope of the thesis are stated in the first chapter. Chapter 2 defines the basis of machine learning and some developed IDS models in the literature. Chapter 3 explains the feature selection process and the proposed model. Chapter 4 provides the evaluation of the proposed model. Chapter 5 compares the proposed model with other studies. Chapter 6 concludes the thesis and provides the limitations and future work.

CHAPTER 2

MACHINE LEARNING AND INTRUSION DETECTION SYSTEM

In cyber security, machine learning has a strong background and versatile application. In this chapter, machine learning basis, intrusion detection models, discussion and criticism about previous studies are given.

2.1. Cyber Security and Network/Host Security Applications

Today, the information age presents many practicalities to us. People can easily check their bank accounts, look at social media, send e-mail to their employer, take a family photo and many other individual processes by using a computer, mobile phone or other smart devices. Although many commercial organizations and state agencies use the internet for their daily business, too, they have other special services, such as VPN, cloud services etc. Even military operations rely on the Internet, which shows us that the Internet is indispensable for our everyday lives.

Basically, the Internet consists of a distributed local network. We keep, work, produce and distribute our data through this network. For all of these reasons, protecting our network is critical. Cyber security is all the measures that protect our resource, network and data. It simply provides the hardware and software for this purpose.

However, hackers, hacktivists and even state sponsored hackers use the Internet for making money, deactivate devices, changing political behavior, destroying our resources, eavesdropping, changing information and many other purposes. These make cyber security an indispensable part of our network.

Some security software and hardware (Zhong et al., 2018) which are prevalent are;

- Antivirus / End-point security software,
- Data Loss Prevention Software,
- Software Based Firewall (Snort, Bro etc.),
- Hardware Based Firewall,

- Web Application Firewall (WAF),
- Load Balancer,
- Intrusion Detection / Prevention System (IDS/IPS),
- Security Information and Event Management (SIEM).

Among these, the most critical measure is the Intrusion Detection/Prevention System (IDS/IPS). IDS/IPS monitors all the network traffic and alerts when it detects an attack. Besides, IPS behaves actively to suppress the attack. The systems are suitable for network security and provide very detailed network traffic logs. They mostly use attack signatures to detect an attack and generate logs. However, periodical updates for attack signatures are critical for these systems.

There are many examples of software-based IDS/IPS. Surricata (Suricata, 2017), Snort (Roesch, 1999) and Bro (Sommer, 2016) are freeware versions of IDS/IPS systems. These applications use attack signatures and rely on daily/hourly signature update, which is critical for success. These applications produce logs which are related to signature/warning and errors. Often, the logs which are produced are hard to analyze and trace. A parsed log example of p0f is presented in Figure 1.


```
        Line 409755: { "_id" : { "$oid" :
"58c1d06f58e5cf04aff99ea3" }, "destination_ip" : "200.200.200.201",
"protocol" : "pcap", "hpfeed_id" : { "$oid" :
"58c1d06e58e5cf04aff99ea0" }, "timestamp" : { "$date" :
"2017-03-10T00:00:14.147+0200" }, "source_ip" : "221.229.162.121",
"source_port" : 4405, "destination_port" : 22, "identifier" :
"fea0bde0-5d6d-11e6-9709-000c297e338e", "honeypot" : "p0f" }
        Line 409756: { "_id" : { "$oid" :
"58c1d06f58e5cf04aff99ea4" }, "protocol" : "ssh", "hpfeed_id" :
{ "$oid" : "58c1d06e58e5cf04aff99e9f" }, "timestamp" : { "$date" :
"2017-03-10T00:00:14.139+0200" }, "source_ip" : "221.229.162.121",
"session_ssh" : { "version" : "SSH-2.0-libssh2_1.4.3" },
"source_port" : 4405, "destination_port" : 22, "identifier" :
"fb52256e-5d6a-11e6-9709-000c297e338e", "honeypot" : "kippo",
"auth_attempts" : [ { "login" : "admin", "password" : "01234567" },
{ "login" : "admin", "password" : "admin520" }, { "login" : "admin",
"password" : "1q2w3e4r5t" }, { "login" : "admin", "password" :
"password1" }, { "login" : "admin", "password" : "P@ssword" } ] }
```


Figure 1 An example of p0f warning/alerts log

Network security engineers actively use IDS/IPS, firewalls and other security software/hardware in many networks. However, despite the active/passive measures, logs which are produced by this security software/hardware are plenty and hard to evaluate manually. For example, port scan is an alert for many IDS/IPS. They produce thousands of logs in every hour. On the other hand, a malicious request to port 139 with RDP is much more critical than port scan and this might happen once a week. Moreover, many IDS/IPS provide alerts or warnings but all decisions are taken by a security expert, which is not ideal for network security.

2.2. Machine Learning

Machine learning is a subset of artificial intelligence that a program learns from experience with help of data without being explicitly programmed. The main idea behind machine learning is making a human level task. Machine learning helps us in many fields: spam detection, face recognition, drug discovery, driverless car, cyberattack detection, speech recognition, budget expectation, etc.

Machine learning methods usually produce good accuracy/detection result for cyberattack detection (Buczak & Guven, 2016). However, in anomaly detection research, Tavallaee et al. (2010) point out the importance of data sets, methods on experiments and performance evaluation criteria.

There are four types of machine learning:

- Supervised Learning
- Semi-supervised Learning
- Unsupervised Learning
- Reinforcement Learning

2.2.1 Supervised and Semi-supervised Machine Learning

In supervised learning, the model uses labeled data to learn from experience. Every instance in the data set has features and a class label. The model uses features and label to build up a classifier. After training, the model predicts new instances.

Classification and regression are the main tasks in supervised learning. In classification problem, the model uses instance's features to assign it to a class. In regression problem, the model uses the features to predict a value. Decision tree (Quinlan, 1999), Naive Bayes (Mukherjee and Sharma, 2012), Support Vector Machine (Cortes and Vapnik, 1995), Multilayer Perceptron (Pal and Mitra, 1992) are examples of supervised learning.

Decision Tree is a well-known machine learning method, because, it is robust to noisy data and capabilities of learning disjunctive expressions (Quinlan, 1999). C4.5 is the mostly preferred type of decision trees (Quinlan, 1992). A simple example of the decision tree structure of IRIS data set is presented in Figure 2.

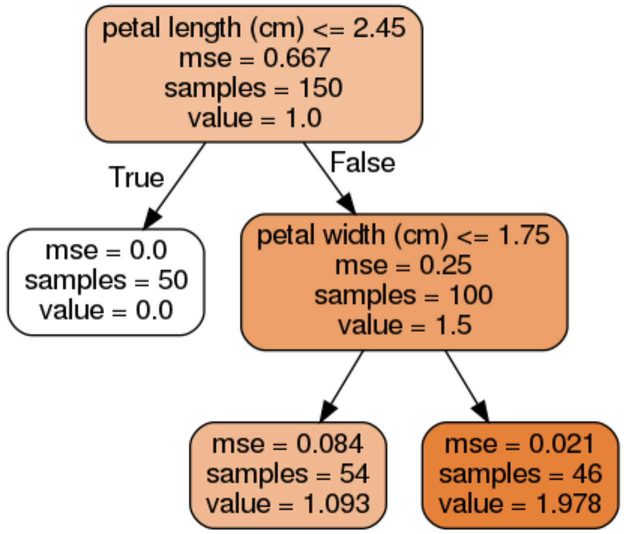

Figure 2 Example of Decision Tree Structure

k-Nearest neighbor is another machine learning method (Cunningham and Delany, 2007). It is an instance-based model. This type of model simply keeps the training example and uses them for classification. It is a lazy method because it delays learning until a new instance is to be classified (Mitchell, 1997).

Support vector machine (SVM) is another widely used machine learning model (Cortes and Vapnik, 1995). It is capable of linear or nonlinear classification, regression and outlier detection. It divides the classes using linear, polynomial or radial kernel.

Ensemble learning methods combine multiple weak classifiers to improve the model accuracy (Buczak and Guven, 2016). Random forest is an ensemble classifier which combines multiple decision trees (Oshiro et al., 2012). Prediction of input is done with voting of decision trees. It is efficient in large data set.

2.2.2 Unsupervised Machine Learning

In unsupervised learning, labelled data is not available, and the model uses clustering and grouping for analyzing future data. Expectation-maximization (Do and Batzoglou, 2008), outlier (Ben-Gal, 2005) and clustering (Jain et al., 1999) are examples of unsupervised learning.

Clustering is a method for grouping objects using similarity within group. It is essential that similarity within a cluster is high as well. An outlier means that the data is very different from the rest of the data (Gogoi et al., 2014)

2.2.3 Reinforcement Learning

Last type of machine learning is reinforcement learning. There is an agent for future acts and each act has a reward or penalty. The purpose of agent's policy is the maximization of the total reward or minimization of the total penalty. Q-learning (Watkins and Dayan, 1992) and Boltzmann Machine (Hinton and Salakhutdinov, 2009) are examples of reinforcement learning.

2.3 Feature Selection and Applications

Despite its benefits and good problem-solving ability, one of the main problems of machine learning is the feature size in the data set. Many machine learning models use thousands of features for training. The more features we use in machine learning application, the more this may yield slow training, more overfitting and curse of dimensionality (Iglesias and Zseby, 2015).

Feature selection acts to solve these problems. Many researchers emphasize the importance of feature selection in machine learning research (Li et al., 2009; Chandrashekar and Sahin, 2014; Iglesias and Zseby, 2015). Removing unnecessary and trivial features reduces the training time and increases accuracy. Guyon and Elisseeff (2003) also assert the importance of feature selection. They argue that feature selection improves the performance of a classifier, reduces the computational effort and presents better data understanding.

Aldehim and Wang (2017) attempts to figure out how to use the data set in feature selection methods. They compared full data sets and part of the data set for the feature selection process and found that if the data set had large enough data, two methods produced the same result.

Najafabadi et al. (2016) also work on feature selection. They used the Kyoto Data set which has 24 features and kNN, C4.5, Naive Bayes classifiers. They also conducted an ANOVA test for measuring the significance of the feature selection process. ANOVA test results showed that the feature selection is important for data preprocessing. Reduced feature data set provided better results than all-features data set.

There are two main machine learning strategies for feature selection: Filter-based and wrapper-based (Khor et al., 2009; Chandrashekar and Sahin, 2014). Filter-based feature selection uses ranking techniques as the principle criteria for variable selection by ordering (Chandrashekar and Sahin, 2014). On the other hand, wrapper-based feature selection uses a classification model to evaluate the feature subsets. It is a cluster-based approach (Ladha and Deepa, 2011).

2.3.1 Filter-based Feature Selection

Filter-based methods use feature ranking for detecting important features. If a feature has no effect over a class, this feature is discarded (Chandrashekar and Sahin, 2014). A threshold can be used for the elimination process. Using the correlation criteria is one of the ranking methods. It simply calculates the Pearson correlation coefficient between a feature and a class label:

$$R_i = \frac{cov(x_i, Y)}{\sqrt{var(x_i) \times var(Y)}} \tag{2.1}$$

where x_i is i^{th} feature, *Y* is the class label, *cov()* is the covariance and *var()* is the variance. Filter-based methods are quick and the result is less overfitting. However, the main disadvantage of these methods is that the selected subset may not be an optimal subset in that a redundant subset can be obtained (Chandrashekar and Sahin, 2014).

2.3.2 Wrapper-based Feature Selection

Wrapper-based feature selection uses the Black-Box technique for ranking features (Chandrashekar and Sahin, 2014). In every iteration, it adds a feature to the subset and then evaluates the classifier success. Successful feature is kept in the subset. If the data set contains n features, there are 2^n available subsets and this is called an NP-hard problem. Some optimized algorithms, such as the Genetic Algorithm or Particle Swarm Optimization present feasible computational subsets. As a result, the main disadvantage of wrapper-based methods is the computational power. However, once a subset is selected, it is much more successful than using a subset obtained by filter-based feature selection.

2.4 Machine Learning Based Intrusion Detection System Models

For cyber security, machine learning studies generally use packet-level data (Cannady, 1998), NetFlow oriented data (Apiletti et al., 2009) and public data sets (Buczak and Guven, 2016). Packet-level data is obtained from the physical interface of computers, switches or routers, etc. They are saved as Packet Capture (PCAP) file format generally. NetFlow, which is a property of CISCO, is another type of network packet traffic collection. It is simply compressed and preprocessed version of network traffic. However, the mostly used data in the cyber-security field is public data sets. They are generally Defense Advanced Research Projects Agency (DARPA), KDD-CUP 99 and NSL-KDD data set.

DARPA 1998 data set was created in Massachusetts Institute of Technology Lincoln Laboratory for testing of the IDS/IPS. A simulation network was used for the traffic capture and nine weeks' data was created. First seven weeks of data was used for the training set and last two weeks of data was used for the testing set. There are four attack classes: Denial of Service (DoS), Probe, Remote-to-Local (R2L), User-to-Root (U2R).

KDD-CUP 99 is created from DARPA 1998 Transmission Control Protocol/Internet Protocol (TCP/IP) data. This data set has 41 attributes and three base components: Basic, content and traffic features. It has four attack classes: Normal, Denial of Service, Probe, Remote-to-Local, User-to-Root. However, this data set is obsolete and has some drawbacks (McHugh, 2000; Tavallaee et al., 2009; Gogoi et al., 2012). For example, time to live (TTL) value in attack data packet is 126 or 153, but these values do not occur in the training records of attack data, the probability distributions of testing and training sets are different from each other. Also, the data set does not include a low foot print attack (Moustafa and Slay, 2015).

UNSW (University of New South Wales)-NB15 attack data set was created by the Australian Centre for Cyber Security (ACCS) to present a more realistic data set. Common Vulnerabilities and Exposures (https://cve.mitre.org) were used for the attack generation. Attack types in data set and features are explained in Section 3.1. There are a number of anomaly-based IDS researches based on this data set (Janarthanan and Zargari, 2017; Khammassi and Krichen, 2017; Moustafa and Slay, 2017; Papamartzivanos et al., 2018; Nawir et al., 2018).

Tavallaee et al. (2009) proposed NSL-KDD data set. Simply, it consists of selected records of KDD-CUP 99 dataset. The inherent problems of KDD-CUP 99 were tried to be solved, but still this data set has some drawbacks as mentioned by (McHugh, 2000).

General approaches in machine learning based intrusion detection systems can be classified into three groups: anomaly-based, signature-based and hybrid. While some researchers focus on only the attack detection (Normal or Attack), others focus on the attack classification (Classification of each specific type of attack).

Shon and Moon (2007) propose a hybrid SVM classifier for combining one-class SVM (unsupervised) and Soft-Margin SVM (supervised). They work on DARPA 1999 data set and use filter-based feature selection algorithm. Also, they use real-time network traffic to test the unsupervised approach. Their research provides 99.90% accuracy on real time network traffic and on DARPA 99 data set accuracy rate is nearly equal to Snort and Bro detection rate which is about 94.19%. This model only detects anomaly; no attack classification is specified.

Pervez and Farid (2014) test another SVM model. They use the NSL-KDD data set. Their approach is attack detection and they use filter-based feature selection. Using only 14 features rather than 41, they achieve 82.68% accuracy rate. Using only three features, they achieve 78.85%. However, the accuracy of each attack class is not provided.

Gogoi et al. (2014) propose the combination of supervised and unsupervised classifiers using KDD-CUP 99, NSL-KDD and real time traffic. The developed version of clustering categorical data based on subspace (CatSub) supervised model (CatSub+) is used for DoS/Probe detection; K-point unsupervised method is used for normal traffic and outlier model is used for User-to-Root and Remote-to-Local attacks detection in a multistage manner. In all classes, they achieve better accuracy rates by using only C4.5 Decision Tree or SVM. It is clear that the combination of classifiers provides better accuracy.

Bolón-Canedo et al. (2011) work on the multiclass classification problem. They use KDD-CUP 99 data set with C4.5 decision tree and Naïve Bayes classifiers. They try two different approaches: multiple class algorithm and multiple binary classifier, but none of them achieve the KDD winner result (Detection rate of Normal:99.45%, U2R:13.16%, DoS: 97.12, Probe: 83.32%). They also assert that Naïve Bayes and decision tree are applicable for large databases.

Moustafa and Slay (2017) work on hybrid feature selection method using NSL-KDD and UNSW-NB15 data sets. They use Naïve Bayes, expectation-maximization and logistic regression classifier. Results show that only 11 features are enough to yield better results for both data sets. Logistic regression for UNSW-NB15 provide 83% accuracy. UNSW-NB15 data set is considered as a complex data set due to behaviors of attack and normal network traffic (Moustafa and Slay, 2016).

Khammassi and Krichen (2017) use Genetic Algorithm-Logistic Regression (GA-LR) for feature selection and C4.5 decision tree for multiclass classification on UNSW-NB15 and KDD-CUP 99 data set. Using only 20 features on UNSW-NB15 data set, they achieve 81.42% accuracy. Also, they agree that UNSW-NB15 data set is a more complex data set than KDD-CUP 99 data set. These two assertions should be tested with a new machine learning model.

Janarthanan and Zargari (2017) analyze UNSW-NB15 and KDD-CUP 99 data set's features for effective network intrusion detection system. According to their findings, *service, sbytes, sttl, smean and ct_dst_sport_ltm* are significant features in UNSW-NB15 data set.

Nawir et al. (2018) use Average One Dependence Estimator, Bayesian Network and Naive Bayes for binary classification on UNSW-NB15 data set. Average One Dependence Estimator is the best classifier with 94.37% accuracy and Naive Bayes is the worst classifier with 75.73% accuracy. They also argue that UNSW-NB15 is relevant for anomaly detection due to synthesized attack patterns. However, they develop models against binary classification.

Table 2 Some example of Machine Learning based Intrusion Detection Models

ML Model	**Authors**	**Approaches**	**Data set Used**
Bayesian Net	Khor et al., 2009	Anomaly	KDD-CUP 99
Bayesian Net	Kruegel, Mutz, Robertson, & Valeur, 2003	Anomaly	DARPA 1999
Decision Tree	Eesa, Orman, & Brifcani, 2015	Anomaly	KDD-CUP 99
Decision Tree	Akyol, Hacibeyoglu, & Karlik, 2016	Anomaly	KDD-CUP 99, ISCX 2012

Decision Tree & Naive Byes	Bolón-Canedo et al., 2011	Anomaly	KDD-CUP 99
Ensemble-Random Forest	Jiong Zhang, Zulkernine, & Haque, 2008	Anomaly, Misuse	KDD-CUP 99
k-NN	Leung & Leckie, 2005	Anomaly	KDD-CUP 99
k-NN, Decision Tree, Naive Bayes	Najafabadi et al., 2016	Anomaly	KYOTO 2006
Logistic Regression	Khammassi & Krichen, 2017	Anomaly	UNSW-NB15, KDD-CUP 99
Naive Bayes & Decision Tree	Amor, Benferhat, & Elouedi, 2004	Anomaly	KDD-CUP 99
Neural Networks	Cordella & Sansone, 2007	Anomaly	KDD-CUP 99
Neural Networks	Cannady, 1998	Misuse	Real time Network Traffic
SVM	Aburomman & Ibne Reaz, 2017	Anomaly	NSL-KDD
SVM	Shon & Moon, 2007	Anomaly, Misuse	DARPA 99
SVM	Ganapathy, Yogesh, & Kannan, 2012	Anomaly	KDD-CUP 99
SVM	Pervez & Farid, 2014	Anomaly	NSL-KDD

Attack classification, which involves more than two classes, is a more difficult problem than attack detection, which involves just a binary decision. Moreover, in attack classification, the classifier's performance may vary according to the attack class. While the accuracy for some classes may be high, for other classes it may be worse than expected (Gogoi et al., 2014). To overcome this accuracy problem, multi-level model might be a solution.

2.5 Performance Metrics

Accuracy, Precision, Recall and F1-Score are used for measuring the performance of models. For multiclass classification, overall accuracy, class detection rate and class FP rate are used. *True Positive* (TP), i.e., positive instances that are classified as a positive; *True Negative* (TN), i.e., negative instances that are classified as a negative; *False Positive* (FP), i.e., negative instances that are classified as a positive; *False Negative* (FN), i.e., positive instances that are classified as a negative, are our basis terms.

Accuracy is the percentage of correctly classified instances and is measured as

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \ . \tag{2.2}$$

Precision is the percentage of positive instances that are correctly labeled and is measured as

$$Precision = \frac{TP}{TP+FP} \ . \tag{2.3}$$

Recall (Detection Rate) is the percentage of actually positive instances and is measured as

$$Recall = \frac{TP}{TP+FN} \ . \tag{2.4}$$

F1-Score is the weighted average of the precision and recall:

$$F1 - Score = 2 \times \frac{(precision \times recall)}{(precision + recall)} \ . \tag{2.5}$$

Overall accuracy is exemplars classified correctly from all exemplars.

Class detection rate is the ratio of exemplars classified correctly to all exemplars from the given class.

Class FP rate is the ratio of exemplars classified incorrectly from given class to all exemplars not from the given class.

Receiver Operating Characteristic curve (ROC curve) is used for the true positive rate against the false positive rate at various threshold settings.

CHAPTER 3

METHODOLOGY

In this chapter, UNSW-NB15 data set, wrapper feature selection method, the results of the feature selection process and the proposed machine learning model are explained. The feature selection method gives us some initial insights about attack classification. Then, the proposed model and the usage of data set are described.

3.1. UNSW-NB15 Data set

UNSW (University of New South Wales)-NB15 Network data set was created by the Australian Centre for Cyber Security (ACCS) to present a more realistic data set. The main idea behind this data set is projection of today's network behavior. IXIA PerfectStorm is used for network traffic simulation. This appliance uses the current Common Vulnerability and Exposure (CVE) list and simulates a specific attack type. By using these latest vulnerabilities and exposures, more realistic attack behaviors are created. The data set is publicly available at https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Data sets. Some selected ground tables of attack classes are presented in Appendix-C.

Tcpdump software was used for capturing the network traffic info into *.pcap format. Totally, 100 GB network traffic was captured. Bro and Argus security tools and scripts were used for creating the features and instance's labels.

There are nine different types of attacks in this data set, which can represent the situation of today's network. Detailed examples of each attack classes are presented in Appendices-C.

Attack types are:

(1) **Analysis:** to penetrate a web application via emails, web scripts etc.
(2) **Backdoor:** to bypass authentication and unauthorized access
(3) **DoS:** to attempt to use up resources of a target
(4) **Exploit:** to benefit from glitch, vulnerabilities and bugs.
(5) **Fuzzers:** to discover vulnerabilities
(6) **Generic:** a technique against the block-ciphers using hash function
(7) **Reconnaissance:** to gather information about a target

(8) Shellcode: piece of code that enables making a target vulnerable
(9) Worm: spreadable small and malicious program

Developers of this data set have created a training set of 175,341 instances and a testing set of 82,332 instances, separately in *.csv format. Traffic category ratios in two sets are nearly the same and well structured. Instance sizes of training and testing sets and attack size ratios are presented in Table 3:

Table 3 UNSW-NB15 Training and Testing Set Statistical Information

	Testing Set			**Training Set**		
Category	**Instance Size**	**Ratio in Testing Set (%)**	**Ratio in Attacks**	**Instance Size**	**Ratio in Training Set (%)**	**Ratio in Attacks**
Normal	37,000	44,94	-	56,000	31,94	-
Analysis	677	0,82	1,49	2000	1,14	1,68
Backdoor	583	0,71	1,29	1746	1,00	1,46
DoS	4,089	4,97	9,02	12,264	6,99	10,28
Exploits	11,132	13,52	24,56	33,393	19,04	27,98
Fuzzers	6,062	7,36	13,37	18,184	10,37	15,24
Generic	18,871	22,92	41,63	40,000	22,81	33,52
Reconnaissance	3,496	4,25	7,71	10,491	5,98	8,79
Shellcode	378	0,46	0,83	1,133	0,65	0,95
Worms	44	0,05	0,10	130	0,07	0,11
Total	***82,332***			***175,341***		

Moustafa and Slay (2016) argue that UNSW-NB15 data set can be considered as a complex data set, as it shows the same characteristics of a modern network traffic.

Originally, there are 49 features in UNSW-NB15. Detailed information about these features is presented in Table 4. In training and testing sets (csv files), only 43 features were used by the creators of this data set. Srcip, sport, dstip, dsport, stime and ltime features were excluded from the testing and training csv files. These six features are related to individual characteristics of the system and are not features for attack-detection. These features were also excluded in our work.

Before applying the feature selection process, nominal features were converted to integer format. In testing and training sets, each nominal value has a unique integer value. After the preprocessing, the testing set (82,332 instances) was used to select the features and obtain initial insights about the accuracy of attack detection.

Table 4 UNSW-NB15 Data Set Features

	Feature	Type	Info
Flow Features	srcip*	nominal	Source IP address
	sport*	integer	Source port number
	dstip*	nominal	Destination IP address
	dsport*	integer	Destination port number
	proto	nominal	Transaction protocol
Basic Features	state	nominal	Indicates to the state and its dependent protocol
	dur	Float	Record total duration
	sbytes	Integer	Source to destination transaction bytes
	dbytes	Integer	Destination to source transaction bytes
	sttl	Integer	Source to destination time to live value
	dttl	Integer	Destination to source time to live value
	sloss	Integer	Source packets retransmitted or dropped
	dloss	Integer	Destination packets retransmitted or dropped
	service	nominal	http, ftp, smtp, ssh, dns, ftp-data, irc and (-) if not much used service
	Sload	Float	Source bits per second
	Dload	Float	Destination bits per second
	Spkts	integer	Source to destination packet count
	Dpkts	integer	Destination to source packet count
Content Features	swin	integer	Source TCP window advertisement value
	dwin	integer	Destination TCP window advertisement value
	stcpb	integer	Source TCP base sequence number
	dtcpb	integer	Destination TCP base sequence number
	smean	integer	Mean of the low packet size transmitted by the src
	dmean	integer	Mean of the low packet size transmitted by the dst
	trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
	res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server's http service.
Time Features	Sjit	Float	Source jitter (mSec)
	Djit	Float	Destination jitter (mSec)
	Stime*	Timestamp	Record start time
	Ltime*	Timestamp	Record last time
	Sintpkt	Float	Source inter packet arrival time (mSec)
	Dintpkt	Float	Destination interpacket arrival time (mSec)
	tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.

(* marks the features that are excluded from the training and testing sets.)

Connection Features	is_sm_ips_ports	Binary	If source and destination IP addresses equal and port numbers equal then, this variable takes value 1 else 0
	ct_state_ttl	Integer	No. for each state according to specific range of values for source/destination time to live.
	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
	is_ftp_login	Binary	If the ftp session is accessed by user and password, then 1 else 0.
	ct_ftp_cmd	integer	No of flows that has a command in ftp session.
	ct_srv_src	integer	No. of connections that contain the same service and source address in 100 connections according to the last time.
	ct_srv_dst	integer	No. of connections that contain the same service and destination address in 100 connections according to the last time.
	ct_dst_ltm	integer	No. of connections of the same destination address in 100 connections according to the last time.
	ct_src_ ltm	integer	No. of connections of the same source address in 100 connections according to the last time.
	ct_src_dport_ltm	integer	No of connections of the same source address and the destination port in 100 connections according to the last time.
	ct_dst_sport_ltm	integer	No of connections of the same destination address and the source port in 100 connections according to the last time.
	ct_dst_src_ltm	integer	No of connections of the same source and the destination address in in 100 connections according to the last time.
Label	attack_cat	nominal	The name of each attack category.
	Label	binary	0 for normal and 1 for attack records

3.2. Feature Selection

After data preprocessing, feature selection process was applied to the UNSW-NB 15 testing set (82,332 instances). WEKA tool (3.8.2 version), which is a tool for machine learning and data mining task, was used for this purpose.

With the feature selection method, we aim to reduce the computation time, improve the accuracy of the model and reduce the data-size for saving. Wrapper feature selection method was applied.

To use WEKA effectively, *.csv files were converted to *.arff. Attribute-Relation File Format (ARFF) which is a file type used by WEKA tools. Simply, it has two distinct sections: Header and Data.

The header of the ARFF file contains the relation names and attributes/types of features. Data starts with '@DATA' mark and continues. Using the WEKA 'Explorer' function, UNSW-NB15 testing set is loaded to WEKA.

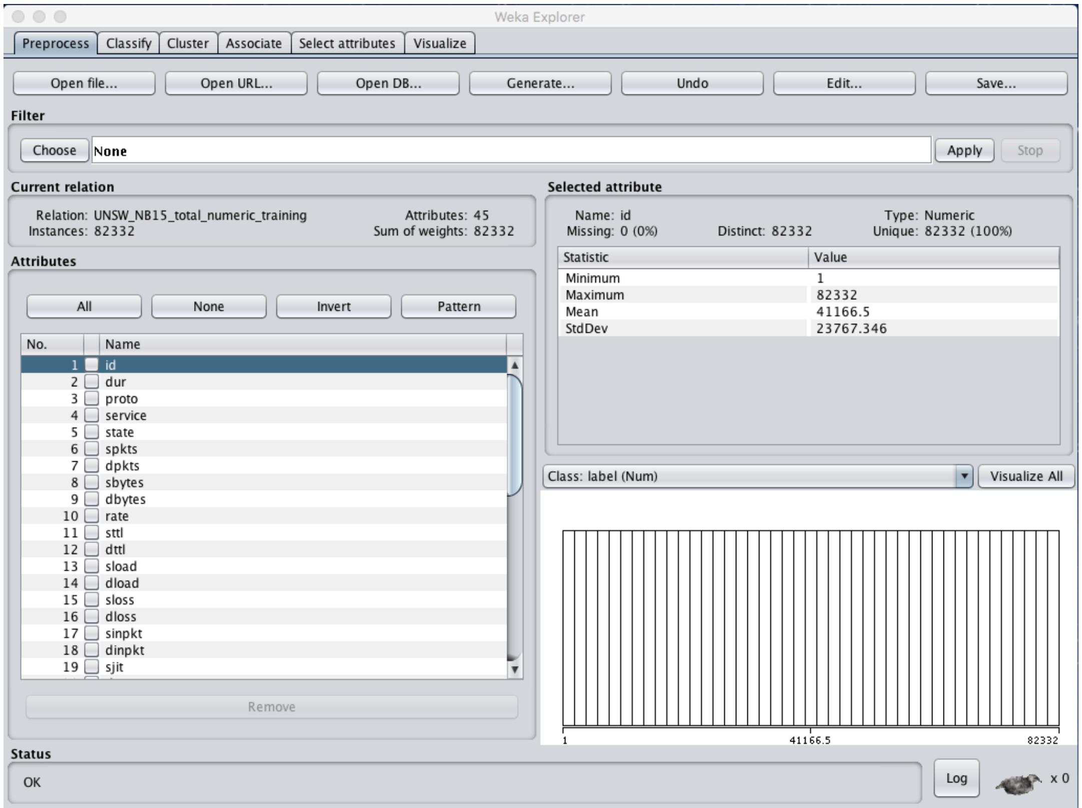

Figure 3 UNSW-NB15 Testing Set for Feature Selection by WEKA

In this research, the feature set is created only once during the whole process. To achieve good multiclass accuracy, good feature set is important. For this reason, the wrapper method was preferred, rather than the filter method.

WEKA's "Feature Selection" module was used. "WrapperSubsetEval" was selected from the *Attribute Evaluator*, "J48 Decision Tree" was selected for estimating the subset accuracy and "Greedy Stepwise" was selected as the *Search Method.* Greedy Stepwise search method is a heuristic method. In each step, it finds the locally optimal choice. Feature selection criteria in this set is "attack_cat". After applying the feature selection options, 19 features were selected by the wrapper feature selection method (Figure 4).

```
Attribute Subset Evaluator (supervised, Class (nominal): 43 attack_cat):
    Wrapper Subset Evaluator
    Learning scheme: weka.classifiers.trees.J48
    Scheme options: -C 0.25 -M 2
    Subset evaluation: classification accuracy
    Number of folds for accuracy estimation: 5

Selected attributes: 3,4,7,8,10,11,12,14,15,27,28,30,31,34,35,36,37,38,41 : 19
                     service
                     state
                     sbytes
                     dbytes
                     sttl
                     dttl
                     sload
                     sloss
                     dloss
                     smean
                     dmean
                     response_body_len
                     ct_srv_src
                     ct_src_dport_ltm
                     ct_dst_sport_ltm
                     ct_dst_src_ltm
                     is_ftp_login
                     ct_ftp_cmd
                     ct_srv_dst
```

Figure 4 Result of Wrapper-based Feature Selection

The selected features by wrapper-based method are presented in Table 5. When the selected features were analyzed, the most important connection properties were found to be Time-to-live (TTL) value, dropped packet size in both directions, transmit packet size in both directions, bit size per second, number of connection attempts and mean of data size in the upper layer. Basic features and connection features are more important feature groups according UNSW-NB15 data set. Since all of these selected features are related to upper layer protocols such as File Transfer Protocol or Hyper Text Transfer Protocol and carried the connection information about application layer protocols, these distinct features are selected by wrapper-based feature selection.

Table 5 Selected Features in UNSW-NB15 Testing Set by Wrapper Method

Selected Features
service, state, sbytes, dbytes, sttl, dttl, sload, sloss, dloss, smean, dmean, response_body_len, ct_srv_src, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, is_ftp_login, ct_ftp_cmd, ct_srv_dst

Filter-based feature selection method was also conducted for comparison with the wrapper-based feature selection method. *"Correlation-based Feature Selection"* was used as the feature evaluation criteria and "Particle Swarm Optimization" was used as the search algorithm. The selected features by the filter-based method are presented in Table 6.

Table 6 Selected Features in UNSW-NB15 Testing Set by Filter Method

Selected Features
proto, service, sbytes, sttl, smean, ct_src_dport_ltm, ct_dst_sport_ltm

Previously, Khammassi and Krichen (2017) has built a GA-LR model and selected 20 features on the UNSW-NB15 data set which are listed in Table 7. GA-LR model is a filter-based feature selection model. However, their 11 features are also present in our feature set.

Table 7 Selected Features by Khammassi and Krichen

Selected Features by Khammassi and Krichen
proto, **service**, **state**, spkts, dpkts, **sbytes**, **dbytes**, **dttl**, **dloss**, sinpkt, djit, swin, tcprtt, **smean**, **dmean**, trans_depth, **response_body_len**, **ct_srv_src**, **ct_dst_sport_ltm**, is_sm_ips_ports

3.2.1. Result of Feature Selections

19 features were selected by the wrapper-based feature selection method. This is fewer than half of all the features. Besides, seven features were selected by the filter-based feature selection method. To measure the success of the feature selection, Decision Tree classifier was used and 10-fold cross validation was applied. Results show that the model overall accuracy using 19 features is better than using all the features or using seven features (Table 8).

Table 8 Overall Accuracy of Decision Tree Using Three Different subsets

	All Features (43 Features)	**Features Selected by Wrapper Method (19 Features)**	**Features Selected by Filter Method (7 Features)**
Model Overall Accuracy	87.80%	88.20%	83.25%

Meanwhile, to measure the effectiveness of the wrapper-based feature selection, kNN, decision tree and neural network were used against multiclass classification. Decision tree and kNN were set for 10-fold cross validation and neural network was set for 70%:30% training/testing ratio. Overall accuracy results are presented in Table 9:

Table 9 Results of classifier with 19 features and 43 features in UNSW-NB15 Testing Set

	Decision Tree (Accuracy – Building Time)	kNN (Accuracy – Building Time)	Neural Network (Accuracy – Building Time)
All Features (43)	87.80%	80.71%	**85.45**%
	15.72 seconds	0.03 seconds	810.83 seconds
Wrapper-Based Subset (19)	**88.19**%	**85.80**%	83.85%
	8.03 seconds	**0.01** seconds	**299.63** seconds

3.2.2. Initial Insights about the Accuracy of Decision Tree Classifier

Except the neural network, other two classifiers provided better overall accuracies with reduced features by wrapper-based method. Besides, the model-built time in all classifiers decreased. Therefore, it can be said that the wrapper feature selection process provides more accurate results and less computational resource than using all the features.

Decision tree performed the best among the classifiers tested. Generally, the decision tree classifier works well at multiclass classification. Due to its better overall accuracy and more versatile structure, the decision tree model is used for multiclass classification.

The confusion matrix of the decision tree model is presented in Table 10. According to the confusion matrix, Normal, Generic, Recon. attack types have fairly good detection rates. Analysis, Backdoor, Shellcode and Worms have small size instances and the reason behind low detection rates might be this low instance size. Despite the fact that DoS, Exploit and Fuzzers have enough instances in the training phase, their detection rate is worse than expected. It is clear that the more problematic attack classes in this data set are DoS, Exploit and Fuzzers. These attack classes also interfere with each other.

Despite the issue of misclassification, this model is good at detecting anomalies with 98.3% detection rate for normal traffic and only 2.9% false positive rate. It is also very promising for traffic labeling.

Table 10 Confusion Matrix of Decision Tree Classifier with 19 Features

		Predicted Class									
		Normal	**Analysis**	**Backdoor**	**DoS**	**Exploit**	**Fuzzers**	**Generic**	**Recon**	**Shellcode**	**Worms**
True Class	Normal	36360	0	0	33	141	371	16	19	60	0
	Analysis	1	53	0	150	289	184	0	0	0	0
	Backdoor	4	0	20	46	310	195	3	2	3	0
	DoS	59	3	6	2471	1210	233	41	26	39	1
	Exploit	319	3	14	2007	7943	528	127	143	38	10
	Fuzzers	803	0	0	297	694	4231	11	15	9	2
	Generic	25	1	6	72	208	36	18504	3	13	3
	Recon	29	0	3	268	356	19	6	2805	10	0
	Shellcode	55	0	1	25	49	25	4	17	202	0
	Worms	3	0	0	0	12	2	2	0	2	23

At this point, our focus turns on solving the poor classification problem. Clearly, DoS, Exploit and Fuzzers are critical attack types and have enough instances for training. It is obvious that a stronger detection mechanism must be developed for attack detection.

3.3. Proposed Method: Hierarchical Multiclass Classifier

Confusion matrix tells us that in UNSW-NB15 Testing set (82,332 instances);

1. DoS class has 4,089 instances. Detection rate is 60.4%, but 1,210 (29.5%) instances intervene with "Exploit" class,

2. Exploit class has 11,132 instances. Detection rate is 71.2%, but 2,007 (18%) instances intervene with "DoS" class,

3. Fuzzers class has 6,062 instances, Detection rate is 69.7%, but 803 (13.2%) instances intervene with "Normal" class and 694 (11.4%) instances intervene with "Exploit" class.

The idea behind Hierarchical Multiclass Classifier model is that if confused classes are separated from other classes, class accuracy and average accuracy of multiclass classification will increase. A classifier which works on multiple classes may not achieve better accuracy for all classes. An appropriate combination of customized classifiers in a hierarchical manner may increase the classification accuracy of all attack classes.

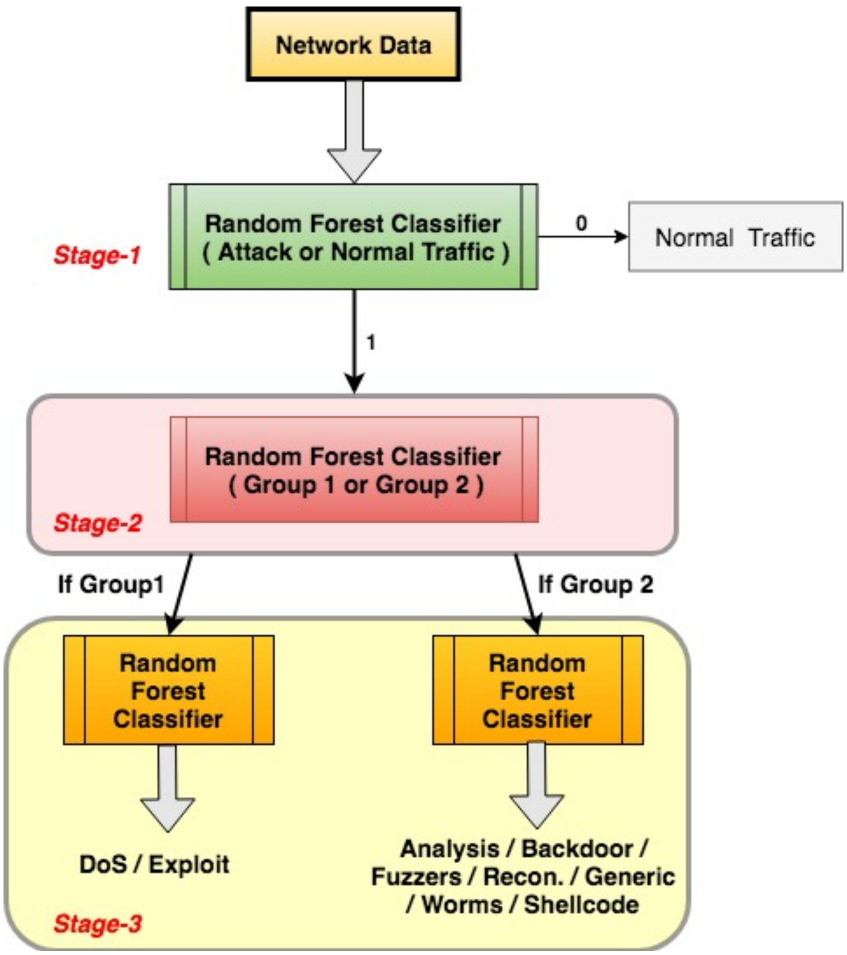

Figure 5 Hierarchical Multiclass Classifier

Hierarchical model is built up with stages. In each stage, there is a specific classifier for detection. Random Forest classifier is used for the detection process. General structure of the model is presented in Figure 5. Firstly, the network traffic (our data set) is monitored. If anomaly is detected, attack classification process occurs. Grouping is the key in hierarchical model. It is obvious that DoS and Exploit are detected easily with a more specific classifier after separated from other attacks.

For our machine learning model, Python3 and Scikit-Learn were used. Python is a popular programming language. Scikit-learn (Pedregosa et al., 2012) is a machine learning library running with Python. Code of Hierarchical Model is presented in Appendix-A.

3.3.1. Stages and Purposes

<u>Stage-1:</u>

A Random Forest Classifier is trained to detect whether there is an attack or not in Stage-1. The main aim in Stage-1 is attack detection. The only required label in instance is Attack or Normal for training.

<u>Stage-2:</u>

In Stage-2, Random Forest Classifier finds whether an attack belongs to Group 1 (DoS, Exploit) or Group 2 (Other Attack Classes). This grouping is important because if more problematic two classes (DoS, Exploit) are excluded from the rest of the traffic, the model can achieve better accuracy. Confusion matrix (Table 10) shows that DoS and Exploit classes intervene with each other in multiclass classification. This makes the overall accuracy of classifier worse. For training of the classifier, binary data (e.g. "1" denotes Group 1 and "2" denotes Group 2.) will be required.

<u>Stage-3:</u>

After Stage-2 classification, hierarchical model will classify exact attack classes. In Stage-3, there are two distinct Random Forest Classifiers. One for DoS/Exploit classes and the other for other attack classes.

<u>DoS / Exploit Detection</u>

Until this point, the classifier does not detect exact attack classes. If Stage-2 classifier classifies the instance into Group 1, DoS/Exploit Classifier will analyze this instance. This classifier works on only two classes and this will provide more successful detection.

<u>Analysis / Backdoor / Fuzzers / Recon. / Generic / Worms / Shellcode Detection</u>

If Stage-2 classifier classifies instance into Group 2, Analysis / Backdoor / Fuzzers / Recon. / Generic / Worms / Shellcode Classifier will analyze the instances. At this part of the stage, DoS and Exploit will be excluded and the classifier will work on less noisy data.

3.3.2. Data set Adjustment for Stages

UNSW-NB15 Training Set (175,341 instances) is used for the evaluation of Hierarchical Multiclass Classifier model. First of all, 60:40% ratio is used for training and testing of the model. After splitting, the training set has 105,204 instances. Also, 60% of UNSW-NB 15 data set was divided into three parts. All data sets were kept in *.csv format. All the instances in the subsets were randomly selected by WEKA.

To train four Random Forest Classifiers, four different data sets were used (Figure 6):

- **60% of UNSW-NB 15 Training set** was used in *Stage-1* for first classifier to detect attack or normal, (Totally, 33,612 instances are "Normal", and 71,592 instances are "Attack".) (Table 11)

- **Part 1** was used for *Stage-2* classification (Group-1 or Group-2). "Normal" label instances were excluded, and 23,574 instances were used. Group-1 (DoS, Exploit) have 8,940 instances and Group-2 (Other attack class) have 14,634 instances (Table 11),

- **Part 2** was used for DoS/Exploit classification in *Stage-3*. Except DoS/Exploit, other instances were excluded. 2,474 instances are DoS ("3" is label), 6,736 ("4" is label) instances are Exploit (Table 11),

- **Part 3** was used for other attack classes classification in *Stage-3*. DoS/Exploit were excluded from Part 3. 14,689 instances were used for training. "1" is Analysis, "2" is Backdoor, "5" is Fuzzers, "6" is Generic, "7" is Recon., "8" is Shellcode, "9" is Worms (Table 11).

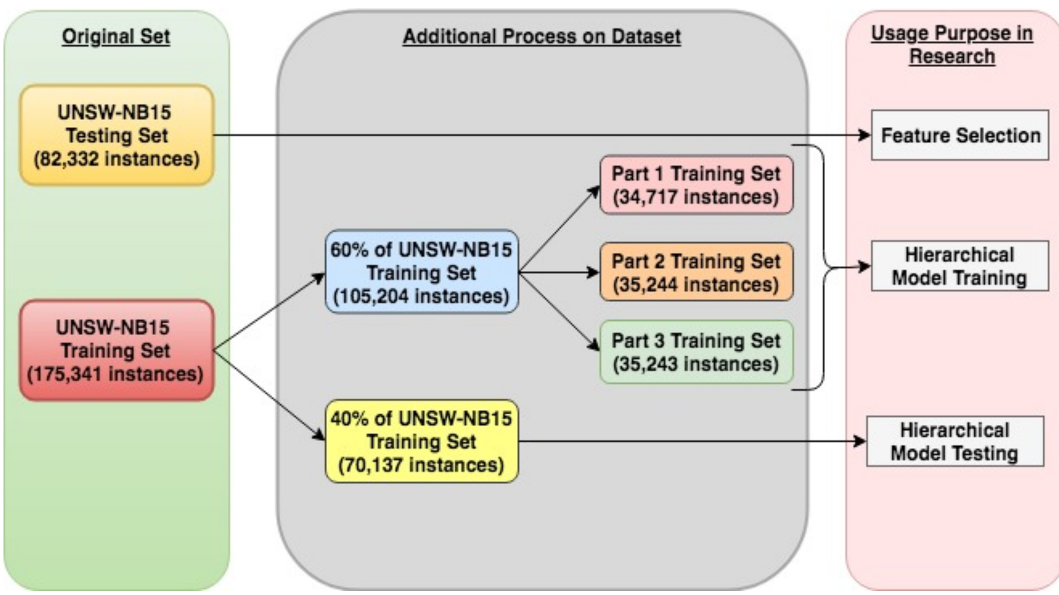

Figure 6 UNSW-NB15 Data sets and Adjustment on Data set

The parts of data set and instance sizes of parts are presented in Table 11. 40% of UNSW-NB15 Training set is used for testing purpose.

Table 11 Data Set Parts and Instance Sizes

	60% of UNSW-NB15 Training Set		**Part 1**		**Part 2**		**Part 3**		**40% of UNSW-NB15 Training Set**
Original Attack Class	**Attack or Not**		**Group 1 or Group 2**		**DoS or Exploit**		**Other Attack Class**		
	Size	**Label**	**Size**	**Label**	**Size**	**Label**	**Size**	**Label**	
Analysis	33612	0	11143	X	11115	X	11260	X	22388
Backdoor	1204		403	GR2	415	X	388	1	796
DoS	1047		323	GR2	359	X	385	2	699
Exploits	7430		2441	GR1	2474	3	2444	X	4834
Fuzzers	19921		6499	GR1	6736	4	6850	X	13472
Generic	10936	1	3665	GR2	3671	X	3538	5	7248
Recon.	23982		7889	GR2	8139	X	8011	6	16018
Shellcode	6299		2093	GR2	2081	X	2129	7	4192
Worms	694		236	GR2	231	X	219	8	439
Analysis	79		25	GR2	23	X	19	9	51
	105,204		**34,717**		**35,244**		**35,243**		**70,137**

CHAPTER 4

EVALUATION OF HIERARCHICAL MULTICLASS CLASSIFIER

4.1. Experiment Setup

For evaluation of the hierarchical multiclass classifier, a random forest classifier has also been implemented for comparison. 60% of UNSW-NB15 test set was used for training and 40% of UNSW-NB15 test set was used for testing. Training and testing times were also measured. Evaluation was conducted on a computer with Intel Core i7-3630QM CPU, 8 GB RAM and Microsoft Windows 10 operating system.

4.2. Results

Performance of the Random Forest classifier and the proposed Hierarchical Multiclass classifier are presented in Table 12. Overall accuracy of the multiclass classification with Random Forest classifier was measured as 78.64%. Training time was 3.2 s. and testing time was 1.62 s.

Table 12 Results for the proposed Hierarchical Multiclass Classifier and the Random Forest Classifier

	Number of Instances in Test Set	*Random Forest Classifier*				*Hierarchical Multiclass Classifier*			
		Correctly Detected Instances Number	Recall	Precision	F1-Score	Correctly Detected Instances Number	Recall	Precision	F1-Score
Normal	22,388	20,885	0.93	0.93	0.93	20,183	0.90	0.93	0.92
Analysis	796	186	0.23	0.06	0.09	139	0.17	0.36	0.24
Backdoor	699	186	0.27	0.05	0.09	71	0.10	0.33	0.16
DoS	4,834	1,887	0.39	0.35	0.37	3,053	0.63	0.33	0.44
Exploits	13,472	7,626	0.57	0.84	0.68	8,599	0.64	0.74	0.68
Fuzzers	7,248	5,207	0.72	0.77	0.75	5,120	0.71	0.70	0.70
Generic	16,018	15,738	0.98	1.00	0.99	15,707	0.98	0.99	0.99
Recon.	4,192	3,104	0.74	0.91	0.82	3,138	0.75	0.88	0.81
Shellcode	439	299	0.68	0.69	0.68	240	0.55	0.64	0.59
Worms	51	25	0.49	0.62	0.55	0	0.00	0.00	0.00
Overall Accuracy		0.78				0.80			

The overall accuracy of the Hierarchical Multiclass classifier was measured as 80.20%, which is greater than the overall accuracy of the Random Forest Classifier. DoS and Exploit classes have also greater detection rates in the former than the latter. Hierarchical model provides 94.80% accuracy in Stage-1. In Stage-1, the attack detection rate was 96.98%. Also, the normal traffic detection rate was 90.15% (See Table 13).

Table 13 Stage-1 Attack Detection Result

		Predicted Class	
		Normal	Attack
True Class	Normal	20,183	2,205
	Attack	1,439	46,310

The training time for the Hierarchical Multiclass classifier was measured as 5.64 s., and the testing time was measured as 158.34 s. During testing, 70,137 instances were evaluated. This means for evaluation of each instance roughly 2 msec. is required. The confusion matrix of the Hierarchical Multiclass Classifier is presented in Table 14.

Table 14 Confusion Matrix of Hierarchical Multiclass Classifier

		Predicted Class									
		Normal	**Analysis**	**Backdoor**	**DoS**	**Exploit**	**Fuzzers**	**Generic**	**Recon**	**Shellcode**	**Worms**
True Class	Normal	20,183	152	0	25	301	1691	0	24	12	0
	Analysis	54	139	12	443	139	4	0	5	0	0
	Backdoor	0	8	71	416	167	13	4	14	6	0
	DoS	12	26	33	3053	1497	98	15	67	33	0
	Exploit	68	46	79	4096	8599	245	47	252	39	1
	Fuzzers	1287	9	5	466	289	5120	8	33	31	0
	Generic	5	3	4	128	131	27	15707	4	9	0
	Recon	8	3	10	544	455	25	6	3138	3	0
	Shellcode	5	0	1	13	56	109	5	10	240	0
	Worms	0	0	0	1	45	4	1	0	0	0

To analyze the model's performance, the learning curves and the Receiver Operating Characteristic (ROC) curves were used. The training set (Table 10) was divided with an 80:20% ratio for test/validation set. To draw the ROC curve, predict_proba() method of "Random Forest Classifier" class in Scikit-learn was used. The predict_proba() method produces a probability array per class containing the probability that the given instance belongs to the given class. Root Mean Square Error (RMSE) was used for error measuring in the learning curves. To draw the learning curves, the classifiers were trained with 1,000 instances in each step and then tested with the validation set. Each time 1,000 instances were added up to the training set.

In Stage-1, there are 105,204 instances (60% of UNSW-NB15). About 84,000 instances were used for the model training and the rest for the validation. Figure 7 shows that the model is overfitting, because, the RMSE in the validation set is higher than in the training set and the gap between the two lines is large. Using more training instances, would not improve the model accuracy and close the large gap between the RMSE's. Some features might be irrelevant for this attack detection. For the ROC curve, the attack class was assumed as "Positive" classification. The ROC curve of Stage-1 is presented in Figure 8.

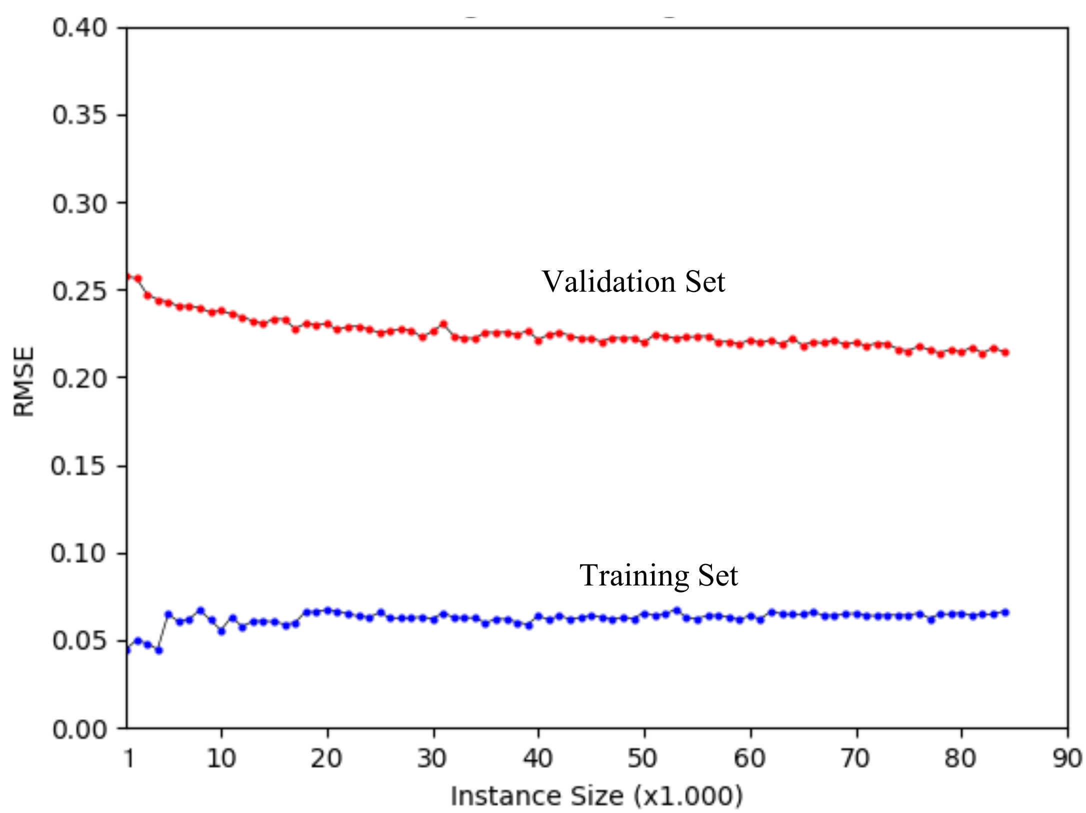

Figure 7 Stage-1 Learning Curve

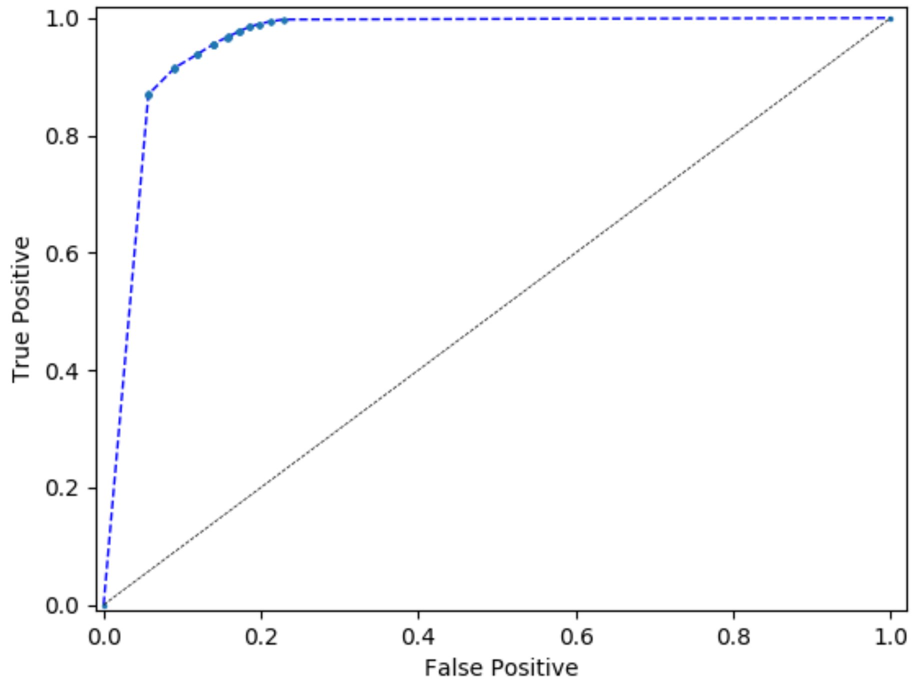

Figure 8 Stage-1 ROC Curve

In Stage-2, there are 23,574 instances (Part 1 data set). About 18,000 instances were used for the training and the rest for the validation. Figure 9 also shows overfitting. If more training data were collected, the model detection capability would improve slightly. For the ROC curve in Figure 10, Group 1 (DoS/Exploit) class was assumed as "Positive" classification.

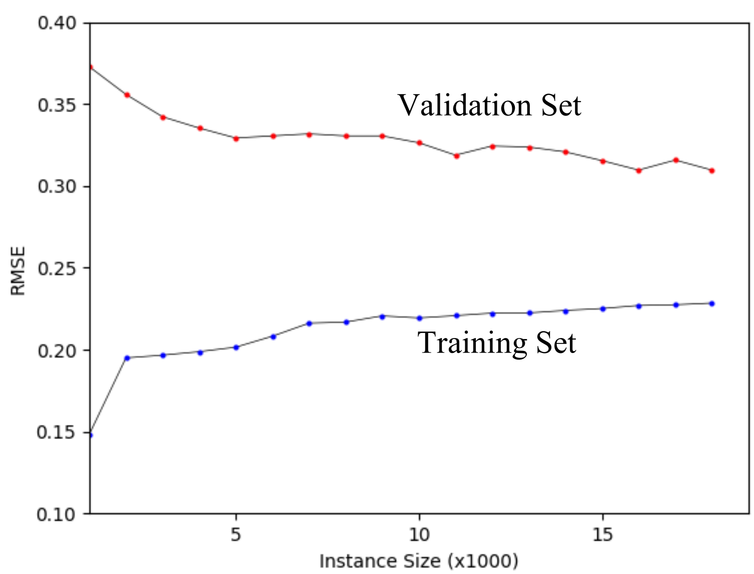

Figure 9 Stage-2 Learning Curve

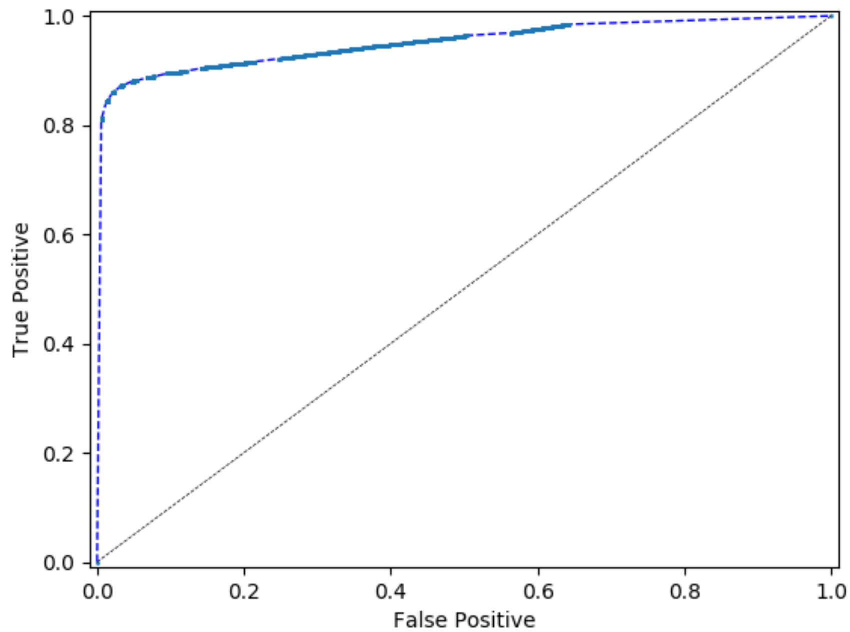

Figure 10 Stage-2 ROC Curve

In Stage-3, there are 9,210 instances (Part 2 data set). About 7,000 instances were used for the training and the rest for the validation. Figure 11 shows that, although a more specific binary classifier was used, the model is simply underfitting and more data would not improve the performance. For the ROC curve in Figure 12, the Exploit class was assumed as "Positive" classification. RMSE error is high and ROC curve is not adjacent to ideal point (upper-left point of figure).

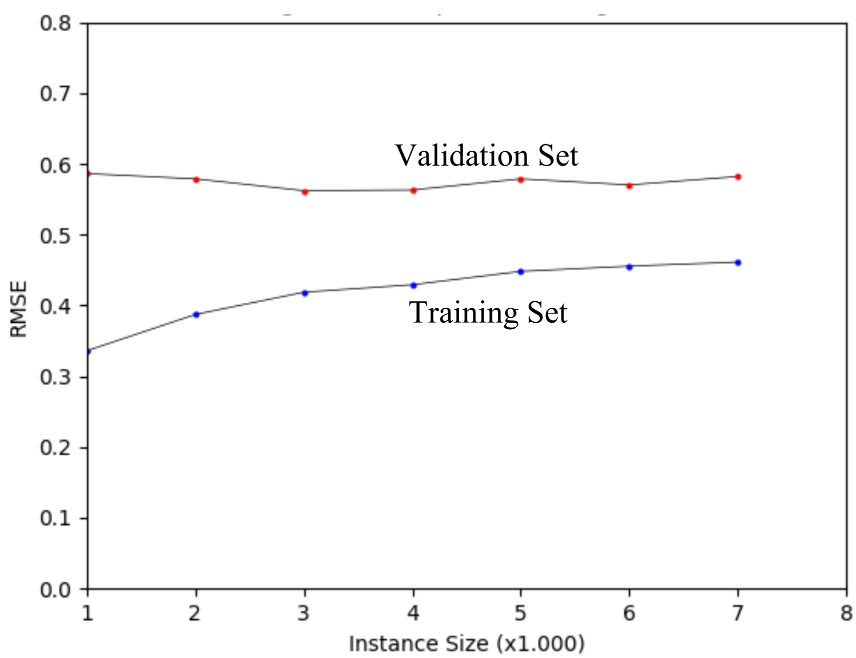

Figure 11 Stage-3 DoS/Exploit Learning Curve

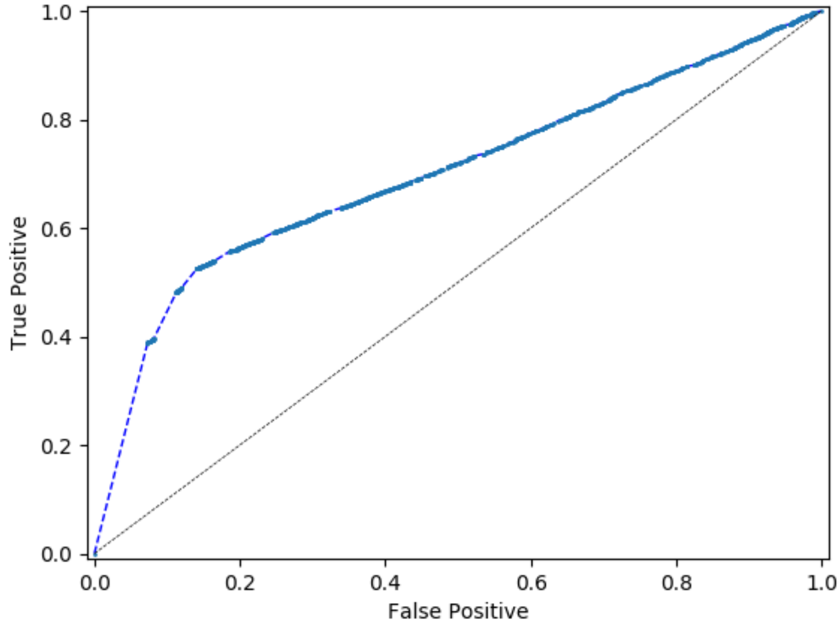

Figure 12 Stage-3 DoS/Exploit ROC Curve

CHAPTER 5

DISCUSSION

Hierarchical Multiclass Classifier model is a model which is built up with Random Forest Classifier. As Tavallaee et al. (2010) stated, the data set is an important part of the research. In many previous studies, the researchers obtained success rates higher than 90%. The results of some studies are presented in Table 15.

Table 15 Results of some Researches about Anomaly-based IDS model

Nu.	Researcher	Data set	Results
1	Shon & Moon	KDD-CUP 99	Attack Detection - 99.90%
2	Najafabadi et al.	Kyoto	Attack Detection - 98.54%
3	Pervez & Farid	NSL-KDD	Attack Detection - 82.37%
4	Khor et al.	KDD-CUP 99	Attack Classification - Normal:99%, DoS: 99%, Probe: %89, R2L: 91.5%, U2R:69%
5	Jiong Zhang et al.	NSL-KDD	Attack Detection - 98.98%
6	Ganapathy et al.	KDD-CUP 99	Attack Detection - 90.25%;
7	Gogoi et al.	KDD-CUP 99	Attack Classification - Normal: 94.68%, DoS: 98.54%, Probe: %93.5, R2L: 48.91%, U2R:97.14%

However, U2R and Probe attacks have limited instances in KDD-CUP 99 and NSL-KDD. DoS, R2L and Normal traffic instances outnumber these. Therefore, if higher detection rates on DoS and Normal traffic is achieved, the model looks as if more successful. In 2000's network behavior, it was understandable, but for today's network behavior, it is out of scope.

Khammassi and Krichen (2017) developed GA-LR model using UNSW-NB15 and KDD-CUP 99. 20 features were selected from UNSW-NB15 data set and while using only the selected 2000 instances, the model achieved 81.42% accuracy. It is important that in GA-LR model, the testing set consists of only 2000 instances. Hierarchical Multiclass Classifier and GA-LR model produced simply the same accuracy rate and low-detection classes (Analysis, Backdoor, Shellcode and Worms) were identical. The comparison of the two models is presented in Table 16.

Khammassi and Krichen (2017) and Moustafa and Slay (2017) described UNSW-NB15 as a complex data set. According to our findings, although there are enough instances for model training, some attack classes are hard to classify.

Table 16 Hierarchical Multiclass Classifier vs. GA-LR Model

	Hierarchical Multiclass Classifier			*GA-LR Model Result (Subset Z7 using C4.5 classifier)*		
	Recall	Precision	F1-Score	Recall	Precision	F1-Score
Normal	0.90	0.93	0.92	0.90	0.92	0.91
Analysis	0.17	0.36	0.24	0.10	0.44	0.16
Backdoor	0.10	0.33	0.16	0.69	0.51	0.58
DoS	0.63	0.33	0.44	0.04	0.36	0.07
Exploits	0.64	0.74	0.68	0.92	0.60	0.72
Fuzzers	0.71	0.70	0.70	0.69	0.70	0.69
Generic	0.98	0.99	0.99	0.97	0.99	0.97
Recon.	0.75	0.88	0.81	0.76	0.90	0.82
Shellcode	0.55	0.64	0.59	0.47	0.53	0.49
Worms	0.00	0.00	0.00	0.38	0.46	0.41
Model Accuracy	0.80			0.81		

Another important point in this study is feature selection. Wrapper feature selection method produced a more feasible feature set than the filter-based method. Also, the overall accuracy rate after the feature selection method was higher than using all the features. In many previous researches, researchers do not analyze the selected features' attributes. Iglesias and Zseby (2015) asserted that the traffic features (number of connections, SYN errors, reject errors, connection ratio of same service, connection ratio of different service etc.) are important features in KDD-CUP 99. Our selected feature set consists of basic features (duration, source/destination bytes, source/destination TTL, source/destination loss, source/destination packet size average) and connection features similar to KDD-CUP 99 traffic features. This shows that the basic features and connection features are more important feature groups. For application on real time network traffic, security analysts must focus on these features.

Janarthanan and Zargari (2017) find out that *service, sbytes, sttl, smean and ct_dst_sport_ltm* are significant features in UNSW-NB15 data set. Our wrapper-based feature selection model also selects these six features.

Although the Hierarchical Multiclass Classifier model provided a better detection rate on DoS and Exploit classes, it is not as good as expected. Apparently, the model suffers while detecting DoS, Exploit and Fuzzers. Feature extraction or some more special traffic feature may increase the model's overall accuracy.

Random Forest Classifier, which is an ensemble classifier and is based on decision trees, classifies 55,143 instances out of 70,137, correctly. On the other hand, the Hierarchical Multiclass Classifier classifies 56,250 instances out of 70,137, correctly. It may seem like a minor improvement in attack detection, but in a real network environment, every single detection is valuable and Exploit and Fuzzers are important attack types.

Hierarchical Multiclass Classifier was also tested with the Khammassi and Krichen's subset (20 features, Table 7) with the same attack distribution in Table 10. Results show that our 19 feature subset is better than Khammassi and Krichen's 20 feature subset (Table 17). Also, while using a different feature subset, Hierarchical Multiclass Classifier produces better overall model accuracy and DoS/Exploit detection rate than Random Forest Classifier.

Table 17 Results for the proposed Hierarchical Multiclass Classifier and the Random Forest Classifier using Khammassi and Krichen's Subset

	Number of Instances in Test Set	*Random Forest Classifier using Khammassi and Krichen's Subset*				*Hierarchical Multiclass Classifier using Khammassi and Krichen's Subset*			
		Correctly Detected Instances Number	Recall	Precision	F1-Score	Correctly Detected Instances Number	Recall	Precision	F1-Score
Normal	22,388	20,618	0.92	0.92	0.92	19,422	0.87	0.92	0.89
Analysis	796	173	0.22	0.06	0.09	51	0.06	0.14	0.09
Backdoor	699	149	0.21	0.05	0.07	75	0.11	0.16	0.13
DoS	4,834	1,772	0.37	0.32	0.34	2,781	0.58	0.32	0.41
Exploits	13,472	7,605	0.56	0.82	0.67	8,302	0.62	0.71	0.66
Fuzzers	7,248	5,078	0.70	0.73	0.72	4,835	0.67	0.60	0.63
Generic	16,018	15,714	0.98	1.00	0.99	15,704	0.98	0.99	0.98
Recon.	4,192	3,094	0.74	0.91	0.81	2,905	0.69	0.81	0.75
Shellcode	439	213	0.49	0.59	0.53	209	0.48	0.52	0.50
Worms	51	24	0.47	0.57	0.52	1	0.02	0.12	0.03
Overall Accuracy		0.77				0.78			

CHAPTER 6

CONCLUSION

This thesis presents an anomaly-based intrusion detection approach for multiclass classification with Random Forest Ensemble Classifier. Usage of a new and realistic data set is important in this thesis, since the objective is providing attack profiles to the current network traffic. So, UNSW-NB15 data set was selected. UNSW-NB15 data set is more complex and realistic than other old-fashioned popular cyberattack data sets such as KDD-CUP 99 and NSL-KDD.

UNSW-NB15 data set has 43 features for training/testing purposes. This leads to curse of dimensionality and overfitting. To eliminate this problem, wrapper feature selection method was applied. After the first multiclass examination with decision tree classifier, a better detection rate was obtained with multiclass classification with only 19 features rather than all features and filter-based selected features. Wrapper-based feature selection provides more feasible feature sets.

When selected features were analyzed carefully, most valuable properties of connections were found as TTL value, dropped packet size in both directions, transmit packet size in both directions, bits size in per second, number of connection attempt, and mean of data size in upper layer. Basic features and connection features are more important feature groups in the UNSW-NB15 data set.

Decision tree classifier was found as the best classifier for attack classification according to our model's overall accuracy in the feature selection phase. Random forest is an ensemble classifier that combines multiple decision trees. For this reason, Random Forest ensemble classifier was selected for our model, which is Hierarchical Multiclass Classifier.

Analysis, Backdoor, Shellcode and Worms have limited numbers of instances in testing and training sets. As a result of low instance sizes, these four classes have low detection rates. If there were more of these, the accuracy of these classes would also increase. However, DoS, Exploit and Fuzzers classes interfere as seen in the confusion matrix. To overcome this problem, a more specific Random Forest Classifier in a hierarchical stage was trained. In our point of view, we pay more attention to the Normal, DoS, Exploit, Fuzzers and Generic classes. As a result, better classification results were achieved by Hierarchical Multiclass Classifier except for Fuzzers.

Grouping and combining classes increased the accuracy of multiclass classification. Besides, the hierarchical model proves that using specific classifiers of different stages can increase both the class accuracy and average accuracy.

The overfitting observed in Stage-1 deteriorates the model's overall accuracy. The "Attack" detection rate is higher (96%) than the "Normal" class detection rate (90%) and if "Normal" class detection rate was the same as that of the "Attack" class, the model overall accuracy would increase.

When Hierarchical Multiclass Classifier was tested with another feature subset, it produced better overall accuracy and specific attack class detection with the Khammassi and Krichen's subset. Despite the improved detection rate, F1-Score improvement was not good enough in the model.

6.1. Future Work

As future work, a more robust classifier can be developed. Since recall and precision values for DoS, Exploit and Fuzzers are not as good as other classes such as Normal, Generic, Recon., etc., more can be done for dealing with these classes. Feature extraction can also be used to achieve better results. Also, it should be noted that some attacks are naturally hard to detect.

UNSW-NB15 data set has some drawbacks. Some attack classes have low instance sizes which make them hard to detect. Data augmentation can be a choice to increase the instance sizes. After data augmentation, a deep learning-based model can provide better detection rates for attack classification.

6.2. Limitations of the Thesis

In this thesis, collected data from a synthetic network was used for Hierarchical Multiclass Classifier. UNSW-NB15 is a public data set. All features related to the network traffic was kept offline as a csv file. Real time network traffic cannot be used as is. Also, in UNSW-NB15 data set, the normal to attack ratio is 45:55%. However, in real network traffic, the attack rate is lower than our test situation.

REFERENCES

Aburomman, A. A., & Ibne Reaz, M. Bin. (2017). A novel weighted support vector machines multiclass classifier based on differential evolution for intrusion detection systems. *Information Sciences*, *414*, 225–246. https://doi.org/10.1016/j.ins.2017.06.007

Akyol, A., Hacibeyoglu, M., & Karlik, B. (2016). Design of multilevel hybrid classifier with variant feature sets for intrusion detection system. *IEICE Transactions on Information and Systems*. https://doi.org/10.1587/transinf.2015EDP7357

Aldehim, G., & Wang, W. (2017). Determining appropriate approaches for using data in feature selection. *International Journal of Machine Learning and Cybernetics*, *8*(3), 915–928. https://doi.org/10.1007/s13042-015-0469-8

Amor, N., Benferhat, S., & Elouedi, Z. (2004). Naive bayes vs decision trees in intrusion detection systems. *ACM Symposium on Applied Computing*. https://doi.org/10.1145/967900.967989

Apiletti, D., Baralis, E., Cerquitelli, T., & D'Elia, V. (2009). Characterizing network traffic by means of the NetMine framework. *Computer Networks*. https://doi.org/10.1016/j.comnet.2008.12.011

Ben-Gal, I. (2005). Outlier Detection. In *Data Mining and Knowledge Discovery Handbook* (pp. 131–146). New York: Springer-Verlag. https://doi.org/10.1007/0-387-25465-X_7

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2011). Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset. *Expert Systems with Applications*, *38*(5), 5947–5957. https://doi.org/10.1016/j.eswa.2010.11.028

Boulaiche, A., & Adi, K. (2018). An auto-learning approach for network intrusion detection. *Telecommunication Systems*, *68*(2), 277–294. https://doi.org/10.1007/s11235-017-0395-z

Buczak, A. L., & Guven, E. (2016). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys and Tutorials*, *18*(2), 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

Cannady, J. D. (1998). Artificial neural networks for misuse detection. *Proceedings of the 21st National Information Systems Security Conference*. https://doi.org/citeulike-article-id:9827770

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

Cordella, L. Pietro, & Sansone, C. (2007). A multi-stage classification system for detecting intrusions in computer networks. *Pattern Analysis and Applications*, *10*(2), 83–100. https://doi.org/10.1007/s10044-006-0053-7

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Cunningham, P., & Delany, S. J. (2007). K -Nearest Neighbour Classifiers. *Multiple Classifier Systems*. https://doi.org/10.1016/S0031-3203(00)00099-6

Do, C. B., & Batzoglou, S. (2008). What is the expectation maximization algorithm? *Nature Biotechnology*. https://doi.org/10.1038/nbt1406

Eesa, A. S., Orman, Z., & Brifcani, A. M. A. (2015). A new feature selection model based on ID3 and bees algorithm for intrusion detection system. *Turkish Journal of Electrical Engineering and Computer Sciences*. https://doi.org/10.3906/elk-1302-53

ENISA. (2017). *ENISA Threat Landscape Report 2017*. Retrieved from europa.eu/publications/enisa-threat-landscape-report-2017

Ganapathy, S., Yogesh, P., & Kannan, A. (2012). Intelligent Agent-Based Intrusion Detection System Using Enhanced Multiclass SVM. *Computational Intelligence and Neuroscience*, *2012*, 1–10. https://doi.org/10.1155/2012/850259

Gogoi, P., Bhattacharyya, D. K., Borah, B., & Kalita, J. K. (2014). MLH-IDS: A Multi-Level Hybrid Intrusion Detection Method. *The Computer Journal*, *57*(4), 602–623. https://doi.org/10.1093/comjnl/bxt044

Gogoi, P., Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2012). Packet and Flow Based Network Intrusion Dataset. *Contemporary Computing*. https://doi.org/10.1007/978-3-642-32129-0_34

Guyon, I., & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research (JMLR)*. https://doi.org/10.1016/j.aca.2011.07.027

Hinton, G., & Salakhutdinov, R. (2009). Deep Boltzman machines. *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*. https://doi.org/10.1109/CVPRW.2009.5206577

Iglesias, F., & Zseby, T. (2015). Analysis of network traffic features for anomaly detection. *Machine Learning*, *101*(1–3), 59–84. https://doi.org/10.1007/s10994-014-5473-9

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, *31*(3), 264–323. https://doi.org/10.1145/331499.331504

Janarthanan, T., & Zargari, S. (2017). Feature selection in UNSW-NB15 and KDDCUP'99 datasets. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)* (pp. 1881–1886). IEEE. https://doi.org/10.1109/ISIE.2017.8001537

Jiong Zhang, Zulkernine, M., & Haque, A. (2008). Random-Forests-Based Network Intrusion Detection Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*. https://doi.org/10.1109/TSMCC.2008.923876

Khammassi, C., & Krichen, S. (2017). A GA-LR wrapper approach for feature selection in network intrusion detection. *Computers and Security*, *70*, 255–277. https://doi.org/10.1016/j.cose.2017.06.005

Khor, K.-C., Ting, C.-Y., & Amnuaisuk, S.-P. (2009). A Feature Selection Approach for Network Intrusion Detection. In *2009 International Conference on Information Management and Engineering* (pp. 133–137). https://doi.org/10.1109/ICIME.2009.68

Kruegel, C., Mutz, D., Robertson, W., & Valeur, F. (2003). Bayesian event classification for intrusion detection. In *Proceedings - Annual Computer Security Applications Conference, ACSAC*. https://doi.org/10.1109/CSAC.2003.1254306

Ladha, L., & Deepa, T. (2011). Feature Selection Methods And Algorithms. *International Journal on Computer Science and Engineering*, *3*(5), 1787–1797. Retrieved from http://journals.indexcopernicus.com/abstract.php?icid=945099

Li, Y., Wang, J. L., Tian, Z. H., Lu, T. B., & Young, C. (2009). Building lightweight intrusion detection system using wrapper-based feature selection mechanisms. *Computers and Security*, *28*(6), 466–475. https://doi.org/10.1016/j.cose.2009.01.001

McHugh, J. (2000). Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*. https://doi.org/10.1145/382912.382923

Mitchell, T. M. (1997). *Machine Learning*. *Annual Review Of Computer Science*. https://doi.org/10.1145/242224.242229

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE. https://doi.org/10.1109/MilCIS.2015.7348942

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, *25*(1–3), 18–31. https://doi.org/10.1080/19393555.2015.1125974

Moustafa, N., & Slay, J. (2017). A hybrid feature selection for network intrusion detection systems: Central points. *Australian Information Warfare and Security Conference, Symposia and Campus Events,* 5–13.

https://doi.org/10.4225/75/57a84d4fbefbb

Mukherjee, S., & Sharma, N. (2012). Intrusion Detection using Naive Bayes Classifier with Feature Reduction. *Procedia Technology*. https://doi.org/10.1016/j.protcy.2012.05.017

Najafabadi, M. M., Khoshgoftaar, T. M., & Seliya, N. (2016). Evaluating Feature Selection Methods for Network Intrusion Detection with Kyoto Data. *International Journal of Reliability, Quality and Safety Engineering*, *23*(01), 1650001. https://doi.org/10.1142/S0218539316500017

Nawir, M., Amir, A., Lynn, O. B., Yaakob, N., & Badlishah Ahmad, R. (2018). Performances of Machine Learning Algorithms for Binary Classification of Network Anomaly Detection System. *Journal of Physics: Conference Series*, *1018*, 012015. https://doi.org/10.1088/1742-6596/1018/1/012015

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012). How Many Trees in a Random Forest? In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 154–168). https://doi.org/10.1007/978-3-642-31537-4_13

Pal, S. K., & Mitra, S. (1992). Multilayer Perceptron, Fuzzy Sets, and Classification. *IEEE Transactions on Neural Networks*. https://doi.org/10.1109/72.159058

Papamartzivanos, D., Gómez Mármol, F., & Kambourakis, G. (2018). Dendron : Genetic trees driven rule induction for network intrusion detection systems. *Future Generation Computer Systems*, *79*, 558–574. https://doi.org/10.1016/j.future.2017.09.056

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2012). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. https://doi.org/10.1007/s13398-014-0173-7.2

Pervez, M. S., & Farid, D. M. (2014). Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. In *SKIMA 2014 - 8th International Conference on Software, Knowledge, Information Management and Applications*. https://doi.org/10.1109/SKIMA.2014.7083539

Quinlan, J. R. (1999). Simplifying decision trees. *International Journal of Human-Computer Studies*, *51*(2), 497–510. https://doi.org/10.1006/ijhc.1987.0321

Quinlan, J. R. (1992). *C4.5: Programs for Machine Learning*. *Morgan Kaufmann San Mateo California*. https://doi.org/10.1016/S0019-9958(62)90649-6

Roesch, M. (1999). Snort: Lightweight Intrusion Detection for Networks. *LISA '99: 13th Systems Administration Conference*. https://doi.org/http://portal.acm.org/citation.cfm?id=1039834.1039864

Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, *177*(18), 3799–3821. https://doi.org/10.1016/j.ins.2007.03.025

Sommer, R. (2016). The Bro Network Security Monitor. *Bro.Org*.

Suricata. (2017). Suricata Open Source IDS / IPS / NSM engine.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009*. https://doi.org/10.1109/CISDA.2009.5356528

Tavallaee, M., Stakhanova, N., & Ghorbani, A. A. (2010). Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *40*(5), 516–524. https://doi.org/10.1109/TSMCC.2010.2048428

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*. https://doi.org/10.1007/BF00992698

Zhong, C., Lin, T., Liu, P., Yen, J., & Chen, K. (2018). A cyber security data triage operation retrieval system. *Computers & Security*, *76*, 12–31. https://doi.org/10.1016/j.cose.2018.02.011

APPENDICES

APPENDIX A

Hierarchical Multiclass Classifier Model Codes

#Required library
```
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from time import time
```

#Starting time of Model Creation
```
start = time()
```

 # Attack or Not classifier training
```
data=np.loadtxt("1OR0.csv",delimiter=",")
x=data[:,0:19]
y=data[:,19]
clf0=RandomForestClassifier(class_weight="balanced", random_state=42)
clf0 = clf0.fit(x,y)
```

 # Attack or Not classifier training
```
data=np.loadtxt("part1.csv",delimiter=",")
x=data[:,0:19]
y=data[:,19]
clf1 = RandomForestClassifier(class_weight="balanced", random_state=42)
clf1 = clf1.fit(x,y)
```

 # DoS vs. Exploit classifier training
```
data=np.loadtxt("part2.csv",delimiter=",")
x=data[:,0:19]
y=data[:,19]
clf2=RandomForestClassifier(class_weight="balanced",random_state=42)
clf2 = clf2.fit(x,y)
```

 # DoS vs. Exploit classifier training
```
data=np.loadtxt("part3.csv",delimiter=",")
```

```
x=data[:,0:19]
y=data[:,19]
clf3 = RandomForestClassifier(class_weight="balanced", random_state=42)
clf3 = clf3.fit(x,y)
```

Model Creation Time

```
print("Training %.2f seconds:" % ((time() - start)))
```

Testing Phase

```
start = time()
data=np.loadtxt("UNSW_NB15_total_numeric_testing_WR_19FS-40percent.csv",delimiter=",")
x=data[:,0:19]
y=data[:,19]
result=np.zeros(len(y))
a=0
knt=False

for p in x:
   test = p[0:19]
   if (clf0.predict(test.reshape(1,-1)) == 0):
      result[a]=0
      a=a+1
      continue
   if clf1.predict(test.reshape(1,-1)) == 1:
      result[a]=clf2.predict(test.reshape(1,-1))
   if clf1.predict(test.reshape(1,-1)) == 2:
      result[a]=clf3.predict(test.reshape(1,-1))
   a=a+1

target_names = ['Normal', 'Analysis', 'Backdoor','DoS','Exploit','Fuzzers','Generic','Recon',
            'Shell','Worms']

acc = accuracy_score(y,result)
conf=confusion_matrix(y,result)


print(classification_report(y,result,target_names=target_names))

print (acc)
print (conf)

print ("Normal:"+str(float(conf[0,0])/np.count_nonzero(y==0)))
print ("Analysis:"+str(float(conf[1,1])/np.count_nonzero(y==1)))
print ("Backdoor:"+str(float(conf[2,2])/np.count_nonzero(y==2)))
print ("DoS:"+str(float(conf[3,3])/np.count_nonzero(y==3)))
print ("Exploit:"+str(float(conf[4,4])/np.count_nonzero(y==4)))
print ("Fuzzers:"+str(float(conf[5,5])/np.count_nonzero(y==5)))
print ("Generic:"+str(float(conf[6,6])/np.count_nonzero(y==6)))
```

```
print ("Recon:"+str(float(conf[7,7])/np.count_nonzero(y==7)))
print ("Shell:"+str(float(conf[8,8])/np.count_nonzero(y==8)))
print ("Worms:"+str(float(conf[9,9])/np.count_nonzero(y==9)))

print("Testing %.2f seconds:" % ((time() - start)))
```

APPENDIX B

RANDOM FOREST CLASSIFIER CODES

#Required library

```
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from time import time

start = time()
```

 # Testing Phase

```
data=np.loadtxt("UNSW_NB15_total_numeric_testing_WR_19FS-60percent.csv",delimiter=",")
x=data[:,0:19]
y=data[:,19]
clf0 = RandomForestClassifier(class_weight="balanced", random_state=42)
clf0 = clf0.fit(x,y)

print("Training %.2f seconds:" % ((time() - start)))
```

 # Testing Phase

```
start = time()
data=np.loadtxt("UNSW_NB15_total_numeric_testing_WR_19FS-40percent.csv",delimiter=",")
x=data[:,0:19]
y=data[:,19]

y_pred = clf0.predict(x)

target_names = ['Normal', 'Analysis', 'Backdoor','DoS','Exploit','Fuzzers','Generic','Recon',
            'Shell','Worms']

acc = accuracy_score(y,y_pred)
conf=confusion_matrix(y,y_pred)

print(classification_report(y,y_pred,target_names=target_names))

print (acc)
print (conf)

print ("Normal:"+str(float(conf[0,0])/np.count_nonzero(y==0)))
print ("Analysis:"+str(float(conf[1,1])/np.count_nonzero(y==1)))
print ("Backdoor:"+str(float(conf[2,2])/np.count_nonzero(y==2)))
print ("DoS:"+str(float(conf[3,3])/np.count_nonzero(y==3)))
```

```
print ("Exploit:"+str(float(conf[4,4])/np.count_nonzero(y==4)))
print ("Fuzzers:"+str(float(conf[5,5])/np.count_nonzero(y==5)))
print ("Generic:"+str(float(conf[6,6])/np.count_nonzero(y==6)))
print ("Recon:"+str(float(conf[7,7])/np.count_nonzero(y==7)))
print ("Shell:"+str(float(conf[8,8])/np.count_nonzero(y==8)))
print ("Worms:"+str(float(conf[9,9])/np.count_nonzero(y==9)))

print("Testing %.2f seconds:" % ((time() - start)))
```

APPENDIX C

SOME SELECTED GROUND TABLES FOR ATTACK TYPES

Analysis

Type	Prot.	Information
Port Scanner	ggp	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	ip	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	ipnip	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	st2	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	cbt	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	egp	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	argus	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	bbn-rcc	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	chaos	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	emcon	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	igp	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	nvp	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	pup	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)
Port Scanner	xnet	Analysis: IP Protocol Scan (https://strikecenter.bpointsys.com/bps/strikes/analysis/portscan/portscan_ip_proto.xml)

Backdoor

Prot.	Information
ospf	HP Performance Manager Tomcat Bypass (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2009_3548_HP_Performance Manager_Tomcat_Bypass.xml)
ospf	Vtiger CRM Unauthenticated Password Reset (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2014_2269_vtiger_crm_pass word_reset.xml)
sctp	HP OpenView Insight Server Backdoor Access (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2011_0276_HP_OpenView_ Backdoor.xml)
sctp	phpmyadmin 3.5.2.2 Backdoor Access and Code Execution (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2012_5159_phpmhyadmin_b ackdoor.xml)
sctp	Vtiger CRM Unauthenticated Password Reset (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2014_2269_vtiger_crm_pass word_reset.xml)
gre	Cisco Network Registrar Default Credentials Backdoor Access (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2011_2024_cisco_network_r egistrar_auth_bypass.xml)
gre	HP OpenView Insight Server Backdoor Access (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2011_0276_HP_OpenView_ Backdoor.xml)
gre	HP Performance Manager Tomcat Bypass (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2009_3548_HP_Performance Manager_Tomcat_Bypass.xml)
tcp	Backdoor: Windows XP CMD.EXE Reverse Shell (https://strikecenter.bpointsys.com/bps/strikes/backdoors/windows_cmd_shell_reverse_xp.x ml)
ospf	Cisco Network Registrar Default Credentials Backdoor Access (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2011_2024_cisco_network_r egistrar_auth_bypass.xml)
ospf	Cisco Network Registrar Default Credentials Backdoor Access (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2011_2024_cisco_network_r egistrar_auth_bypass.xml)
ospf	phpmyadmin 3.5.2.2 Backdoor Access and Code Execution (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2012_5159_phpmhyadmin_b ackdoor.xml)
ospf	phpmyadmin 3.5.2.2 Backdoor Access and Code Execution (https://strikecenter.bpointsys.com/bps/strikes/backdoors/cve_2012_5159_phpmhyadmin_b ackdoor.xml)
tcp	Backdoor: Girlfriend v1.35 Client Connection (https://strikecenter.bpointsys.com/bps/strikes/backdoors/trojan_girlfriend_02.xml)

DoS

Type	Prot.	Information
Miscellaneous	tcp	Cisco DCP2100 SADownStartingFrequency Denial of Service (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/cisco_dcp2100_denial_of_service.xml)
Miscellaneous	tcp	Cisco DCP2100 SADownStartingFrequency Denial of Service (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/cisco_dcp2100_denial_of_service.xml)
Browser	tcp	Mozilla Firefox OBJECT Tag Crafted Style Null Dereference (https://strikecenter.bpointsys.com/bps/strikes/denial/browser/firefox_display_moz_deck_null_deref.xml)
Miscellaneous	tcp	Tri PLC Nano 10 PLC Denial of Service (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/cve_2013_2784_tri_PLC_nano10_dos.xml)
SNMP	udp	Cisco SNMP Trap Service GET Request DoS (162) (https://strikecenter.bpointsys.com/bps/strikes/denial/snmp/cisco_snmptrap_snmp_01.xml)
Miscellaneous	tcp	Apple OS X QuickDraw GetSrcBits32ARGB Memory Corruption Denial of Service (POP3) (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/osx_quickdraw_getsrcbits32argb_pop3_download.xml)
Browser	tcp	Mozilla Firefox XUL menupopup.menu Null Pointer Dereference (https://strikecenter.bpointsys.com/bps/strikes/denial/browser/firefox_xul_null_menu.xml)
Browser	tcp	Apple Quicktime for Windows QTPlugin.ocx ActiveX Control SetBgColor Denial of Service (https://strikecenter.bpointsys.com/bps/strikes/denial/browser/apple_quicktime_activex_setbgcolor.xml)
FTP	tcp	Microsoft IIS FTP Server NLST Infinite Recursion DoS (https://strikecenter.bpointsys.com/bps/strikes/denial/ftp/ms09_053_iis_ftpd_nlst_infinite_recursion_dos.xml)
FTP	tcp	Microsoft IIS FTP Server NLST Infinite Recursion DoS (https://strikecenter.bpointsys.com/bps/strikes/denial/ftp/ms09_053_iis_ftpd_nlst_infinite_recursion_dos.xml)
FTP	tcp	Microsoft IIS FTP Server NLST Infinite Recursion DoS (https://strikecenter.bpointsys.com/bps/strikes/denial/ftp/ms09_053_iis_ftpd_nlst_infinite_recursion_dos.xml)
Miscellaneous	tcp	Wireshark Profinet DCP Dissector Name of Station Set Request Format String Vulnerability (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/wireshark_profinet_dcp_ident_request_dos.xml)
Miscellaneous	tcp	Sybase Open Server Function Pointer (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/sybase_open_server_function_pointer.xml)

Exploit

Type	Prot.	Information
Unix 'r' Service	udp	Solaris rwalld Format String Vulnerability (https://strikecenter.bpointsys.com/bps/strikes/exploits/rservices/solaris_rwall_format_string.xml)
Browser	tcp	Windows Metafile (WMF) SetAbortProc() Code Execution [009] (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/wmf_009.xml)
Miscellaneous Batch	tcp	HP Data Protector Backup (https://strikecenter.bpointsys.com/bps/strikes/exploits/misc/cve_2011_1729.xml)
Cisco IOS	tcp	Cisco IOS HTTP Authentication Bypass Level 64 (https://strikecenter.bpointsys.com/bps/strikes/exploits/ios/cisco_auth_bypass_level_64.xml)
Browser	tcp	Microsoft Internet Explorer Frameset Memory Corruption (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/ms06_042_html_frameset_memory_corruption.xml)
Browser	tcp	Microsoft Internet Explorer Frameset Memory Corruption (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/ms06_042_html_frameset_memory_corruption.xml)
SCADA	tcp	Mitsubishi EZPcAut260.dll ActiveX Control ESOpen Buffer Overflow (https://strikecenter.bpointsys.com/bps/strikes/exploits/scada/cve_2014_1641_Mitsubishi_EZPcAut260_ActiveX_Control_ESOpen_bo.xml)
Browser	tcp	Microsoft Internet Explorer Layouts Handling Memory Corruption (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/ms11_018_layouts_handling_memory_corruption.xml)
Browser	tcp	Microsoft Internet Explorer ActiveX Arbitrary Command Execution (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/ms03_040_malicious_popups.xml)
Browser	tcp	Microsoft Internet Explorer ActiveX Arbitrary Command Execution (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/ms03_040_malicious_popups.xml)
Miscellaneous Batch	tcp	BigAnt Server Arbitrary File Upload (https://strikecenter.bpointsys.com/bps/strikes/exploits/misc/cve_2012_6274.xml)
SCADA	tcp	Advantech WebAccess SCADA webvact NodeName2 Buffer overflow (https://strikecenter.bpointsys.com/bps/strikes/exploits/scada/cve_2014_0766_advantech_webaccess_scada_webvact_nodename2_bo.xml)
Miscellaneous Batch	tcp	HP SiteScope Default User information (https://strikecenter.bpointsys.com/bps/strikes/exploits/misc/osvdb_74865_hp_sitescope_default_credential.xml)

Fuzzers

Type	Prot.	Information
OSPF	ospf	Fuzzer: OSPF Database Description Packet: Basic (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/ospf/dbd_basic.xml)
OSPF	ospf	Fuzzer: OSPF Hello Packet: Invalid Length, Long Payload (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/ospf/hello_invalid_length_long_payload.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)
HTTP	tcp	Fuzzer: HTTP GET Request Invalid URI (https://strikecenter.bpointsys.com/bps/strikes/fuzzers/http/get_invaliduri.xml)

Generic

Type	Prot.	Information
IXIA	tcp	Alt-N_MDaemon_WorldClient_Service_Memory_Corruption_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/alt-n_mdaemon_worldclient_service_memory_corruption_attack.xml)
SIP	udp	RFC 4475: SIP Torture Tests: Missing Required Header Fields (CSeq) (https://strikecenter.bpointsys.com/bps/strikes/generic/sip/rfc_4475_3_3_1_missing_header_field_cseq.xml)
IXIA	tcp	Apple_QuickTime_udta_Atom_Buffer_Overflow_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/apple_quicktime_udta_atom_buffer_overflow_attack.xml)
IXIA	tcp	Adobe_Shockwave_Player_DIR_Files_PAMI_Chunk_Code_Execution_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/adobe_shockwave_player_dir_files_pami_chunk_code_execution_attack.xml)
SIP	udp	RFC 4475: SIP Torture Tests: Unknown Protocol Version (https://strikecenter.bpointsys.com/bps/strikes/generic/sip/rfc_4475_3_1_2_16_unknown_protocol_version.xml)
SMTP	tcp	SMTP: Executable File Attachment in Archive (SCR in TAR.GZ) (https://strikecenter.bpointsys.com/bps/strikes/generic/smtp/attachment_tar_gz_scr.xml)
IXIA	tcp	Microsoft_Excel_File_Importing_Code_Execution_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/microsoft_excel_file_importing_code_execution_attack.xml)
SMTP	tcp	SMTP: Executable File Attachment in Archive (BAT in RAR) (https://strikecenter.bpointsys.com/bps/strikes/generic/smtp/attachment_rar_bat.xml)
TFTP	udp	TFTP GET Request - Long File Name (512 bytes) (Octet) (https://strikecenter.bpointsys.com/bps/strikes/generic/tftp/tftp_octet_long_get_512.xml)
IXIA	tcp	HP_WEB_JETADMIN_issue2_GET (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/hp_web_jetadmin_issue2_get.xml)
IXIA	tcp	Adobe_Reader_and_Acrobat_util_printf_Stack_Buffer_Overflow_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/adobe_reader_and_acrobat_util_printf_stack_buffer_overflow_attack.xml)
IXIA	tcp	Apple QuickTime Color Table ID Heap Corruption attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/apple_quicktime_color_table_id_heap_corruption_attack.xml)
IXIA	tcp	Apple_QuickTime_STSD_Atoms_Handling_Heap_Overflow_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/apple_quicktime_stsd_atoms_handling_heap_overflow_attack.xml)
IXIA	tcp	Cisco_WebEx_Player__WRF_Stack_Buffer_Overflow_attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/cisco_webex_player__wrf_stack_buffer_overflow_attack.xml)
IXIA	tcp	InterNetNews Control Message Handling Buffer Overflow attack (https://strikecenter.bpointsys.com/bps/strikes/generic/ixia/internetnews_control_message_handling_buffer_overflow_attack.xml)
SIP	udp	RFC 4475: SIP Torture Tests: Invalid Time Zone in Date Header Field (Negative Offset) (https://strikecenter.bpointsys.com/bps/strikes/generic/sip/rfc_4475_3_1_2_12_invalid_timezone_negative_offset.xml)

Reconnaissance

Type	Prot.	Information
HTTP	tcp	Domino Web Server Database Access: /doladmin.nsf (https://strikecenter.bpointsys.com/bps/strikes/recon/http/domino/access_domino_doladmin_nsf.xml)
SunRPC Portmapper (UDP) TCP Service	udp	SunRPC UDP Portmapper GETPORT Request (iostatv2/tcp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_udp/service_tcp/iostat_v2_tcp.xml)
SunRPC Portmapper (TCP) UDP Service	tcp	SunRPC TCP Portmapper GETPORT Request (etherifv3/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_tcp/service_udp/etherif_v3_udp.xml)
SunRPC Portmapper (TCP) TCP Service	tcp	SunRPC TCP Portmapper GETPORT Request (schedv3/tcp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_tcp/service_tcp/sched_v3_tcp.xml)
SunRPC Portmapper (TCP) UDP Service	tcp	SunRPC TCP Portmapper GETPORT Request (x25_inrv3/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_tcp/service_udp/x25_inr_v3_udp.xml)
SunRPC Portmapper (UDP) UDP Service	udp	SunRPC UDP Portmapper GETPORT Request (statusv3/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_udp/service_udp/status_v3_udp.xml)
SunRPC Portmapper (UDP) UDP Service	udp	SunRPC UDP Portmapper GETPORT Request (kerbdv3/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_udp/service_udp/kerbd_v3_udp.xml)
SunRPC Portmapper (TCP) TCP Service	tcp	SunRPC TCP Portmapper GETPORT Request (swu_svrv2/tcp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_tcp/service_tcp/swu_svr_v2_tcp.xml)
SunRPC Portmapper (UDP) UDP Service	udp	SunRPC UDP Portmapper GETPORT Request (rpcbindv3/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_udp/service_udp/rpcbind_v3_udp.xml)
SunRPC Portmapper (UDP) UDP Service	udp	SunRPC UDP Portmapper GETPORT Request (loggerv1/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_udp/service_udp/logger_v1_udp.xml)
SunRPC Portmapper (TCP) UDP Service	tcp	SunRPC TCP Portmapper GETPORT Request (remote_dbxv3/udp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_tcp/service_udp/remote_dbx_v3_udp.xml)
SunRPC Portmapper (UDP) TCP Service	udp	SunRPC UDP Portmapper GETPORT Request (iostat2v3/tcp) (https://strikecenter.bpointsys.com/bps/strikes/recon/sunrpc/portmap_udp/service_tcp/iostat2_v3_tcp.xml)

Shellcode

Type	Prot.	Information
Multiple OS	tcp	Shellcode: Multi-OS Shell (solaris/linux/irix) - dymitri (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/multi/shell_multi_dymitri_3_tcp.xml)
Mac OS X	udp	Shellcode: Mac OS X PPC Reverse Shell - metasploit (UDP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/osx/reverse_ppc_metasploit_udp.xml)
Linux	tcp	Shellcode: Linux SPARC Reverse Connect Shell - metasploit (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/linux/reverse_sparc_metasploit_tcp.xml)
Solaris	tcp	Shellcode: Solaris SPARC Reverse Connect Shell - metasploit (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/solaris/reverse_sparc_metasploit_tcp.xml)
Windows	tcp	Shellcode: Windows x86 Execute Command - metasploit (TCP) Variant 1 (https://strikecenter.bpointsys.com/bps/strikes/shellcode/win32/exec_x86_metasploit_1_tcp.xml)
Linux	tcp	Shellcode: Linux x86 Reverse Connect TCP Shell - metasploit (https://strikecenter.bpointsys.com/bps/strikes/shellcode/linux/reverse_x86_udp_metasploit_tcp.xml)
OpenBSD	udp	Shellcode: OpenBSD x86 Bind Shell - noir (UDP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/openbsd/bind_x86_noir_udp.xml)
Mac OS X	udp	Shellcode: Mac OS X PPC Reverse Stage - metasploit (UDP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/osx/reverse_ppc_stage_metasploit_udp.xml)
BSD	tcp	Shellcode: BSD x86 Bind Shell (random) - MayheM (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/bsd/bind_x86_random_tcp.xml)
BSD	tcp	Shellcode: BSD x86 FindRecv Stage - metasploit (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/bsd/findrecv_x86_stage_metasploit_tcp.xml)
SCO Unix	udp	Shellcode: SCO OpenServer x86 Shell - minervini (UDP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/sco/shell_x86_minervini_udp.xml)
Linux	udp	Shellcode: Linux x86 Bind Shell - metasploit (UDP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/linux/bind_x86_metasploit_udp.xml)
Windows	udp	Shellcode: Windows x86 Download Execute - metasploit (UDP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/win32/downloadexec_x86_metasploit_udp.xml)
Windows	udp	Shellcode: Windows x86 Add User - metasploit (UDP) Variant 1 (https://strikecenter.bpointsys.com/bps/strikes/shellcode/win32/adduser_x86_metasploit_1_udp.xml)
Windows	tcp	Shellcode: Windows x86 Reverse Stage - metasploit (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/win32/reverse_x86_stage_metasploit_tcp.xml)
BSD	tcp	Shellcode: BSD x86 chroot() - s0t4ipv6 (TCP) (https://strikecenter.bpointsys.com/bps/strikes/shellcode/bsd/chroot_x86_s0t4ipv6_tcp.xml)

Worms

Prot.	Information
tcp	Lupper.A XML-RPC Propogation Request Variant 8 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_08.xml)
tcp	Trojan.MDropper Word Document (http) Variant 2 (https://strikecenter.bpointsys.com/bps/strikes/worms/mdropper_http_02.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 6 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_06.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 7 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_07.xml)
tcp	Linux Lupper A Work Propogation via HTTP (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_http_download.xml)
tcp	Linux Lupper A Variant 1 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_awstats_0.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 3 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_03.xml)
udp	ISS Realsecure/BlackICE Witty Worm (https://strikecenter.bpointsys.com/bps/strikes/worms/witty.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 10 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_10.xml)
tcp	Code Red Worm (https://strikecenter.bpointsys.com/bps/strikes/worms/codered_a.xml)
tcp	Linux Lupper A Variant 2 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_awstats_1.xml)
tcp	Trojan.MDropper Word Document (http) Variant 1 (https://strikecenter.bpointsys.com/bps/strikes/worms/mdropper_http_01.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 13 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_13.xml)
udp	Nimda Worm TFTP Request Admin.dll (https://strikecenter.bpointsys.com/bps/strikes/worms/nimda_tftp_req.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 1 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_01.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 9 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_09.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 2 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_02.xml)
udp	Microsoft SQL Server Slammer/Saphire Worm (https://strikecenter.bpointsys.com/bps/strikes/worms/slammer.xml)
tcp	X97EmbedAn Excel Document (http) (https://strikecenter.bpointsys.com/bps/strikes/worms/x97embedan_http_01.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 11 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_11.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 12 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_12.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 14 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_14.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 4 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_04.xml)
tcp	Lupper.A XML-RPC Propogation Request Variant 5 (https://strikecenter.bpointsys.com/bps/strikes/worms/linux_lupper_a_xmlrpc_05.xml)

TEZ İZİN FORMU / THESIS PERMISSION FORM

ENSTİTÜ / INSTITUTE

Fen Bilimleri Enstitüsü / Graduate School of Natural and Applied Sciences ☐

Sosyal Bilimler Enstitüsü / Graduate School of Social Sciences ☐

Uygulamalı Matematik Enstitüsü / Graduate School of Applied Mathematics ☐

Enformatik Enstitüsü / Graduate School of Informatics X

Deniz Bilimleri Enstitüsü / Graduate School of Marine Sciences ☐

YAZARIN / AUTHOR

Soyadı / Surname : Sarıkaya
Adı / Name : Alper
Bölümü / Department : Bilişim Sistemleri

TEZİN ADI / TITLE OF THE THESIS (**İngilizce** / English) :

Anomaly-Based Cyber Intrusion Detection System with Ensemble Classifier

TEZİN TÜRÜ / DEGREE**:** **Yüksek Lisans** / Master X **Doktora** / PhD ☐

1. **Tezin tamamı dünya çapında erişime açılacaktır. /** Release the entire work immediately for access worldwide. X
2. **Tez iki yıl süreyle erişime kapalı olacaktır.** / Secure the entire work for patent and/or proprietary purposes for a period of **two year. *** ☐
3. **Tez altı ay süreyle erişime kapalı olacaktır.** / Secure the entire work for period of **six months**. ***** ☐

Yazarın imzası / Signature .... **Tarih** / Date : 6 Eylül 2018