2017

Representing Statistical Network-Based Anomaly Detection by Using Trust

Younghun Chae
University of Rhode Island, iplusu@my.uri.edu

Follow this and additional works at: https://digitalcommons.uri.edu/oa_diss

Recommended Citation

Chae, Younghun, "Representing Statistical Network-Based Anomaly Detection by Using Trust" (2017).
Open Access Dissertations. Paper 557.
https://digitalcommons.uri.edu/oa_diss/557

This Dissertation is brought to you for free and open access by DigitalCommons@URI. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of DigitalCommons@URI. For more information, please contact digitalcommons-group@uri.edu.

REPRESENTING STASTICAL NETWORK-BASED

ANOMALY DETECTION BY USING TRUST

BY

YOUNGHUN CHAE

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

UNIVERSITY OF RHODE ISLAND

2017

DOCTOR OF PHILOSOPHY DISSERTATION
OF
YOUNGHUN CHAE

APPROVED:

Dissertation Committee:

Major Professor	Lisa DiPippo
	Natallia Katenka
	Victor Fay-Wolfe
	Yan Sun

Nasser H. Zawia
DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND
2017

ABSTRACT

A large number of systems that rely on the Internet pervasively launched into our daily lives during the last decades. The openness and scalability of the Internet have provided a flexible platform for private customers, professionals, and academics with non-commercial or commercial interests. Therefore, there has been growing interest in network security. Network or computer systems may employ anti-virus, firewall, intrusion prevention system, or intrusion detection system to maintain the safe environment. Among the systems, the intrusion detection system plays a very important role in minimizing the damage caused by different attacks. The system attempts to learn the features of behaviors and events of system and/or users over a period time to build a profile of normal behaviors, then looks for malicious behaviors that deviate from the normal profile. Thus, it is able to detect attacks even when detailed information of the attack does not exist.

A wide variety of techniques were proposed to approach the system including machine learning, data mining, hidden Markov models, and statistical analysis. This research mainly considers the statistical analysis-based method due to the simplicity and effectiveness. However, the method has a few disadvantages. First, the results are not intuitive because it requires the related statistical background. Second, it is difficult to integrate the results of different statistical analyses due to the different ranges. Last, the methods provide only two states; normal or malicious behaviors.

This research proposes a trust management scheme as a solution to these problems. A trust management scheme is able to scale the results of statistical analyses

between 0 (unreliable) and 1 (reliable), so the results would not only be intuitive but also be able to integrate the various results into one. Moreover, the trust values can provide more than normal and malicious behaviors, such as a-little-suspicious, suspicious, or highly-suspicious states.

We will show how well the trust management scheme can represent the results of various statistical analyses, how the system can find the appropriate thresholds for classifying the malicious behaviors in this dissertation. The proposed trust management scheme will be applied to two real-world datasets, and we will discuss the experiments results.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude and appreciation to my advisor Professor Lisa DiPippo and Professor Natallia Katenka for their continuous guidance, generous support, and inspiration of research. From my first moments at University of Rhode Island, Professor Lisa DiPippo allowed me to join her research project and her support encouraged my academic endeavors as well as provided financial support to allow me to continue my study. Professor Natallia Katenka inspired me for the new research and provided useful resources. She always gave me deep and thorough research ideas for this thesis. I have learned a lot about the basic and advanced knowledge for this research and other fields and she has advised me a lot of things for my improvements. Without their help, it would be impossible to complete my degree and this dissertation.

Last and not least, I would like to thank my wife, Miri Chung, for her understanding and patience to make this dissertation possible. She encouraged me whenever I had problems and took care of me in love. I also wish to thank my parents for their encouragement and supports.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGMENTS	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER 1	1
INTRODUCTION.....	1
1.1 Problem Statement	1
1.2 Research goals.....	4
1.3 Data description	5
1.4 Dissertation Structure.....	7
CHAPTER 2	9
BACKGROUND AND RELATED WORK	9
2.1 Intrusion Detection System (IDS).....	10
2.1.1 Misuse Detection.....	11
2.1.2 Anomaly Detection	11
2.2 Statistical analysis-based Anomaly Detection System (ADS).....	13
2.2.1 Packet-based statistical analysis.....	14
2.2.2 Flow-based statistical analysis	15
2.2.3 Problems.....	17

2.3	Network graphs	19
2.3.1	Bipartite network graph.....	20
2.3.2	Degree	21
2.3.3	Bipartite network projection	22
2.3.4	Clustering coefficient.....	23
2.3.5	Betweenness centrality.....	25
2.4	Trust	25
2.4.1	Trust management schemes	26
2.4.2	Beta Reputation.....	28
2.4.3	Predictability Trust.....	29
2.5	Outlier-based adaptive threshold.....	36
2.6	Intrusion overview	42
2.6.1	Scanning attacks.....	43
2.6.2	Deception of identity.....	47
2.7	Summary	48
CHAPTER 3		49
STATISTICAL NETWORK ANALYSIS BY USING TRUST		49
3.1	Node characterization.....	50
3.2	Modified Beta Reputation.....	52
3.3	Predictability Trust.....	57

3.4 Overall trust.....	63
3.5 Summary	64
CHAPTER 4	66
RESULTS AND ANALYSIS.....	66
4.1 Graph-based representation of network flows	66
4.2 Relationship between statistical analysis and trust management scheme	70
4.1.1 Experiment design.....	71
4.1.2 Experimental results.....	75
4.3 Static and adaptive thresholds.....	78
4.3.1 Experiment design.....	78
4.3.2 Experimental results.....	88
4.3.3 Static threshold for clustering coefficient	96
4.4 Detecting attacks using trust management scheme.....	97
4.4.1 Experiment design.....	99
4.4.2 Experimental results.....	110
4.5 The trust management scheme in special cases	153
4.5.1 Evaluation of one temporary error	154
4.5.2 Evaluation of on-off attacks	156
4.5.3 Long-term redemption	158
4.6 URI summarized results.....	160

4.6.1	Host degrees	161
4.6.2	Port degrees	164
4.6.3	Betweenness centrality	168
4.7	Summary	171
CHAPTER 5		173
CONCLUSION AND FUTURE DIRECTIONS		173
APPENDIX I		175
A.	Pseudocode for the predictability trust.....	175
B.	Pseudocodes for the Everyday malicious pattern-based ROC curves	177
C.	Pseudocodes for the malicious Average degree pattern-based ROC curves...	178
D.	Pseudocodes for the At Least Once malicious pattern-based ROC curves.....	179
E.	Pseudocodes for the trust-based ROC curves	180
F.	Pseudocodes for the outlier-based adaptive threshold	182
G.	NPB and MPB for the predictability trust.....	183
APPENDIX II		184
A.	Comparisons of the amount of dataset being analyzed in the European dataset 184	
B.	Comparisons of the amount of dataset being analyzed in the URI dataset.....	184
C.	The intersections between the adaptive and each static threshold in the host- degree analysis ROC curves.....	187

D. The intersections between the adaptive and each static threshold in the port- degree analysis ROC curves.....	188
BIBLIOGRAPHY	189

LIST OF TABLES

TABLE	PAGE
Table 1. Clustering coefficient of each node	24
Table 2. The patterns of on-off attacks	59
Table 3. Average host-degree on each day	79
Table 4. TPR of the host-degree based analysis on each degree threshold.....	80
Table 5. FPR of the host-degree based analysis on each degree threshold.....	81
Table 6. Average port-degree on each day.....	82
Table 7. TPR of the port-degree based analysis on each degree threshold.....	83
Table 8. FPR of the port-degree based analysis on each degree threshold.....	84
Table 9. The highest TPR of each threshold with the corresponding FPR in the Host-degree analysis on Day 1.....	90
Table 10. The highest TPR of each threshold with the corresponding FPR in the Host-degree analysis on Day 14.....	91
Table 11. The highest TPR of each threshold with the corresponding FPR in the Port-degree analysis on Day 1.....	94
Table 12. The highest TPR of each threshold with the corresponding FPR in the Port-degree analysis on Day 14.....	95
Table 13. List of the Well-Known Ports	109

LIST OF FIGURES

FIGURE	PAGE
Figure 1. Bipartite graph representation.....	20
Figure 2. Bipartite network projection of Figure 1 network	22
Figure 3. Clustering coefficient.....	23
Figure 4. Normal Behavior Window (<i>NBW</i>)	32
Figure 5. Changes of size in Malicious Behavior Window (<i>MBW</i>) depending on the current trust	33
Figure 6. Malicious Behavior Window (<i>MBW</i>)	35
Figure 7. Log scaled host-degree distribution for one day of data.....	37
Figure 8. Log scaled port-degree distribution for one day of data.....	38
Figure 9. Log scaled betweenness centrality distribution for one day of data.....	38
Figure 10. Distribution of clustering coefficients	42
Figure 11. Hostscan [37].....	44
Figure 12. Portscan [37].....	45
Figure 13. Hybrid Scan	46
Figure 14. Clustering coefficient distribution for one day of data.....	51
Figure 15. The changes of the Beta reputation depending on the behavior patterns ...	60
Figure 16. The changes of the PRT depending on the behavior patterns	62
Figure 17. Comparisons of the amount of IP addresses to be analyzed between the original and graph represented in the European dataset.....	68

Figure 18. Comparisons of the amount of ports to be analyzed between the original and graph represented in the European dataset.....	68
Figure 19. Comparisons of the amount of IP addresses to be analyzed between the original and graph represented in the URI dataset	69
Figure 20. Comparisons of the amount of ports to be analyzed between the original and graph represented in the URI dataset	69
Figure 21. Relationships between the degree-based analysis and trust-based.....	76
Figure 22. European dataset log-scaled host-degree histogram.....	85
Figure 23. Host-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 1.....	88
Figure 24. Host-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 14.....	89
Figure 25. Host-degree analysis and the AUC during the 14 days (Line plots)	89
Figure 26. Host-degree analysis and the AUC during the 14 days (Box plots).....	90
Figure 27. Port-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 1.....	92
Figure 28. Port-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 14.....	93
Figure 29. Port-degree analysis and the the AUC during the 14 days (Line plots)	93
Figure 30. Port-degree analysis and the AUC during the 14 days (Box plots).....	94
Figure 31. Clustering coefficient analysis and the AUC during the 14 days (Line plots)	96

Figure 32. Clustering coefficient analysis and the AUC during the 14 days (Line plots)	97
Figure 33. Data Analysis Scenario of the European dataset	102
Figure 34. Data Analysis Scenario of the URI dataset	105
Figure 35. Definitions of terms of the network flows	108
Figure 36. A normal node that has not been classified as malicious	113
Figure 37. A hostscan attack node that has been classified as malicious	116
Figure 38. A hostscan attack node that has been showed in the non-URI flows	120
Figure 39. A hostscan attack node that has been showed in the URI- server flows	123
Figure 40. A portscan attack node that has been classified as malicious	127
Figure 41. A portscan attack node that has been showed in the URI- server flows	131
Figure 42. A portscan attack node that has been showed in the non-URI flows	135
Figure 43. A hybrid-scan attack node that has been classified as malicious	138
Figure 44. A hybrid-scan attack node that has been showed in the URI-client flows	142
Figure 45. A hostscan attack node that has not been classified as malicious	145
Figure 46. A portscan attack node that has not been classified as malicious	148
Figure 47. A hybrid-scan attack node that has not been classified as malicious	151
Figure 48. A node that showed a temporary error	155
Figure 49. A suspected on-off attack node	157
Figure 50. A hostscan attack node that has been showed in the URI-client flows	158
Figure 51. The host degree-based detection rates of the client nodes	161
Figure 52. The host degree-based detection rates of the server nodes	162
Figure 53. The host degree-based detection rates of the non-URI nodes	163

Figure 54. The port degree-based detection rates of the client nodes	165
Figure 55. The port degree-based detection rates of the server nodes	166
Figure 56. The port degree-based detection rates of the non-URI nodes.....	167
Figure 57. The betweenness centrality-based detection rates of the client nodes.....	168
Figure 58. The betweenness centrality-based detection rates of the server nodes.....	169
Figure 59. The betweenness centrality-based detection rates of the non-URI nodes	170

CHAPTER 1

INTRODUCTION

The internet along with the corporate network plays a major role in work/personal use and has evolved into a global communication and service network. As of 2013, 83.3% of the United States households reported computer ownership, and 74.4% of all households reported Internet use [1]. The openness and scalability of the Internet have provided a flexible platform for private customers, professionals, and academics, with non-commercial or commercial interests. For instance, electronic commerce, social networking, and national infrastructure for which delivering oil, gas, power, and water, rely on access to the information network. However, there exist various threat sources and vulnerabilities including software bugs, operating system bugs, and operating system misconfigurations. The number of the new threats continues to grow, and the creation of new malware was observed an average of 230,000 new types each day in the second quarter of 2015. Moreover, a total of 21 million new threats were observed during the three months [2].

1.1 Problem Statement

In recent years, there has been growing interest in network intrusion detection due to the emergence of various kinds of large-scale distributed systems, such as Internet, server clusters, cellular networks, computer networks, distributed databases, network file systems, etc. Networks are complex interacting systems, where failures and

attacks are common. Therefore, network-monitoring systems that are used for identifying, diagnosing and treating anomalies, have become a fundamental part of network operations. Without this kind of monitoring systems, networks are not able to operate efficiently or reliably. Anomaly detection is one of the intrusion detection monitoring methods that characterizes known or unknown anomalous patterns of an attack, a virus, or a malware. There are many proposals for anomaly detection methods that are based on pattern, rule, state, integration, or statistical analysis of the data that can be collected, for example, at the firewall of a given computer system. Such data usually represents network flows that consist of transaction time, content type, source IP address, destination IP address, source port, destination port, IP protocol, size of sent and received packets, source country, destination country, packet contents, etc. The data will provide an accurate detection of malicious nodes if an Anomaly Detection System (ADS) can analyze every network flow separately. However, it might not be possible due to the physical and practical limits to the amount of data storage and computation because the number of network flows are huge. For instance, the data considered in this dissertation has been collected from one of the network firewalls at the University of Rhode Island consists of more than 3.6 billion network flows during the 90 days from February 10th to May 28th. Since each flow includes chunks of multiple packets that have been exchanged between a source node and a destination node, the actual number of packets that have been observed was much bigger than 3.6 billion. Thus, this dissertation attempts to reduce the dataset to be analyzed to detect malicious behaviors.

The term network refers simply to a collection of elements and their inter-relations [3]. The relationships between the nodes in a network can be presented as a graph, and it allows us to apply various graph theory techniques for a statistical analysis-based Anomaly Detection System (ADS). One of the advantages of the graph is that we can reduce the amount of data that needs to be analyzed because the duplicate relationships between nodes can be presented on one edge in a graph. We analyzed the generated graphs from network flows data with statistical analysis methods that assume the normal data instances occur in high probability regions while malicious data instances occur in low probability regions.

To create appropriate rules for decision-making about network node behaviors in a dynamic network environment, we adopted an outlier detection method from the financial analysis [4]. The nodes in a dynamic network environment, such as the Internet, show different behaviors over time because the network topology changes and nodes and/or edges may come and go. Thus, a threshold, which detects malicious behaviors, needs to be dynamically selected depending on the given dataset, and we used the outlier detection method to implement an adaptive threshold. The adaptive threshold detects malicious behaviors and is used for a trust management scheme that is introduced in next paragraph.

The concept of trust and the trust management schemes adopted from the social science will help to express a large amount of past data implicitly by providing a cumulative evaluation method for each node. Also, by construction the trust management scheme will help to convert the various ranges of the graph-based node characteristics and behaviors into a measure of trust with a fixed range of values

between zero and one, where a value close to zero indicates less trustful while a value close to one indicates more trustful. Since each statistical analysis method produces different ranges of results, the related statistical background will be required to understand the results. The big numbers may represent malicious behaviors in some statistical analysis methods while the numbers can represent normal behaviors in some methods. If we convert the results of each statistical analysis via the trust management schemes, they can be expressed as understandable information without the related statistical background in a fixed range of values between zero (less trustful) and one (trustful). Using trust values computed from multiple node characteristics for the same node can be integrated into one single measure that will reflect the overall level of maliciousness of the node.

Note that in this dissertation, the trust scheme originally proposed by [5] has been modified to allow indicating how close a behavior to the normal behavior and malicious behavior.

1.2 Research goals

An efficient statistical analysis-based ADS needs to: a) store and analyze less data, b) be able to deal with compressed, encrypted, or fragmented packets, c) show low false positive rate, and d) create appropriate rules for decision-making about network node behaviors in a dynamic network environment.

First of all, this dissertation uses graph-based statistical analysis method because a graph can reduce the amount of data by presenting the relationships between nodes on one edge, and the method focuses on the unencrypted and uncompressed part of the

network packets, for example, the source IP address, destination IP address, source port, and destination port. To implement the graph-based analysis, the ADS needs to represent data as Network Graph(s). Second, we develop 1) a trust management scheme to avoid erroneously labeling nodes with a few malicious behaviors and 2) an outlier-based adaptive threshold to create appropriate rules for decision-making about network node behaviors in a dynamic cyber environment. A statistical analysis method detects malicious behaviors that occur in low probability regions, and the evaluation should be careful to minimize its false positive rate. Moreover, the analysis should be able to adapt to a dynamic network environment, such as the Internet, to maintain an appropriate detection rate.

This dissertation aims to achieve the following goals. First, we will represent the network flows dataset as Network Graph(s). Second, the behaviors of each node will be characterized by using graph-based analysis. Third, the system will detect malicious behaviors using statistical analysis method. Fourth, we will develop a dynamic threshold method, so that the ADS can adapt to a dynamic cyber environment. Last, each node will be evaluated based on aggregate behavior over time to ensure that no one behavior causes the node to be labeled as malicious.

1.3 Data description

The proposed anomaly detection system will be evaluated using two different datasets. The first dataset was collected by European Internet Service Provider in 2007, and the second dataset was collected at the firewall of the University of Rhode Island (URI) computer network in 2014.

The first dataset is the same dataset that was used by Ding et al. in [6] for intrusion detection via community-based method (Section 2.2.2). The dataset contains network flows collected by a European Internet Service Provider (ISP), sampled based on 1:1000 packet sampling rate, and stored over 14 days from May, 1st to May, 14th in 2007. Totally, 34,647,824 network flows were collected from 43,016 unique source nodes to 1,449,598 unique destination nodes where each flow is represented by a source port, a destination port, time and duration of the flow, and size in bytes and packets. Additionally, the dataset contains DShield Logs obtained from a community-based collaborative firewall log correlation system by the SANS Institute [7]. The DShield Logs contains 9,724,227 unique IP addresses marked as possibly malicious. Although not complete, DShield Logs have been proven to be useful for validation of proposed intrusion detection methods in a number of studies [6], [12].

The second dataset, significantly bigger than the first dataset, contains network flows that were collected by University of Rhode Island Information Technology Service (URI-ITS) at one of the firewalls in the university, sampled based on 1:100 packet sampling rate, and stored from February 11th to May 22nd in 2014 for 90 days. Totally, 36,986,233 network flows were observed that access from/to the University of Rhode Island (URI) Network. 8,616 nodes belonged to the URI Network among 19,468,221 unique source nodes, and 31,279 nodes belonged to the URI Network among 528,222 unique destination nodes where each flow is represented by a source port, a destination port, time and duration of the flow, size in bytes and packets, a type of application, a sequence number, an action, a flags, a source country, a destination country, etc.

There have been a few special events while collecting the flows. One of the servers in URI network was attacked on March 6th, so the collection was stopped at 23:59:59. We started collecting the flows on March 11th again, but there was another attack. The collection needed to be stopped at 06:15:57, and we started the collection at 15:26:08 next day. We had to stop the collection at 15:27:23 on April 22nd again due to another attack and resumed the collection at 12:00:50 on May 6th.

1.4 Dissertation Structure

Chapter 2 will present the background and related scholar research work in statistical analysis-based Anomaly Detection System (ADS) and trust management scheme. The chapter will give reviews of the types of intrusions, types of ADS, graph theory, statistical analysis methods, adaptive threshold, and trust management scheme that have been used in this dissertation. Chapter 3 will propose a new method for detecting malicious nodes from network flows based on a statistical analysis-based anomaly detection by representing the results by using trust. This chapter also introduces an adaptive threshold that creates appropriate rules for decision-making in a dynamic cyber environment. Chapter 4 will provide descriptions of each experiment design and the results that have been composed for evaluating the goals of this dissertation. The experiments will show 1) network graph can reduce the dataset, 2) graph-based analysis can characterize each node's behaviors, 3) the proposed trust management scheme can be used to represent the statistical characteristics that we are investigating, 4) the proposed adaptive threshold performs better than static thresholds in a dynamic environment, and 5) the trust management will evaluate each node in

aggregate and over time. Chapter 5 of this dissertation will discuss how this work has met the goals specified in Section 1.2 and summarize findings for the experiments in Chapter 4. The future direction of the research work is also discussed in the chapter.

CHAPTER 2

BACKGROUND AND RELATED WORK

A variety of methods has been proposed and developed for the purpose of intrusion detection. Some methods detect malicious nodes by matching the traffic pattern or the packets using a set of predefined rules that have been already known. On the other hand, some methods monitor the behaviors over time and detect malicious behaviors without the predefined rules. These methods can implement an Intrusion Detection System (IDS) and protect the system from various threats. This dissertation approached the latter method to implement an IDS based on statistical analysis methods.

This chapter presents the related works of this dissertation. Section 2.1 provides an introduction to IDS. Section 2.2 discusses the statistical analysis-based anomaly detection systems, the related definitions, the statistical analysis methods, and terminology that are used in this dissertation. Section 2.3 introduces the related graph theory, and Section 2.4 provides an introduction to trust management schemes that can evaluate. Section 2.5 discusses an outlier detection method to implement an adaptive threshold to create appropriate rules in a dynamic network environment, and Section 2.6 overviews the potential intrusions to computer and computer networks. Section 2.7 summarizes this chapter and discusses how the reviewed studies contributed to this research.

2.1 Intrusion Detection System (IDS)

For a clear understanding of ADS, we need to know IDS first. In recent years, the attacks have been sophisticated, and new vulnerabilities (known as Zero-day exploits) are routinely brought. As complete prevention of computer attacks is not possible, the IDSs play a very important role in minimizing the damage caused by different attacks. An IDS is a special purpose device or software to detect suspicious behaviors, such as unauthorized access to a computer system or a network. The concept of IDS was first introduced by James P. Anderson [8] and formed by Dorothy Denning [9] in 1980 and 1987 respectively. An IDS audits activities performed in computer systems and networks, then analyzes the data to find evidence of malicious behaviors. The activities can be network packets, operating system calls, audit records produced by operating system auditing facilities, or log messages produced by applications [10]. When IDS detects an attack, an alert that describes the type of the attack is provided.

There are two complementary trends in intrusion detection for IDS: misuse detection and anomaly detection [11]. Misuse detection is also known as knowledge-based detection that aims to identify patterns of known bad behavior. On the other hand, anomaly detection is known as behavior-based detection that builds a profile of the normal operation from the historical data of the activities of system and/or users that aims to look for abnormal behavior. These methods can be used together; a “hybrid detection” that is based on both the normal behavior of the system and the intrusive behavior of the attackers.

2.1.1 Misuse Detection

The misuse detection method analyzes a system and/or user activities and tries to find a match between the activities and the definitions or signatures of known attacks that have been identified as bad behaviors [12], [13] . Thus, the misuse detection method is commonly known as a signature-based method. This method requires developing a knowledge database and a rule engine to detect intrusions.

This method is generally used in commercial products due to its advantages [14]. First, misuse detection method is very efficient in detecting attacks with usually high levels of accuracy and with a minimal number of false positives [15]. Second, this method can quickly detect specially designed intrusion tools and techniques. Last, it provides an easy-to-use tool for maintaining and monitoring systems, even if an administrator of the systems is not an expert in security [13].

However, there also exist disadvantages. First, misuse detection method can only detect known attacks, so the systems must be updated with newly discovered attack definitions or signatures. Second, since this method works on the basis of predefined signatures, it is not able to detect attacks when they changed slightly and a variant of the attacks is obtained, even if the attacks are well-known [13]. Last, misuse detection method is not able to handle compressed, encrypted or fragmented packets because the signatures cannot be verified.

2.1.2 Anomaly Detection

Anomaly detection method attempts to learn the features of behaviors and events of system and/or users over a period time to build a profile of normal behaviors, then

looks for malicious behaviors that deviate from the normal profile. Thus, anomaly detection method is commonly known as a behavior-based method. Since the misuse detection method is no longer accurate, this method is typically conceived as a more powerful method in academic research due to its theoretical potential for addressing novel attacks [14].

By using a collection of common activities of a user, anomaly detection method builds *Individual Profile* and compares the user's current behaviors to the profile with little deviation from the expected norm. This may cover specific user events. *Group Profile* is built by using a collection of common activities of a group, and it is expected that each member in the group follows the group activity patterns. System designers are able to build their own profiles for their purposes, such as resource profiles or executable profiles [16].

The advantages of anomaly detection method are as follows [13], [17]. First, anomaly detection method is able to detect attacks even when detailed information of the attack does not exist. Second, this method can be used to obtain signatures or definitions for misuse detection method [13]. Third, it is possible to handle compressed, encrypted, or fragmented network packets because the method can be implemented by analyzing the header information that does not be compressed, encrypted, or fragmented in the Internet. Last, it is less dependent on operating-system-specific mechanisms [18]. On the other hand, anomaly detection method also has disadvantages. First, the high false positive rate is generally cited as the main disadvantage of anomaly detection method [13], [18]. Second, this method requires a large set of training data in order to construct normal behavior profile and the

detection is not available during the learning phase [13]. Moreover, behaviors can change over time, so [18] emphasized the importance of the periodic on-line retraining of the behavior profile.

A wide variety of techniques were proposed to approach the anomaly detection problem including machine learning [19]–[23], data mining [24]–[30], hidden Markov models [31]–[33], and statistical analysis [6], [34]–[39]. This research mainly considers the statistical analysis-based method among these methods because it is simple and effective to detect known and unknown attacks [28], [40], [41]. Besides, the statistical analysis-based method does not require prior knowledge about normal activity in order to set limits; instead it learns what constitutes normal activity from its observations, and confidence intervals automatically reflect this increase knowledge [9].

2.2 Statistical analysis-based Anomaly Detection System (ADS)

Statistical analysis-based ADS builds a representation of the network normal behavior by using different features of the network traffic data. The system observes the activity of subjects and generates profiles to represent their behaviors. Typically, it maintains the two profiles that have been discussed in Section 2.1.2; the *Individual Profile* and *Group Profile*. As new events occur, the method updates the current profile and compares it to the previously built profile using some stochastic model [42]. The statistical analysis-based ADS assumes that normal data instances occur in high probability regions of a stochastic model while malicious data instances occur in low probability regions of a stochastic model [43]. Numerous researchers have

proposed the statistical analysis-based ADS, and [42] classified them into two categories; packet-based and flow-based analysis.

2.2.1 Packet-based statistical analysis

Packet-based statistical analysis gathers information about individual packets, and it analyzes the captured packets directly. Staniford et al. [38] proposed Statistical Packet Anomaly Detection Engine (SPADE) that uses a simple frequency-based approach. SPADE was specifically designed for detecting port scan attacks. It observes the TCP SYN packets over a period time and maintains the anomaly score depending on the frequency of a given packet has been seen. Javitz et al. [34] showed a statistical analysis-based analysis that maintains statistical profiles, which are Q and S statistics, to monitor normal behaviors. The Q statistic represents long-term behaviors, and S statistic is deduced from the Q statistic. The S reflects the degree of abnormality of recent behavior, and a node with high S value is classified as a malicious node. When the most recent value for Q has a low probability of occurrence, S has a large value, and vice-versa. In [44], Porras et al. proposed Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD). EMERALD monitors the behaviors by using four statistical measures; discrete types, numerical quantities, traffic intensity, and event distribution. In [39], [45], [46], the authors have been interested in the actual contents of the network flows, which is called payload. In [39] the Anagram detected malicious by using language-independent n-grams based statistical analysis. 1-gram represents the network traffic model that is attack-free and statistically analyzes byte-value frequency distributions [47]. The Anagram extended

the method to the n-grams analysis that extracts from known malicious packets. To put it again, the Anagram computes the mean frequency as well as the variance and standard deviation of each ASCII character and compares to the recent packets. Even a simple statistical analysis, such as histogram, can be used for detecting intrusions [48].

A Bayesian network also provides intrusion detection method with an approach that models statistical dependencies and causal relationships between system variables [49]–[51]. It represents the probabilistic relationships between variables (nodes) and their conditional dependencies (links) via a directed acyclic graph. In Bayesian approaches, conditional probabilities are used to predict the traffic class of network events. Kruegel et al. introduced an event classification-based intrusion detection scheme using Bayesian networks in [50]. The method built a decision network based on special characteristics of individual attacks.

2.2.2 Flow-based statistical analysis

The processing and storage resources required for Packet-based analysis have increased with the number of captured packets. Moreover, with growing use of encrypted and/or compressed packets, the analysis methods have focused more on the unencrypted part of the network packets, for example, the source IP address, destination IP address, source port, and destination port. Based on the information, researchers created network graph models to detect malicious behaviors. Akoglu et al. claimed that graph-based approaches are vital and necessary for anomaly detection systems for the following four reasons; inter-dependent nature of the data, powerful representation, relational nature of problem domains, and robust machinery [52].

- **Inter-dependent nature of the data:** Most relational data could be inter-dependent, and it is necessary to account for related objects in finding anomalies.
- **Powerful representation:** The inter-dependencies could be represented by links (edges) between the related objects. Graphs effectively capture the long-range correlations with the multiple paths. This enables representing the incorporations of nodes and edge attributes/types.
- **Relational nature of problem domains:** The inter-dependencies could exhibit anomalies. For example, a failure of a machine could be an indicator of the possible other failures of the machine or other machines.
- **Robust machinery:** Attackers would not be able to disguise their malicious behaviors without knowing the entire characteristics and dynamic operations of the normal behaviors.

The graphs can be analyzed based on either structure-based or community-based methods. The *structure-based method* identifies substructures in the graph and finds malicious connections. In [53], the authors proposed a Graph-based hierarchical IDS (GrIDS), which aims to detect network-wide intrusions that involve connections between many nodes. GrIDS construct activity graphs representing hosts and network activity, and models an organization as a hierarchy of departments and hosts. Choi et al. visualized the structures of attacks by using parallel coordinates in [37]. The method displayed several network attacks, such as Denial of Service, Slammer, Host scan, and Port scan attacks, on the plane of parallel coordinates by using source IP

address, destination IP address, destination port, and the average packet length information in a flow.

The *community-based method* finds connected groups of nodes in the graph and looks for malicious behaviors. Sun et al. developed a method that finds outliers based on network clustering method [54]. The authors calculated the core-connectivity-similarity for each edge in the graph and used a network clustering method, SCAN [55], to find the minimum threshold for the outliers. Ding et al. proposed a community-based anomaly detection system in [6]. The paper proposed a detection system for malicious sources that uses the definition of intrusion in a social environment, mainly in order to characterize all the nodes in the network flows.

This dissertation has its foundations in the work of the flow-based statistical analysis over the packet-based statistical analysis because of the advantages of the flow-based analysis. First, this analysis method requires less processing and storage resources than the packet-based statistical analysis. The method analyzes the header information that identifies the source and destination of each packet, and the duplicate relationships between nodes will be presented on one edge in a graph. Second, this method also analyzes the header information that is an unencrypted part of the network packets. Third, it allows us to apply various graph theory techniques for statistical analysis methods.

2.2.3 Problems

As we discussed in Section 1.1, the statistical analysis-based method has a few disadvantages. First, the results are not intuitive. The method in [34] provides two

statistical results, which are Q and S statistics, and the human supervisors are required to interpret the results with the relative statistical background because the results are not intuitive whether the number represents normal or malicious, and how big of a number can be considered as a malicious behavior. Second, it is difficult to integrate multiple statistical analysis results into one because the statistical analyses generate the results in different ranges. For instance, the degrees and betweenness centrality can be any number while the clustering coefficient generates a number between 0 and 1. Last, the statistical requires a threshold when they detect malicious behaviors. SPADE [38] used one of the static thresholds between 9 and 15 to find malicious nodes, and the results varied depending on each threshold. Besides, the method provides only the normal and malicious states but does not provide how close to normal or malicious. Anagram [39] also depends on the static similarity threshold between the profiled attack payloads and the recent payloads. This method might provide various states depending on how similar the recent payload is to the profiled known attack payloads. However, when a smart attacker manipulates the payload, it would not be possible to detect the attack. The parallel coordinate method [37] provides fast detection method with very nice visualization, but it also requires static thresholds for each attack and provides only the normal or malicious states. The community-based anomaly detection method [6] depends on statistical analysis results for detecting intruders. The research showed betweenness centrality and clustering coefficient are good indicators of the intruders. However, the results require knowledge of sophisticated statistical concepts.

This dissertation proposes a trust management scheme as a solution to these problems. A trust management scheme is able to scale the results of statistical analyses

between 0 and 1, so the supervisor will know how much the recent behavior or accumulated behaviors are close to the normal or malicious behaviors. Since the trust management scheme scaled the results of various statistical analyses within the same range, they also can be integrated into one with a sophisticated method. Moreover, the trust management scheme will provide more than normal and malicious states, such as a-little-suspicious, suspicious, or highly-suspicious states. For the last problem, we employed an adaptive threshold as the solution.

2.3 Network graphs

Frequently ‘network’ is used interchangeably with the term ‘graph’ since networks are most commonly represented in a formal manner using graphs of various kinds [3]. The subfield of mathematics known as graph theory provides a body of definitions, tools, techniques, and results for discussing graphs and their properties. In this section, we introduce some relevant network graph models that can represent network flows and the techniques that can characterize network node behaviors.

This dissertation has its foundation on a graph-based and a community-based anomaly detection system that has been proposed in [6]. The authors proposed a detection system for malicious sources that uses the definition of intrusion in a social environment, mainly in order to characterize all the nodes in the network flows. The authors used the bipartite network graph, bipartite network projection, degree, clustering coefficient, and betweenness centrality for the measures in the paper and showed that the malicious nodes tend to have relatively high degree, the low clustering coefficient and high betweenness centrality.

2.3.1 Bipartite network graph

Figure 1. Bipartite graph representation

A bipartite network is a graph $G=(V_1, V_2, E)$, where V_1 and V_2 are two disjoint sets of nodes respectively the top and bottom nodes in Figure 1, and each edge in E has one endpoint in V_1 and the other in V_2 . This graph is used for representing ‘membership’ networks. Each edge has endpoints of differing sets, but not between nodes in the same set [3]. Bipartite graphs can represent several types of real-world data, and Sun et al. claimed that the graph can show interesting characteristics of the networks [52], [56], [57].

- **Internet:** In a network traffic bipartite graph, the edges represent the connections between a source node to a destination node, and the differing sets are a group of source nodes and a group of destination nodes. The graph of connections between source nodes and destination nodes can be direct evidence of network attacks, such as Denial of Service, Hostscan, or Portscan attacks [6], [37].
- **P2P systems:** The endpoint in V_1 is a set of files, and the other in V_2 is a set of peers. The graph shows the popularity of each file and point of interests of each peer depending on the set of files [56].

- **Stock markets:** Each set represents the traders and stocks respectively. The edges show the actions of buying and selling, and it can represent the characteristics of the traders and stocks. Besides, it will be useful to identify abnormal traders [56].
- **Research publications:** The authors and conferences from V_1 and V_2 respectively, and the publication relationships were modeled by the edges. The graph can be used for grouping the authors depending on the field of interests and will show the active authors in the field [56]. The research publication bipartite graph can represent the citation networks as well.

As we discussed in Section 2.2.2, the graph-based statistical analysis is interested in the relationships between the source and destination nodes. The network flows can be represented by source nodes (V_1) and destination nodes (V_2), and the nodes constitute the two disjoint sets. Moreover, each flow has one endpoint in the source node set and the other in the destination set (E). Considering these facts, the definition of the bipartite graph shows that the graph is enough to represent the network flows datasets.

2.3.2 Degree

In graph theory, the degree represents the number of edges from the vertex and to the vertex. In network analysis, it shows the number of connections from a source node to the destination nodes [3]. This can be very valuable information that could indicate that a source node is an attacker or that a destination node has been attacked

by an attack that usually shows high degrees (i.e. DDoS, probing, etc.). In Figure 1, V_1 and V_2 are two disjoint sets of nodes respectively the top and bottom nodes, and each edge in E has one endpoint in V_1 and the other in V_2 . Each source nodes from S1 to S11 in the group V_1 has out-degrees 1, 2, 1, 1, 1, 2, 1, 1, 2, 1, and 1. On the other hand, each destination nodes from D1 to D4 has in-degrees 3, 4, 3, and 3.

2.3.3 Bipartite network projection

Figure 2. Bipartite network projection of Figure 1 network

A bipartite network projection is an extensively used method for compressing information in a bipartite network. If nodes a and b share at least one common destination, they are connected in the bipartite network projection [58]. So that the projection discovers the community structure of a bipartite network. For the examples in Section 2.3.1, 1) the projection of the Internet shows the groups of the source nodes, 2) it creates different groups based on the interests in P2P systems, 3) the Stock market presents the groups of interests in the fields and stocks, and 4) the projection of the Research publications can model the co-authorships. Figure 2 shows the one mode projection of the Figure 1 network. The bipartite network projection-based community

detection approaches were proposed by [6], [57], [59]–[64]. Ding et al. formed a projection graph to identify communities in the network and utilized it to find malicious nodes, which do not respect community boundaries in [6].

2.3.4 Clustering coefficient

Figure 3. Clustering coefficient

In graph theory, a clustering coefficient is a metric that quantifies the degree how much the nodes densely connected each other in a graph and tend to cluster together. Clustering coefficient was introduced in social network analysis [65]. The authors presented two versions of clustering coefficient; global and local. Global clustering coefficient is based on the triangular connections between neighboring three nodes and gives an indication of the clustering in the network while Local clustering coefficient gives an indication of the embeddedness of single nodes [65]. In network analysis, it can be a measure of how source nodes tend to create tightly knit groups [3]. The clustering coefficient can be calculated by observing the number of triangles (3-loops) that pass through a node and the maximum number of 3-loops that could pass through the node.

$$C_n := \frac{\text{number of triangles that pass through the node } k}{\text{maximum number of triangles that pass through the node } k} \quad (1)$$

[6] showed that low clustering coefficient can be a measurement of intrusion, which has the tendency to not respect community boundaries. The low clustering coefficient represents whether the node is a member of weakly tied communities and high clustering coefficient represents whether the node is a member of tightly tied communities. For instance, one triangle that passes through node $N2$ ($N2-N3-N4$) is 1, and the maximum number of triangles that could pass through b is 3 (a-b-c, a-b-d, b-c-d), then

$$C_b = \frac{1}{3}$$

In Figure 2, each node from S1 to S11 has the following clustering coefficient values.

Node	1	2	3	4	5	6	7	8	9	10	11
CC	1	0.4	1	1	1	0.4	1	1	0.33	1	1

Table 1. Clustering coefficient of each node

We need to note that the nodes with less than two neighbors are assumed to have a clustering coefficient of 0 or Not a Number (NaN). This research analyzes the clustering coefficients and characterizes the nodes that are not densely tied in its communities.

2.3.5 Betweenness centrality

In graph theory, a betweenness centrality is a measure of a centrality of a vertex within a graph and quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. In network analysis, it can measure the amount of connections of a node to the communities in a network flow [3]. High betweenness centrality represents the nodes that connected to many communities, which do not respect community boundaries [6]. For instance, I may access to many web pages and servers that belong to the University of Rhode Island because I am a student at the university. If my computer has tried to access the servers that belong to the Boston University, Brown University, Harvard University, Massachusetts Institute of Technology, University of Connecticut, and many other universities, my computer would be the bridge among the communities in each university when we present the relationships in a graph. In this case, many shortest paths will pass my computer, and it will increase the betweenness centrality.

2.4 Trust

Trust is an important but complex concept in social science, which represents a directional relationship between two parties that can be called *trustor* and *trustee*. It helps people to make assessments and decisions in unpredictable circumstances by reducing the uncertainty [66]. Trust is also a very important feature for network-based systems that have nodes, which need to interact with one other. In order to facilitate interactions in such system, trust must be addressed for the successful cooperation of nodes in systems such as the internet, wireless sensor networks, e-marketplaces, the

Grid, or peer-to-peer network systems. A trust management scheme manages reputations of each node in a system. The main purpose of a reputation system is to differentiate the states of the nodes in a system depending on each node's behaviors. This section introduces trust management schemes and trust computation methods.

2.4.1 Trust management schemes

Michiardi et al. proposed Collaborative Reputation mechanism protocol (CORE) with the concept of reputation in order to evaluate each node's contribution to the network in [67]. CORE focused on detecting Denial of Service attacks in a wireless sensor network system by using subjective (observation), indirect (positive reports by others), and functional (task-specific behavior) reputations. These are formed and updated with time by direct observations and reported information from other nodes by weighted trust technique [68]. Buchegger, et al. [69], [70] have proposed "Cooperation Of Nodes, Fairness In Dynamic Ad-hoc NeTworks" (CONFIDANT). By adopting Bayesian reputation [5] and trust system, it allows second-hand evaluations. CONFIDANT uses both direct observation of a node and the second-hand information from other nodes by weighted trust technique. This helps to evaluate a neighboring node, and the reputation is used to isolate the malicious nodes from its network [68]. DiPippo et al. proposed Secure Adaptive Routing Protocols (SARP) in [71]. SARP provides a multi-dimensional trust evaluation mechanism that can adapt to dynamic changes in the trust values of nodes in the network to route data from a data source to a base-station. The research used multiple behavioral evaluation methods and represented them by using a trust management scheme; forwarding trust, reporting

trust, availability trust, loop trust, and overall trust. The authors also presented that multiple evaluation results can be integrated into one, and they were able to efficiently find malicious nodes in the network by using the integrated trust. Chae et al. proposed REdeemable reputation based Secure Routing Protocol (RESRP) in [66] that extends the SARP to provide flexible and additional defense scheme against On-off attacks, which attempt to disturb trust management schemes by behaving well and badly alternatively [72], [73]. The authors proposed a new trust management scheme and successfully integrated with the previously proposed trust management scheme, which is SARP.

As shown in these examples, trust management schemes are the tools that manage reputations of each node's behaviors to find malicious nodes. The CORE [67] showed that the trust can be used for detecting network attacks, and the CONFIDANT [69], [70] showed how trust is able to use not only the direct observations but also indirect observations. The SARP [71] successfully integrated multiple trust values, and RESRP [68] showed that the trust management scheme can be easily extended. Moreover, trust management schemes can provide the solutions for the first and second problems that have been discussed in Section 2.2.3; the results of statistical analyses are not intuitive and difficult to integrate multiple analysis results into one. First, it can generate results between 0 (unreliable) and 1 (reliable), which can be understandable for the human supervisors without having the related statistical background. Second, the results generated by various statistical analyses can be integrated into one value because they will be converted in the same range.

2.4.2 Beta Reputation

Bayesian framework relies on ratings, either positive or negative, and use probability distributions, such as the Beta distribution, to come up with reputation scores [5], [74]. The goal of a Bayesian framework is to predict the future behavior of a node whether it will be normal or malicious. By analyzing the probability that the expectation lies within an acceptable level of error, reputation can be determined. The framework has a parameter θ , which is calculated from the observations of the node's prior behaviors, that represents the probability of the node will be normal or malicious on the next behavior.

The beta reputation system [5] is based on statistical updating of beta probability density functions (PDF), which is based on the Bayesian framework. The current reputation score is computed by using the combination of the previous reputation score with the new behavior. The reputation score is represented in the form of the beta PDF parameter tuple (α, β) , where α and β represent the fulfillment of positive and negative behaviors respectively. The beta distribution $f(p|\alpha, \beta)$ can be expressed by using the gamma function Γ as:

$$f(p|\alpha, \beta) := \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha) \Gamma(\beta)} p^{\alpha-1} (1 - p)^{\beta-1} \quad (2)$$

, where $0 \leq p \leq 1$ and $\alpha, \beta > 0$ with the restriction that the probability variable $p \neq 0$ if $\alpha < 1$, and $p \neq 1$ if $\beta < 1$ [75]. The probability expectation value of the beta distribution is given by

$$E(p) := \frac{\alpha}{\alpha + \beta} \quad (3)$$

, where the α and β represent the positive rating and negative rating respectively. The Beta reputation set the α as the rate of positive behaviors B_p plus 1 and β as the rate of observed negative behaviors B_n plus 1, where $B_p, B_n \geq 0$. Hence, the trust value can be predicted by following equation.

$$T := \frac{B_p + 1}{B_p + B_n + 2} \quad (4)$$

2.4.3 Predictability Trust

The beta reputation can have a behavior-based redemption feature. To understand the behavior-based redemption, assume that a friend had a bad behavior in the past, but since then the friend has behaved very well several times. Thus, we can expect that the friend will behave well in the next behavior [66]. Similarly, if a node behaves normal several times, we can expect the node will behave well in the next behavior even if the node showed a few malicious behaviors in the past. However, the behavior-based redemption may not allow a node to regain the trust after being blocked by systems due to a few malicious behaviors because the method relies on subsequent behaviors. If the malicious behaviors have been caused by unintentional temporary errors or an attacker has used fake identifications, it might affect many normal users. Thus, it needs to be cautious before blocking nodes that showed a few malicious behaviors. Moreover, the behavior-based redemption can be disturbed by a smart attacker who wants to remain active in the system by disguising the malicious

behaviors as temporary errors. The attacker can disguise the malicious behaviors by being normal and malicious alternatively so that trust is always redeemed before another attack occurs. This type of attack is referred as an *On-off attack* [66]. This dissertation refers to a $nN-mM$ On-off attack when a node performs n normal and m malicious behaviors alternatively. For instance, if a node showed 4 normal behaviors after 1 malicious behavior, it is a $4N-1M$ On-off attack node.

Chae et al. proposed Predictability trust (*PRT*) to provide defense against On-off attacks in [66], [68]. The authors developed Dynamic Sliding Windows (*DSW*) that change the size of the sliding windows depending on the current reputation. They used the *DSWs* for providing efficient and flexible trust management scheme and successfully detected and defended against On-off attacks.

2.4.3.1. Predictability Trust Calculation

The *PRT* is computed based on how well a node's behavior meets expectations. This model is based on the probability distribution over the binary events, which are normal or malicious behaviors. For instance, if a node showed either higher out-degree than the system designer's expectation or higher port-degree than the system designer's expectation, the system will consider that the node showed a malicious behavior, which is hostscan attack or portscan attack respectively, and we set the event as Malicious Predictability Behavior (*MPB*). On the other hand, if the node behaved as the system designer expected, we set the event as Normal Predictability Behavior (*NPB*). The *PRT* counts the number of *NPB* and *MPB*, and computes the trust as in (5), using the formula of Beta reputation system in (4).

$$PRT := \frac{NPB + 1}{NPB + MPB + 2} \quad (5)$$

Chae et al. showed that a system designer might accept a certain level of damage risk to maintain the desired effectiveness of the system in [66]. When the system detects an On-off attack that causes little damage, for instance, *20N-1M* On-off attack, it may be allowed being active in the system. The authors have proposed dynamic sliding window to achieve this goal.

2.4.3.2. Sliding Windows

Sliding Window (*SW*) is a tool for evaluating neighbor nodes, and the main purpose is to keep track of the past behaviors of a node. It would be the best practice if we could store and observe the entire behaviors of each node, but this is unattainable because a system has limited storage and processor speed. For these reasons, Chae et al. employed *SW*, which allows a certain number of behaviors to be stored, in [66], [68]. A *SW* updates and stores the latest behavior history by removing the oldest observation in its memory when there is a new observation. In the papers, they implemented two types of *SW* for the *PRT*: a fixed sliding window for normal behaviors (*NBW*) and a dynamic sliding window for malicious behaviors (*MBW*).

The *NBW* and *MBW* keep track of both normal and malicious behaviors. Thus, the number of normal behaviors in a *NBW* and the number of malicious behaviors in *MBW* are affected by both normal behaviors and bad behaviors. When a new behavior is observed and stored, it pushes out the oldest behavior from both windows.

Therefore, this structure allows behavior based redemption, so even if a node had several malicious behaviors, but it is not blocked by the security system, the *PRT* of the node can be recovered after several normal behaviors. Besides, the *NBW* and *MBW* will keep the latest behaviors because the first feature discards the oldest behaviors from the windows. Thus, the system designer can manage the system as the fresh information. Last, this structure requires more number of malicious behaviors to detect a patterned attack than the Separated structure and Mixed structure. This allows the system to be more circumspect for discriminating malicious nodes.

Figure 4. Normal Behavior Window (NBW)

a) Normal Behavior Window: The main purpose of the *NBW* is to observe the number of normal behaviors among the stored behaviors. A *NBW* stores both normal and malicious behaviors, but it counts only the number of normal behaviors. This makes it possible to consider only the fresh normal behaviors while keeping in mind the overall pattern of behaviors in the recent past. A system designer is able to provide a specific number of opportunities for malicious nodes by setting the size of *NBW* [66].

Figure 4 shows how the *NBW* works. The top part of the figure shows the initial status of a *NBW* with a size of five, and the *NBW* kept four normal behaviors and one malicious behavior. The bottom part of the figure shows the same window after four more behaviors. In this figure, the behavior history in the storage is represented by the white boxes, so it removes any behaviors in the shaded boxes. The boxes on the left of the “*Current*” arrow are the past behaviors that it stores in the storage. The box that the arrow points out is the latest behavior, and the boxes on the right side of the arrow represent space for future behaviors that have not occurred yet. In this example, the number of normal behaviors changed from four to three after four behaviors in 2N-1M attack [66].

Figure 5. Changes of size in Malicious Behavior Window (*MBW*) depending on the current trust

b) Malicious Behavior Window: The authors were more interested in the malicious behaviors than the normal behaviors because they are harmful to the system, and the primary purpose of *PRT* is to detect malicious nodes. However, to avoid erroneously labeling nodes as malicious, we need to be cautious in discriminating the malicious nodes. For these reasons, the authors developed a *MBW* that allows us to

observe more previous malicious behaviors depending on the current trust value. So as trust decreases the size of *MBW* increases as shown in Figure 5.

A *MBW* stores normal and malicious behaviors but counts only the number of malicious behaviors. The size of the window changes dynamically as the trust of the node changes, and a system designer is able to set a maximum window size for the *MBW*. Chae et al. performed a set of analytical tests that indicate the tolerance to attack for a given maximum window size and discussed the analytical results in [68]. By referencing the analytical results, a system designer is able to determine how much damage the system can tolerate and set the *MBW* maximum size accordingly [66].

$$Size := i \times MinTrust + j \quad (6)$$

$$MaxSize := i \times MinTrust + j \quad (7)$$

$$MinSize := i \times MaxTrust + j$$

While the system designer set the maximum and minimum *MBW* size, the dynamic size of the *MBW* is computed by (6), where $0.0 \leq MinTrust \leq MaxTrust \leq 1.0$, and the i and j are computed using simultaneous equations (7). By adjusting the values of *MaxTrust* and *MinTrust*, the designer can affect the range of the size for the *MBW* [66].

Figure 6. Malicious Behavior Window (MBW)

When a node observes a malicious behavior, the current trust gets lower. Since the *MBW* wants to observe more number of previous behaviors when the trust is low, it increases the size of the window. Figure 6 shows an example how the *MBW* works. When a malicious behavior was observed from a node, the current trust for the node gets lower. In Figure 6, we see on the top a window size of five with one malicious behavior. Thus, we want to observe more number of previous behaviors. Later, as presented in the window on the bottom of the figure, two more malicious behaviors have occurred. The *MBW* size has increased because the malicious behavior lowered the trust of the node so that we can count three malicious behaviors in the *MBW* while the *NBW* in Figure 4 allows us to count only two malicious behaviors.

One of the advantages of the *PRT* is that the method is able to provide long-term observation-based second chances to each node and allow a certain level of risk in order to maintain the efficiency of the system [66], [68]. For instance, if an attacker hides/disguises the IP addresses of the attack nodes or composes attacks behind a device such as VPN or Proxy server as we discussed in Section 2.6.2, the system still

allow the other normal users could connect to the system by taking a certain amount of risks.

However, the trust management schemes still require thresholds to evaluate a behavior whether it is normal or malicious. This dissertation uses an outlier method to provide adaptive thresholds for the trust management scheme.

2.5 Outlier-based adaptive threshold

Even though trust management schemes can provide a method to avoid erroneously labeling nodes with a few malicious behaviors, the methods still have an issue that the evaluation method depends on thresholds. It is not easy to set an appropriate threshold in a dynamic environment, such as the Internet because if a threshold is too high, malicious nodes cannot be detected while a low threshold has a possibility that normal nodes would be classified as malicious. One of the proposed ways to solve this issue is adaptive threshold method.

Adaptive threshold method was proposed in [76] and is also known as time-varying threshold method. The method adjusts its thresholds depending on the recently collected data sets. An outlier is something that does not belong to any existing group or class. The method computes outlier factor of every point in the dataset based on the nearest neighborhood it has [77]. Many researchers have proposed ADSs based on the outlier detection methods [78]–[80].

An outlier can be defined as an observation point that is distant from other observations [81], [82], and it may indicate a systematic error, faulty data, or

erroneous procedures. Since outliers occur in low probability region, this research considered them as anomalous and malicious behaviors.

Figure 7. Log scaled host-degree distribution for one day of data

Figure 8. Log scaled port-degree distribution for one day of data

Figure 9. Log scaled betweenness centrality distribution for one day of data

Figure 7, 8, and 9 show the distribution of host-degree, port-degree, and betweenness centrality for one day of the European dataset (Section 1.3) respectively, and they showed heavy-tailed distributions. A heavy-tailed distribution has heavier tails than the exponential distribution, and it is common within computer systems from degree distributions in the internet and social networks to file sizes and interarrival times of workloads [83]. The right tail of the distribution is of interest because it is the low probability region that malicious data instances occur (Section 2.2). For instance, the maximum host-degree was 6,455 in the European dataset, and it represents that the node attempted to access to 6,455 different IP addresses in a day. As shown in the following simple equation, $6455/(24 \times 60) \approx 4.5$, the node attempted to access average 4.5 different IP addresses every minute, and the behaviors of the node cannot be considered as normal. Besides, the maximum port-degree was 14,186 in the European dataset, and this also represents the node attempted to access to average 9.85 different ports every minute by the following simple equation, $14186/(24 \times 60) \approx 9.85$.

Schluter et al. [4] proposed a method that identifies multiple outliers in heavy-tailed distributions, and we employed this method to implement the adaptive threshold for the trust management scheme. The authors treated the problem of multiple outlier detections in heavy-tailed distribution through outward testing using the k most extreme order statistics. In other words, if we consider a sample of size n and the associated order list is $X_{1,n} \geq X_{2,n} \geq \dots \geq X_{k,n} \geq \dots \geq X_{n,n}$, the method finds the k extreme order statistics that might not belong to the population of interest.

A sequence of null hypotheses was indexed by κ and noted as $H_{0,k}$, where $k = 1, 2, \dots$ and the k^{th} order statistic $X_{k,n}$ belongs to the population of interest. The dataset is tested sequence of null hypotheses $H_{0,k}, H_{0,k-1}, \dots, H_{0,1}$. The outlier point k was specified if R_i, r_i exceeds the critical level $\delta^{-1/\alpha}$, where δ is the size of tests.

The R_i is a ratio of order statistics, and the authors compared this statistical results because the ratios converge for heavy-tailed distributions extreme spacings diverse and are dependent [4].

$$R_i := \frac{X_{i,n}}{X_{i+1,n}} \quad (8)$$

,and the r_i is the individual critical level for the overall test to have size δ .

$$r_i := [1 - (1 - \delta)^{1/k}]^{-1/(\alpha i)} \quad (9)$$

, where the α is the nuisance parameter. The α is estimated by the Hill estimator [84]. However, the original Hill estimator requires a specific number of extremes to be included for estimation. Thus, Schluter et al. [4] modified the Hill estimator.

$$\widehat{\alpha}_{-k}(\kappa) := \left[\frac{k-1}{\kappa - (k-1)} \ln X_k - \frac{\kappa-1}{\kappa - (k-1)} \ln X_\kappa + \frac{1}{\kappa - (k-1)} \sum_{i=k}^{\kappa} \ln X_i \right]^{-1} \quad (10)$$

,where the κ is the number of extremes. The authors noted that there are more sophisticated methods to robustly estimate the tail parameter, but the tail index is just a nuisance parameter for this method [4]. Therefore, they referred (10) as an estimator.

If R_i and r_i of $H_{0,k}$ do not exceed the critical level $\delta^{-1/\alpha}$, the next null hypothesis $H_{0,k-1}$ will be tested with R_{k-1} . The test will be continued until the first outlier is detected or the sample maximum has been tested.

We note that the outlier-based adaptive threshold method cannot be applied to the clustering coefficient because the clustering coefficient does not show the heavy-tailed distribution as shown in Figure 10. Thus, we decided to set a static threshold only for the clustering coefficient analyses. The static threshold has been decided based on a statistical analysis that can maximize the efficiency of the detection. The detailed statistical analysis is presented in Section 4.3. Also, the zero clustering coefficient values were excluded from the analysis because it represents there was not enough connections to evaluate the node with the clustering coefficient (Section 2.3.4).

Figure 10. Distribution of clustering coefficients

The outlier point of a dataset will be used to set our adaptive threshold method for the trust management scheme. The outlier point will be a threshold that classifies malicious behaviors, and the statistical results of the rest of the behaviors will be used for another threshold that classifies normal behaviors. The detailed usage will be described in Section 3.2.

2.6 Intrusion overview

An intrusion to computer and computer network is any unauthorized activity that attempts to compromise the integrity, confidentiality or availability on a computer or a computer network. The system intrusions may cause the following dangers. First, personal data can be stolen. In [16], the authors warned that the digital personal data loss is more dangerous threats to people than physical personal data loss because

people can know the physical data loss and can take precautions, but in digital loss, people may even never know that their data was stolen. Second, privacy can be compromised. In these days the information about an individual is stored online, such as credit and debit cards, personal address, phone numbers, etc. Once the systems that store the information is compromised, the privacy of those individuals is compromised as well. Last, the compromised systems can be a part of cybercrime. A cracker who breaks into your system might use the system for the purpose of cybercrime.

In order to detect intrusions, it is necessary to have a clear understanding of how attacks work. This section presents an introduction to potential intrusions to computer and computer networks. Among the various attacks, this research focuses on the Hostscan and Portscan attacks, which can be easily detected by the statistical analysis-based IDS.

2.6.1 Scanning attacks

This research considers the hostscan and portscan attacks because they are relatively easy to detect by using the statistical analysis-based IDS. Scanning is a technique for discovering available target nodes and the target nodes' open ports. By sending port probes, which are disguised as client requests, to a range of target IP and port addresses. Although sometimes system administrators use this technique to verify security policies of their network systems, the technique is used by attackers to identify running services on a target node with the aim of exploiting a known vulnerability of that service.

Choi et al. defined nine graphical signatures, which are the Portscan, Hostscan, Worm, Source-spoofed DoS (port fixed), Backscatter, Source-spoofed DoS (port varied), Distributed hostscan, Network-directed DoS, and Single-source DoS, in [37].

2.6.1.1. Hostscan

Figure 11. Hostscan [37]

An Internet Protocol address (IP address) is a numerical label assigned to each device and identifies each device in a network for communication. An attacker or a worm virus may scan multiple IP addresses to find victim machines, which provide a known vulnerable service. Hostscan is a technique for discovering available target nodes and the target nodes' open ports. As shown in Figure 11, the attacker sends port probes, which are disguised as client requests, to the same port on a range of target IP

addresses, the attacker is aware of a particular vulnerability and tries to find susceptible machines. The most common are SYN scans and ACK scans but other TCP flags combinations can be used [85]. This dissertation assumes that the hostscan nodes will show high out-degrees and low port-degrees.

2.6.1.2. Portscan

Figure 12. Portscan [37]

Network ports are the entry points to a machine that is connected to the Internet. Through a port, a service is able to communicate with a client application, process requests and send responses back. An attacker may sometimes exploit vulnerabilities in the server code so the attacker can obtain access to sensitive data or execute malicious code on the server remotely. Portscan is a technique for discovering

available target nodes' available ports. As shown in Figure 12, this probes several destination ports on a single target node. If the attacker naively scans a target node, it is easy to detect because even the target node can detect the attack without any support from other machines. This scan frequently carried out via UDP [85]. This dissertation assumes that the portscan nodes will show low host-degrees and high port-degrees.

2.6.1.3. Hybrid-scan

Figure 13. Hybrid Scan

Hybrid scan is a technique that probes several destination ports on several target nodes. In [86], the authors described that this method can yield a hit-list, which is a collection of potentially vulnerable target nodes, for future exploitation. This dissertation assumes that the hybrid-scan nodes will show high host-degrees and high

port-degrees. In other words, the node will show low degree trust and low port degree trust.

Since these types of attacks attempt the relatively large number of connections [37] or do not respect the community boundaries [6], this research will detect the attacks by using flow-based analysis and graph-based methods as we discussed in Section 2.2.2.

2.6.2 Deception of identity

The flow-based analysis method inherits the disadvantages of the ADS. An attacker may disturb the method and increase the false positive rate (Section 2.1.2) by forging the identity. When a system connects to the Internet, the Internet Service Provider (ISP) assigns an IP address to the system. The system's activity can be tracked with the IP address, and the IP address can specify the location of the system and track the Internet activities. The sophisticated hackers know about hiding their tracks. This makes it difficult for systems to decide which IP addresses should be blocked because it could affect many normal users.

There exist many methods that can hide the IP address [87]–[89]. First, Virtual Private Network (VPN) provider can assign a new IP address to the system, and every traffic from the system routes through the VPN network. This makes it possible to hide the true IP address assigned by the ISP. Second, there are thousands of free web proxy servers that the system can use to hide the true IP address and connects the Internet anonymously by connecting the network through an intermediate “proxy”, which relays the information back and forth between the system and the destination.

When the VPN or the proxy server was blocked by the secure system, the other users who connect the system through the VPN or the proxy server would be blocked together no matter they were normal users or not. Thus, the secure systems need to be cautious before blocking the suspicious IP addresses.

2.7 Summary

We provided the background and related scholar research work in this chapter. Section 2.1 reviewed the types of IDS and presented the advantages and disadvantages of ADS. Section 2.2 discussed the Statistical analysis-based ADS with the advantages of the flow-based statistical analysis. The related graph theory for the flow-based statistical analysis was presented in Section 2.3 and discussed how the theory can characterize each network node's behaviors. Section 2.4 provided the contextual literature review of the trust management scheme that was used for representing the results of each statistical analysis method, and Section 2.5 gave a background research for an adaptive threshold. Section 2.6 presented the attacks that this dissertation aims to detect with the statistical analysis-based ADS. Chapter 3 will present an ADS (Section 2.1) that can detect network attacks (Section 2.6) with the flow-based statistical analysis (Section 2.2) by characterizing each node's behaviors (Section 2.3) and representing the results with a trust management scheme (Section 2.4) in a dynamic environment (Section 2.5).

CHAPTER 3

STATISTICAL NETWORK ANALYSIS BY USING TRUST

This dissertation proposes a statistical network analysis method by using trust to implement an effective IDS, especially an effective ADS. First of all, we reduced the amount of dataset to be analyzed by representing the relationships between nodes with graphs as we discussed in Section 2.3. Graph has been used by many research proposals to implement relationship-based ADS [6], [37], [56], [59], [62], [63]. In particular, Choi et al. [37] detected malicious behaviors by using parallel coordinate graphs with degree analysis, and Ding et al. [6] showed that the nodes, which do not respect community's boundaries, can be classified as malicious nodes. However, it is hard to understand the produced results of the statistical analyses in the two research proposals without statistical background because each result has different meaning depending on the analysis method. For the same reason, the results cannot be integrated into one value. Moreover, a certain threshold is required for each analysis to detect malicious behaviors, but it is difficult to decide an appropriate value in a dynamic environment, such as the Internet, because a high threshold cannot detect malicious nodes while a low threshold has a possibility that normal nodes would be classified as malicious as we discussed in Section 2.5.

This chapter proposes an analysis method to achieve the goals of this dissertation in Section 1.2. Section 3.1 discusses how we can characterize each node with statistical analyses, and Section 3.2 proposes a method that represents the produced

results using a trust management scheme. Section 3.3 presents how the Predictability Trust (PRT) integrates multiple statistical analyses, and Section 3.4 shows how the produced trust values can be integrated into one. Finally, Section 3.5 summarizes Chapter 3.

3.1 Node characterization

The parallel coordinates analysis method [37] showed that each scanning attack has its own behavior signature as we discussed in Section 2.4. A hostscan attack node shows high host-degree and low port-degree, a portscan attack node shows low host-degree and high port-degree, and a hybrid-scan attack node shows high host-degree and high port-degree. Therefore, we can characterize each node's behaviors by analyzing the degree information whether it is a scanning attack node or not. Moreover, the clustering coefficient and betweenness centrality can be used for characterizing the nodes that do not respect the community's boundaries as shown in [6]. Thus, this dissertation utilizes the host-degree, port-degree, clustering coefficient, and betweenness centrality for characterizing each node's behaviors from bipartite and projection graphs.

There exist two issues we need to address to utilize the four factors. First, [37] detected malicious nodes based on the high degree information, and [6] showed that malicious nodes tend to show low clustering coefficient or high betweenness centrality information. However, it is not clear that how big degree and betweenness centrality is malicious and how small it is normal. It is also not clear how small clustering coefficient is malicious and how big it is normal. Moreover, the Internet is a dynamic

environment that changes every day, every hour, and every minute. Thus, we needed to create an appropriate rule for detecting the malicious behaviors by considering the overall behavior conditions. To solve this issue, we applied the outlier-based adaptive threshold method to the degree and betweenness centrality information to detect malicious behaviors that occur in low probability regions. We note that the outlier-based adaptive threshold method cannot be applied to the clustering coefficient information because it does not show the heavy-tailed distribution as shown in Figure 14.

Figure 14. Clustering coefficient distribution for one day of data

Second, the degree, clustering coefficient, and betweenness centrality are represented by simple numbers, and it is not intuitive what the numbers mean without

the related statistical background. Besides, each information has different ranges of numbers. For instance, the host-degree showed the range from 1 to 6,455, port-degree showed the range from 1 to 14,186, clustering coefficient showed the range from 0 to 1, and the betweenness centrality showed the range from 0 to 109,171,143.80161 in the European dataset. Thus, it is not possible to integrate the information into one value. We present a trust management scheme as a solution to deal with these two issues.

3.2 Modified Beta Reputation

Beta reputation was proposed by Jøsang et al. in [5], and we discussed this technique in Section 2.4.2. The proposed Beta reputation predicts the future behaviors of each node based on the past behaviors. We employed the Beta reputation; 1) to rate each different ranges of behavior information between 0 (unreliable) and 1 (reliable), 2) to provide a simple calculation method, 3) to provide accumulative evaluation method, so reduce the amount of data to be analyzed, 4) to avoid erroneously labeling nodes with a few malicious behaviors, and 5) to integrate multiple behavior information into one.

In our Modified Beta Reputation method, we defined two thresholds to establish appropriate standards for normal and malicious behavior. We define a Maximum Threshold (*MaxTh*) as a certain value that identifies outliers. The Minimum Threshold (*MinTh*) is defined as an average of the observed values except for the outliers.

Since the Beta reputation relies on a certain threshold to decide whether a behavior met the expectation of a system or not (Section 2.4.2), we employed the

adaptive threshold in Section 2.5 to deal with a dynamic environment, and the threshold is called as *MaxTh* in this dissertation. At the same time, we set another adaptive threshold to detect normal behaviors, which occur in high probability regions, because the range of the collected information can be huge as shown in Section 3.1.

The average is the most popular and well-known measure of central tendency, which is a single value that attempts to describe a set of data by identifying the central position within that set of data [90]. However, the average has one main disadvantage: it is particularly susceptible to the influence of outliers [90]. Since we computed the outliers by using the adaptive threshold, we could overcome the disadvantage. Thus, this research used the average of the behaviors except for the outliers, and the threshold is called as *MinTh* in this dissertation.

We modified the trust management scheme to utilize the two adaptive thresholds because the beta reputation uses only one threshold. Besides, the modified beta reputation can show how the behavior, which is placed between the *MinTh* and *MaxTh*, is closed to the normal behavior or malicious behavior.

$$NBR_i := \begin{cases} 1 & , BV \leq MinTh \\ 0 & , BV \geq MaxTh \\ 1 - \frac{BV_i - MinTh}{MaxTh - MinTh} & , otherwise \end{cases} \quad (11)$$

$$MBR_i := \begin{cases} 1 & , BV \geq MaxTh \\ 0 & , BV \leq MinTh \\ \frac{BV_i - MinTh}{MaxTh - MinTh} & , otherwise \end{cases} \quad (12)$$

$$\begin{aligned}
NNB_i &:= NNB_i + NBR_i \\
NMB_i &:= NMB_i + MBR_i
\end{aligned} \tag{13}$$

$$T_i := \frac{NNB_i + 1}{NNB_i + NMB_i + 2} \tag{14}$$

,where each symbol represents the minimum threshold of the dataset (*MinTh*), the maximum threshold of the dataset (*MaxTh*), Behavior Value of node i (BV_i), Normal Behavior Rate of node i (NBR_i), Malicious Behavior Rate of node i (MBR_i), Number of Normal Behaviors of node i (NNB_i), Number of Malicious Behaviors of node i (NMB_i), and Trust of node i (T_i) respectively. The BV_i includes host-degree, port-degree, clustering coefficient, and betweenness centrality of node i that we discussed in Section 2.3.2, 2.3.4, and 2.3.5, and we the calculated trust values to characterize each node's behaviors.

The outlier-based adaptive threshold technique (Section 2.5) was used for setting the *MaxTh* values in (11) and (12), which decides whether the BV_i was normal or malicious. As we discussed in Section 2.5, the order of tests is from the maximum value to the minimum value. This was applied to the host-degree, port-degree, and betweenness centrality evaluations because the high value is worse than the low value of each BV_i . Thus, the $X_{1,n}$ is the maximum value and $X_{n,n}$ is the minimum value in the dataset of the BVs . The *MinTh* can be decided after removing the outliers from the dataset.

By using the *MinTh* and *MaxTh* values, we can set the NBR_i and MBR_i . When BV_i is greater than the *MaxTh*, the NBR_i will be set to 0, and the MBR_i will be set to

1. On the contrary, if BV_i is smaller than the $MinTh$ value, the NBR_i will be set to 1, and the MBR_i will be set to 0. The equation $(BV - MinTh)/(MaxTh - MinTh)$ in (11) and (12) rates how close the BV_i to the $MinTh$ and $MaxTh$ values. For instance, if we set the $MinTh$ to 5, $MaxTh$ to 10, and BV to 6, the NBR and MBR of the node will be

$$NBR_i := 1 - \frac{BV_i - MinTh}{MaxTh - MinTh} = 1 - \frac{6 - 5}{10 - 5} = \mathbf{0.8}$$

$$MBR_i := \frac{BV_i - MinTh}{MaxTh - MinTh} = \frac{6 - 5}{10 - 5} = \mathbf{0.2}$$

Besides, this method made it possible that the inactivity of a node would not affect the trust changes. Since the both NBR and MBR of the node would be 0s, the both NBB and MBB will keep the same value.

As we note in Section 2.5, we did not apply the outlier-based adaptive threshold method to the clustering coefficient because the clustering coefficient does not show the heavy-tailed distribution as shown in Figure 10. Thus, we decided to set a static threshold only for the clustering coefficient analyses. The static threshold will be decided by comparing the detection efficiency of each threshold from 0 to 1 in the European dataset and DShield list (Section 1.3), and the detailed statistical analysis result is presented in Section 4.3.3.

The calculated NBR_i and MBR_i are accumulated over time to the NNB_i and NMB_i respectively by (13). The modified Beta reputation equation (14) computes the trust value for each node by using the accumulated behaviors, which are the NNB_i and

NMB_i . This research maintains the NNB_i , NMB_i , and trust values for each type of behaviors; host-degree (HDT_i), port-degree (PDT_i), clustering coefficient (CCT_i), and betweenness centrality (BCT_i).

By employing the modified Beta reputation, we can convert the different ranges of node characterization information between 0 and 1 that represent unreliable and reliable respectively. Besides, the method provides a low time complexity. The outlier-based adaptive threshold requires $O(n \log n)$ for a sorting algorithm (i.e. Merge Sort) [91] and $O(n)$ for the evaluation from the highest value to the lowest value that we discussed in Section 2.5, where the n represents the number of unique values. Also, the average-based adaptive threshold requires $O(m)$ for the addition of all the values with one division, and the Beta Reputation requires $O(m)$ for the thresholds comparisons with the trust calculations that was discussed in this section, where the m represents the number of nodes. Thus, the total complexity of the modified Beta reputation is $O(n \log n) + O(m)$. The accumulative evaluation feature also saves the storage because this method requires the space complexity $O(m)$ for each trust type. Moreover, it provides a method to avoid erroneously labeling nodes with a few malicious behaviors because only one malicious behavior does not lower the trust value at once, and the successive normal behaviors can recover the trust. Since the different ranges of the node characterization information were converted into a same range, it is also possible to integrate multiple characteristics into a single value, so the integrated trust can efficiently find malicious nodes that attempt different types of attacks alternatively as we discussed in Section 2.4.1.

3.3 Predictability Trust

In addition to the types of trust in Section 3.2, this research employed Predictability Trust (*PRT*) for ADS systems to evaluate a node with multiple statistical characteristics and to provide defense against On-off attacks. First of all, an attack node might attempt different attacks each day, so it would be difficult to detect the attack node if each trust value works independently. The *PRT* is not only able to consider a single type of behaviors but also a combination of multiple types depending on the design of systems. In this dissertation, the *PRT* examined the host-degree, port-degree, clustering coefficient, and betweenness centrality together to detect any type of alternative hostscan, portscan, hybrid-scan, or any types of attacks that do not respect the community's boundaries. We also note that this research employed two kinds of threshold as we discussed in Section 3.2, and we cannot be sure whether a node is normal or malicious if the behavior is greater than the *MinTh* and smaller than the *MaxTh* while it is clear the node is normal if the behavior is smaller than the *MinTh*, and the node is malicious if the behavior is greater than then *MaxTh*. Thus, we decided not to consider the unclear behaviors for the *PRT*, and they will not affect to either the *NPB* or *MPB*.

$$PRT := \frac{NPB + 1}{NPB + MPB + 2} \quad (15)$$

The *PRT* will be computed by (5), and the *NPB* and *MPB* will be set by the following conditions. The pseudocode is shown in APPENDIX I.A.

- If all the behaviors placed between the *MaxTh* and *MinTh*, neither the *NPB* nor *MPB* will be increased.
- If any one of the behaviors exceeded the *MaxTh*, the *MPB* will be increased, but not *NPB*.
- If any one of the behaviors did not exceed the *MinTh*, and the other behaviors did not exceed the *MaxTh*, the *NPB* will be increased, but not *MPB*.

Secondly, as we discussed in Section 2.4.3, the *PRT* provides a cautious evaluation method that determines whether the behaviors of a node are unintentional temporary errors or a part of On-off attacks. The *PRT* with Sliding windows (*SW*) is a method that detects On-off attacks, which aim at disturbing trust management schemes [66], [68]. As shown in the following examples, the Beta reputation, as well as the modified Beta reputation, can be easily disturbed by an On-off attack. We evaluated the Beta reputation with four different types of behavior patterns shown in Table 2. Each *T* and *F* represent a normal and malicious behavior respectively.

	1	2	3	4	5	6	7	8	9	10	11	12	...	50
(a) 1 Time Error	F	T	T	T	T	T	T	T	T	T	T	T	...	T
(b) Close-set	F	F	F	F	F	F	F	F	F	F	T	T	...	T
(c) Evenly Distributed	T	T	T	T	F	T	T	T	T	F	T	T	...	F

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	T	F
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
T	F	T	T	F	T	T	T	T	T	T	T	F	T	T	T	T
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	
F	T	T	T	F	F	T	T	T	F	F	T	T	T	T	T	

(d) Random

Table 2. The patterns of on-off attacks

The behavior type (a) assumes that a node showed a temporary error, and the node can be classified as malicious with the only mistake as we discussed in Section 2.6.2. It may increase the false positive rate of an ADS, and our trust management scheme can avoid this situation. The behavior type (b), (c), and (d) are the possible *4G-IB* On-off attack patterns, and (c) and (d) were designed to recover the trust enough and attempt other attacks on the system. The Beta reputation can be disturbed by these types of behavior patterns due to the behavior-based redemption feature (Section 2.4). Moreover, some random behavior patterns, such a pattern shown in Table 2.(d), can also disturb the Beta reputation method.

(a) One-time error

(b) Close-set attacks

(c) Evenly distributed On-off attacks

(d) Random On-off attacks

Figure 15. The changes of the Beta reputation depending on the behavior patterns

Figure 15 shows the changes of the Beta reputation values depending on the behavior patterns in Table 2. The X-axis and Y-axis represent the behavior number and the corresponding trust value respectively. The Beta reputation recovered the trust value in the one-time error type due to the behavior-based redemption feature and the successive normal behaviors as shown in Figure 15.(a), and it showed that a node with

one mistake does not have very low trust value as expect. On the other hand, close-set attack type showed a very low trust value after the 10 malicious behaviors as shown in Figure 15.(b), and it might be blocked if the system has a trust threshold 0.3. On the other hand, Figure 15.(c) and (d) maintained high trust value, even they showed the same number of malicious behaviors to the close-set type, and these cases showed the vulnerability of the Beta reputation. The two pattern types could disturb the Beta reputation with well distributed normal and malicious behaviors.

Calculating the *PTR*, (5) also requires the sizes for the Normal Behavior Window (*NBW*) and Malicious Behavior Window (*MBW*). To show how the *PRT* provides a defense against On-off attacks in Table 2, we set the size of the *NBW* to 5 that can observe up to five previous behaviors, and the *MBW* has two sizes; the minimum size was set to 5, and the maximum size was set to 50. These *NBW* and *MBW* combination can detect up to $4N-1M$ On-off attacks [68].

(a) One-time error

(b) Close-set attacks

(c) Evenly distributed On-off attacks

(d) Random On-off attacks

Figure 16. The changes of the PRT depending on the behavior patterns

The *PRT* showed similar results to the Beta reputation in Figure 16.(a) and (b). In Figure 16.(a), it recovered the trust value for the one-time error type but slower than the Beta reputation due to the one malicious behavior. The figure also showed that the one malicious behavior does not much lower the trust value. Figure 16.(b) showed that the many successive malicious behaviors lowers the trust similar to the Beta

reputation. On the other hand, the *PRT* showed different changes for the evenly distributed and random On-off attacks., and they showed how the *PRT* provided defense against On-off attacks. The Beta reputation was disturbed by the alternative normal-malicious behaviors while the *PRT* could successfully lower the trust values of the On-off attack patterns as shown in Figure 16.(c) and (d). Since the *PRT* extends the size of the dynamic sliding window depending on the current trust, it was able to observe more number of malicious behaviors and lowered the trust values.

By employing the *PRT*, we were able to consider multiple node characterization information together and provide defense against On-off attacks with low time and space complexity. The *PRT* requires $O(M \times N)$ for traveling the maximum sliding window size M of N nodes, and $O(N)$ for the trust calculation. Thus, the total time complexity of the *PRT* is $O(M \times N)$. Besides, the sliding windows can save more space if they were implemented in bit-wise operation because the *PRT* considers only normal (*TRUE*) or malicious (*FALSE*). Thus, each sliding window for each node requires $M/8$, and the space complexity of the *PRT* is $O(M \times N)$.

3.4 Overall trust

We defined the Overall Trust (*OT*) value as a compilation of all of the other trust values to evaluate if a node is malicious or not. The *OT* provides concise information to a system or a human supervisor by considering overall characteristics of each node and makes it possible to take appropriate actions based on the information. This dissertation used the *OT* to integrate multiple trust values, including the HDT_i , PDT_i , CCT_i , BCT_i , and PRT_i , into one. There exist many other ways that can implement this

type of trust. For example, Chae et al. proposed an *OT* that each trust has own weight that can contribute to the *OT* in [68].

$$\begin{aligned} \text{Overall Trust} &= \sum_{i=1}^n T_i \cdot W_i \\ \sum_{i=1}^n W_i &= 1 \end{aligned} \tag{16}$$

,where the T_i represents a type of Trust, and the W_i represents the weight of the Trust in the Overall Trust. Other than this method, the average of the trust values, maximum trust value, or minimum trust value can be used for implementing the *OT*. Among the various methods, we decided to use the minimum trust value in this research to quickly detect the malicious node.

3.5 Summary

Chapter 3 proposed a statistical analysis-based ADS. However, the results of the statistical analysis are not intuitive and require statistical background to understand the numbers. Moreover, it is not possible to integrate multiple behavior information into one because each analysis method produces different ranges of numbers (Section 3.1). This dissertation proposed a trust management scheme as a solution that can convert each characteristic between 0 (unreliable) and 1 (reliable), so a human supervisor can understand the results without the related statistical background. The Beta reputation (Section 2.4.2) was modified for creating appropriate rules for decision-making about network node behaviors in a dynamic network environment by using an adaptive threshold (Section 2.5), and the modified Beta reputation was proposed in Section 3.2.

Besides, the trust makes it possible to integrate multiple behavior information into one because they will be presented in the same ranges. We used two methods to integrate the trust values. The Predictability trust was used for detecting malicious behavior patterns across the characteristics (Section 3.3), and the Overall trust represents multiple trust values in one single number (Section 3.4). In next chapter, we will show how much the graph can reduce the network flows, how well the trust management scheme represents the statistical analysis, and how well the proposed ADS detects malicious behaviors.

CHAPTER 4

RESULTS AND ANALYSIS

Chapter 4 evaluates the efficiency of the proposed graph-based representation, trust management scheme, and adaptive threshold methods, and how the methods achieved the goals stated in Chapter 1 by using the European dataset, DShield Logs, and URI dataset. Section 4.1 shows how much the graph-based representation reduced the datasets to be statistically analyzed. Section 4.2 describes how the trust management can represent the results of the statistical analysis appropriately, and Section 4.3 shows the effectiveness of the adaptive threshold by comparing a few static thresholds. This dissertation presents the detection results of the scanning attacks in the European dataset and URI datasets by using the proposed methods in Section 4.4. Section 4.5 shows how the trust management scheme works in a few special cases. Finally, Section 4.5 summarizes this chapter.

4.1 Graph-based representation of network flows

This section shows the amount of data that can be reduced by using graph-based presentation. As discussed in Section 3.2, one of the advantages of using graph-based analysis is that the proposed methods can reduce the amount of dataset to be analyzed because many duplicate relationships between two nodes can be presented as one edge in a graph (Section 2.2.2).

For instance,

Source node	Destination node
...	...
1.1.1.1	1.1.1.2
<i>1.1.1.1</i>	<i>1.1.1.3</i>
<i>1.1.1.1</i>	<i>1.1.1.3</i>
<i>1.1.1.1</i>	<i>1.1.1.3</i>
1.1.1.1	1.1.1.4
...	...

can be reduced by the graph presentation like the following.

Source node	Destination node
...	...
1.1.1.1	1.1.1.2
<i>1.1.1.1</i>	<i>1.1.1.3</i>
1.1.1.1	1.1.1.4
...	...

In a large dataset, this method can reduce a huge amount of data by removing duplicate relationships and can save the processing and storage resources required for statistical analysis. Since this dissertation mainly focuses on detecting three types of scanning attacks (Section 2.6.1) and the nodes that do not respect community boundaries (Section 2.3.4, 2.4.4, and 2.4.5), we used the parts of network flows that are not compressed, encrypted, or fragmented, namely the IP address and port numbers. The following figures show how much the graph representation can reduce the datasets.

Figure 17. Comparisons of the amount of IP addresses to be analyzed between the original and graph presented in the European dataset

Figure 18. Comparisons of the amount of ports to be analyzed between the original and graph presented in the European dataset

Figure 19. Comparisons of the amount of IP addresses to be analyzed between the original and graph represented in the URI dataset

Figure 20. Comparisons of the amount of ports to be analyzed between the original and graph represented in the URI dataset

Figure 17 and Figure 18 show the amount of reduced data to be analyzed when the graph representation is applied to the European dataset. By using graphs the original network flow datasets were reduced by at least 81.1% and up to 86.9% for the IP address (Figure 17) and at least 77.9% and up to 80.8% for the ports (Figure 18). The detailed comparisons are described in APPENDIX II.A. Additionally, Figure 19 and Figure 20 show the amount of reduced data to be analyzed for the URI dataset. We could reduce the original URI network flow dataset by at least 31.1% and up to 54.3% for the IP addresses and at least 63.6% and up to 85.3% for the port. The detailed numbers are described in APPENDIX II.B.

4.2 Relationship between statistical analysis and trust management scheme

Since malicious behaviors could have been caused by unintentional temporary errors or by attackers who have used fake identification (Section 2.6.2), this dissertation is more interested in the patterns of the malicious behaviors rather than each malicious behavior of the nodes in a system. Moreover, it is difficult to be sure that a node is an attacker by observing a few malicious behaviors. Thus, monitoring systems need to be cautious before blocking a node from accessing the systems because many normal users can be affected, which is called False Positive. In this work, we employed a trust management scheme to quantify the patterns, and we will show how well the trust management scheme can represent the patterns of the malicious behaviors in this section.

4.1.1 Experiment design

In what follows, we analyzed the host-degree of the European datasets (Section 1.3) that have been used for community-based statistical analysis in [6]. The authors mentioned that the malicious sources often send traffic to many other nodes, and so candidates for detection are with high host-degree. Therefore, this experiment focuses on the host-degree analysis and shows how the trust management scheme can represent the degree-based analysis method over time. We label a node as malicious if any node shows higher host-degree than a critical threshold. The threshold values are used for creating ROC curves that can show the efficiency of each detection method. This experiment compares the efficiency of different behavior patterns to the trust management scheme by using the ROC curves and the corresponding AUCs.

4.1.1.1 Degree-based pattern analysis methods

Here, we apply three possible accumulative over time patterns of malicious behaviors: *Everyday*, *Average Degree*, and *At Least Once*.

- *Everyday*: Node is labeled as malicious if the node showed higher host-degrees than the threshold every day. The procedures are shown in APPENDIX I.B.
- *Average Degree*: Node is labeled as malicious if the average host-degrees of the nodes during the 14 days is greater than the threshold. The procedures are shown in APPENDIX I.C.
- *At Least Once*: Node is labeled as malicious if the node showed higher host-degrees than the threshold at least once during the 14 days. The procedures are shown in APPENDIX I.D.

It is worth noting that the *At Least Once* method is the most stringent, the *Everyday* method is the least stringent, and the *Average Degree* method is a moderately stringent monitoring strategy.

The ROC curves for each pattern were created by using the pseudocodes in APPENDIX I.B, C, D based on the above conditions with the host-degree thresholds from 1 (the minimum host-degree during the 14 days) to 6455 (the maximum host-degree during the 14 days). Each day, we classified a node as malicious, if a node showed a higher host-degree than the host-degree threshold. After 14 days, we classified a node as malicious if a node was malicious everyday according to Everyday accumulation pattern, or at least once during 14 days according to At Least Once pattern, or if its average host-degree was above average according to Average Degree pattern. After classifying each node as malicious or not with the specified method, we determined if the node was present in the DShield Logs to compute the TPR, FPR, and ROC curves. Please note that the DShield Logs are not a perfect representation of all malicious nodes, and our methods can only detect the malicious behaviors based on host-degree information, such as hostscan or a hybrid-scan (Section 2.6.1). Therefore, the ROC curves cannot represent the absolute efficiency of each method but can be used for comparing the efficiency of different methods.

We computed each ROC curve by checking that the detected malicious nodes based on the three methods are present in the DShield Logs or not. When a detected malicious node is in the DShield Logs, it increases the number of True Positives (*TP*) by one while if the node is not on the DShield Logs, it increases the number of False

Positives (FP) by one. When a node is in the DShield Logs but is not classified as a malicious node, it increases the number of False Negative (FN) by one while if the node is not listed in the DShield Logs, it increases the number of True Negative (TN) by one. The True Positive Rate (TPR) and False Positive Rate (FPR) are computed using (17) and (18).

$$TPR = \frac{TP}{TP + FN} \quad (17)$$

$$FPR = \frac{FP}{TN + FP} \quad (18)$$

The TPR and FPR computed for each host-degree threshold are used for creating a ROC curve. Each ROC curve is used for computing the corresponding Area Under a ROC Curve (AUC) that is a common evaluation metric for binary classification problems. If the classifier is accurate enough, the TPR increases quickly and the AUC is close to 1.

4.1.1.2 Trust-based pattern analysis method

In this section, we describe how the ROC curves are produced for Trust-based detection method. Specifically, we use the following equations that are based on the Beta reputation [5] that were described in Section 2.4.2. For each day over the 14 days period, we compute the host degree of each node. If a node shows behavior with the host-degree higher than the host-degree threshold, we classify the node's behavior as negative. Otherwise we classify its behavior as positive. The rate of positive behaviors and the rate of negative behaviors for each node we accumulated over the 14 days. The accumulated values were used to calculate the trust values for each node.

$$B_i^p(t) := B_i^p(t-1) + \begin{cases} 1 & , \text{Behavior} < \text{Degree threshold} \\ 0 & , \text{Behavior} \geq \text{Degree threshold} \end{cases} \quad (19)$$

$$B_i^n(t) := B_i^n(t-1) + \begin{cases} 1 & , \text{Behavior} \geq \text{Degree threshold} \\ 0 & , \text{Behavior} < \text{Degree threshold} \end{cases} \quad (20)$$

$$T_i(t) := \frac{B_i^p(t) + 1}{B_i^p(t) + B_i^n(t) + 2} \quad (21)$$

In Equations (19), (20), and (21), t represents the number of days over which the trust is accumulated, T_i represents the trust value of node i , B_i^p represents the number of positive behaviors of node i , and B_i^n represents the number of negative behaviors of node i . The detailed procedures are shown in APPENDIX I.E.

The trust values calculated using Equations (19), (20), and (21) with host-degree thresholds from 1 (the minimum host-degree during the 14 days) to 6455 (the maximum host-degree during the 14 days), were utilized to produce the ROC curves. In order to create the ROC curve, we had to define a way to use trust to determine if a node is malicious. To do this, we considered the trust value after day 14 and selected a trust value under which we would label the node as malicious, the trust threshold. We constructed nine different ROC curves using trust thresholds ranging from 0.1 to 0.9 with 0.1 intervals. We then compare the detected malicious nodes based on the trust to the DShield Logs and compute the corresponding TP , FP , FN , and TN . These numbers are used for computing each True Positive Rate (TPR) and False Positive

Rate (*FPR*) by using (17) and (18), and to create the ROC curves and the corresponding *AUCs* depending on each trust threshold.

4.1.2 Experimental results

Figure 21 shows the ROC curves for three simple accumulative methods (i.e., *Everyday*, *Average Degree*, and *At Least Once*) and the trust-based method. Among the three malicious patterns, the *At Least Once* method showed the highest detection efficiency, and it means that the method showed the detection efficiency close to the DShield Logs. In other words, the DShield Logs also classified the malicious nodes with one or a few behaviors. However, there is a possibility that the malicious behaviors have been caused by unintentional temporary error or an attacker has used a fake identification, so ADSs should not classify a malicious node with a few malicious behaviors as we discussed in Section 2.6.2. In other words, detecting a malicious node with a few malicious behaviors cannot be the best practice in an ADS because it increases False Positive, which is one of the disadvantages of the ADSs.

Figure 21. Relationships between the degree-based analysis and trust-based analysis.

One interesting observation we made was that two lines in each analysis were overlapped, which are the *Everyday* malicious and trust threshold 0.1 pairs and the *At Least Once* malicious and trust threshold 0.9 pairs. Moreover, the trust threshold 0.7 showed very similar line to the *Average Degree* malicious. We explain these results below.

- ***Everyday* and 0.1 pairs:** If a node showed malicious behaviors every day during the 14 days, the trust of the node would be $(0 + 1)/(0 + 14 + 2) = 0.0625$. If the node showed only one normal behavior during the 14 days, the trust of the node would be $(1 + 1)/(1 + 13 + 2) = 0.125$. Thus, when we set the trust threshold to 0.1, every node that showed malicious behaviors every day during the 14 days will

be classified as malicious. This is why these two graphs showed the identical results.

- ***At Least Once and 0.9 pairs:*** If a node showed normal behaviors every day during the 14 days, the node would have its trust value $(14 + 1)/(14 + 0 + 2) = 0.9375$ while if the node showed only one malicious behavior during the 14 days, the trust value would be $(13 + 1)/(13 + 1 + 2) = 0.875$. Thus, when we set the trust threshold to 0.9, every node that showed malicious behavior at least once during the 14 days will be classified as malicious. Therefore, these two graphs showed the same results.

From the *Everyday and 0.1 pairs* and the *At Least Once and 0.9 pairs*, we can deduce following equation that decides the corresponding trust threshold.

$$T_{th} = \frac{k + 1}{d + 2} \quad (22)$$

, where the trust threshold T_{th} can represent the behavior pattern with at least k malicious behaviors over d days. These results showed we can represent the Everyday and At Least Once accumulative methods with the trust method simply by changing the trust threshold. More generally, the trust-based analysis can provide flexibility for system designers who can vary of strictness of the monitoring system by choosing the trust thresholds.

4.3 Static and adaptive thresholds

In Section 4.2 we have shown that the trust management scheme can be used to represent various malicious patterns of behavior and provide a flexible design for the system designers. The trust management scheme depends on a threshold that is used to evaluate a behavior and to determine whether it is normal (reliable) or malicious (unreliable). In this section, we present the modified trust calculation, which we proposed in Section 3.2, and demonstrate that using an adaptive threshold is more efficient than using static thresholds in a dynamic environment.

4.3.1 Experiment design

In what follows, we compare the use of the adaptive threshold to five different static thresholds with ROC curves and AUCs. We chose a constant value of 2 for the first static threshold and show that if the threshold is too low, the rate of false positive can be excessive. Other three values of the thresholds are determined based on the average, which is the most popular and well-known measure of central tendency [90]; the average degree of all the nodes, average degree of the nodes that are not listed in the DShield Logs, and the average degree of the nodes that were listed in the DShield Logs. The last static threshold was determined based on the True Positive Rate (TPR) of the ROC curves during the 14 days. We analyzed the ROC curves and chose the value that ensured above 70% TPR over the 14 days.

4.3.1.1 Static thresholds for the host-degree analysis

Day	1	2	3	4	5	6	7	8
Not In	7.28	6.82	6.92	7.14	8.91	8.83	6.22	6.23
In	120.61	132.07	121.28	129.7	121.83	108.95	108.67	127.61
All	8.55	8.32	8.23	8.53	10.96	10.65	7.53	7.64
Day	9	10	11	12	13	14	Average	
Not In	6.48	6.71	6.79	9.14	9.08	6.35	≈ 7.35	
In	128.9	149.1	150.33	118.87	145.67	211.06	≈ 133.9	
All	7.94	8.2	8.27	11.02	11.13	7.79	≈ 8.91	

Table 3. Average host-degree on each day

Table 3 shows the average host-degrees on each day and the average host-degrees over the 14 days for each category (i.e., All nodes, nodes In DShield list, nodes Not In DShield list). From the results above, we determined three static thresholds: 7 - the average host-degree of the nodes not listed in the DShield Logs; 9 - the average host-degree of all nodes, and 134 - the average host-degrees of the nodes listed in the DShield Logs. Since a host-degree is a number of connections, we have rounded the averages in Table 3. The static threshold of 134 is used to illustrate the risk of choosing a threshold that is too high. In this case, the detection rate is low., We have observes that the malicious sources often send traffic to many other nodes and hence have a high out-degree.

Threshold	1	2	3	4	5	6	7	...
Day-1	1	0.95	0.91	0.85	0.78	0.72	0.67	...
Day-2	1	0.93	0.89	0.84	0.79	0.75	0.71	...
Day-3	1	0.95	0.9	0.86	0.84	0.79	0.75	...
Day-4	1	0.93	0.89	0.84	0.79	0.76	0.71	...
Day-5	1	0.94	0.87	0.8	0.77	0.74	0.69	...
Day-6	1	0.93	0.88	0.85	0.79	0.74	0.69	...
Day-7	1	0.95	0.88	0.8	0.73	0.67	0.65	...
Day-8	1	0.93	0.87	0.81	0.77	0.73	0.69	...
Day-9	1	0.94	0.89	0.83	0.77	0.7	0.67	...
Day-10	1	0.93	0.83	0.79	0.74	0.67	0.64	...
Day-11	1	0.92	0.86	0.81	0.75	0.72	0.69	...
Day-12	1	0.95	0.88	0.82	0.76	0.73	0.67	...
Day-13	1	0.92	0.85	0.79	0.74	0.7	0.66	...
Day-14	1	0.94	0.9	0.86	0.83	0.79	0.76	...

Table 4. TPR of the host-degree based analysis on each degree threshold

Threshold	1	2	3	4	5	6	7	...
Day-1	1	0.5	0.34	0.25	0.21	0.18	0.15	...
Day-2	1	0.5	0.32	0.24	0.19	0.16	0.14	...
Day-3	1	0.5	0.33	0.25	0.2	0.17	0.14	...
Day-4	1	0.5	0.33	0.25	0.2	0.17	0.15	...
Day-5	1	0.57	0.41	0.32	0.27	0.23	0.21	...
Day-6	1	0.56	0.4	0.31	0.26	0.22	0.19	...
Day-7	1	0.49	0.32	0.24	0.19	0.16	0.13	...
Day-8	1	0.49	0.32	0.23	0.18	0.16	0.13	...
Day-9	1	0.49	0.32	0.24	0.19	0.16	0.14	...
Day-10	1	0.49	0.32	0.24	0.19	0.16	0.14	...
Day-11	1	0.5	0.33	0.24	0.2	0.17	0.15	...
Day-12	1	0.56	0.41	0.32	0.27	0.23	0.2	...
Day-13	1	0.57	0.4	0.31	0.26	0.22	0.19	...
Day-14	1	0.49	0.31	0.24	0.19	0.16	0.14	...

Table 5. FPR of the host-degree based analysis on each degree threshold

By analyzing the ROC curves over the 14 days, we select another static threshold. Based on the TPR and FPR calculated using the host-degree threshold from 1 to 6455 and summarized in Table 4 and Table 5, we choose the host-degree threshold of 5 that ensured the TPR above 70% over the 14 days. The shaded part of Table 4 shows that the TPR was at least 73% while the FPR was at best 26% during the 14 days as shown in the shaded part of Table 5.

In conclusion, we chose the five static thresholds for the host-degree based analysis; 7, 9, and 134 based on the averages, 5 based on the TPR analysis, and a constant threshold of 2.

4.3.1.2 Static thresholds for the port-degree analysis

Day	1	2	3	4	5	6	7	8
Not In	10.82	10.88	10.84	11.5	16.59	16.42	10.4	9.65
In	125.67	129.11	115.2	125.54	121.88	122.17	98.27	108.5
All	12.1	12.29	12.04	12.8	18.5	18.35	11.52	10.8
Day	9	10	11	12	13	14	Average	
Not In	9.71	10.21	10.72	16.37	15.28	9.34	≈ 12.05	
In	113.85	133.78	126	101.97	127.9	209.59	≈ 125.67	
All	10.95	11.51	11.91	17.84	16.97	10.76	≈ 13.45	

Table 6. Average port-degree on each day

Table 6 shows the average port-degrees on each day and the average port-degrees over the 14 days for each category of nodes category (All nodes, nodes In DShield list, nodes Not In DShield list). From these results, we determine three static thresholds similarly to Section 4.3.1.1. Specifically, we fix threshold values at 12, 13, and 126, the average port-degree of the nodes not listed in the DShield Logs, the average port-degree of all the nodes, and the average port-degree of the nodes listed in the DShield Logs respectively. Since a port-degree is a number of, we rounded the averages. The static threshold 126 was also selected to illustrate the risk risk of choosing a threshold that is too high.

Threshold	1	2	3	4	5	6	7	...
Day-1	1	0.94	0.89	0.82	0.77	0.71	0.67	...
Day-2	1	0.91	0.87	0.82	0.78	0.75	0.71	...
Day-3	1	0.94	0.88	0.84	0.82	0.79	0.75	...
Day-4	1	0.92	0.88	0.83	0.79	0.76	0.73	...
Day-5	1	0.93	0.85	0.78	0.75	0.71	0.68	...
Day-6	1	0.93	0.87	0.83	0.78	0.72	0.69	...
Day-7	1	0.93	0.87	0.79	0.73	0.68	0.65	...
Day-8	1	0.91	0.86	0.81	0.76	0.73	0.69	...
Day-9	1	0.93	0.87	0.81	0.75	0.69	0.66	...
Day-10	1	0.92	0.84	0.79	0.74	0.67	0.64	...
Day-11	1	0.91	0.85	0.82	0.75	0.71	0.69	...
Day-12	1	0.94	0.87	0.81	0.77	0.73	0.67	...
Day-13	1	0.92	0.85	0.78	0.74	0.7	0.68	...
Day-14	1	0.92	0.88	0.83	0.79	0.76	0.73	...

Table 7. TPR of the port-degree based analysis on each degree threshold

Threshold	1	2	3	4	5	6	7	...
Day-1	1	0.4	0.29	0.24	0.2	0.18	0.16	...
Day-2	1	0.39	0.27	0.22	0.19	0.17	0.16	...
Day-3	1	0.38	0.27	0.22	0.19	0.17	0.15	...
Day-4	1	0.38	0.27	0.22	0.19	0.17	0.15	...
Day-5	1	0.51	0.38	0.32	0.27	0.24	0.22	...
Day-6	1	0.52	0.39	0.32	0.28	0.25	0.22	...
Day-7	1	0.38	0.27	0.22	0.19	0.16	0.15	...
Day-8	1	0.37	0.26	0.2	0.17	0.15	0.14	...
Day-9	1	0.37	0.26	0.21	0.18	0.15	0.14	...
Day-10	1	0.37	0.26	0.21	0.18	0.16	0.14	...
Day-11	1	0.38	0.27	0.22	0.18	0.16	0.15	...
Day-12	1	0.5	0.37	0.31	0.27	0.24	0.21	...
Day-13	1	0.51	0.37	0.31	0.26	0.23	0.21	...
Day-14	1	0.36	0.25	0.2	0.17	0.15	0.14	...

Table 8. FPR of the port-degree based analysis on each degree threshold

Another static threshold has been chosen by analyzing the ROC curves during the 14 days. The TPR and FPR have been calculated using the port-degree threshold from 1 (the minimum port-degree during the 14 days) to 14186 (the maximum port-degree during the 14 days) and summarized in Table 7 and Table 8. Since the port-degree threshold 5 ensures above 70% TPR over the 14 days, we select the last static threshold as 5. The shaded part of Table 7 shows that the TPR is at least 73% while the FPR is at best 28% during the 14 days.

Thus, we select the five static thresholds for the host-degree based analysis; 2, 5, 12, 13, and 126.

4.3.1.3 Adaptive thresholds for the host-degree and port-degree

Figure 22. European dataset log-scaled host-degree histogram

Figure 22 shows the host-degree distribution inferred the European network flows, and it clearly displays heavy-tailed distribution. Thus, it is reasonable to apply the outlier-based adaptive threshold method introduced in Section 2.5. We calculate the maximum and minimum thresholds for the trust calculation using Equations (8), (9), and (10) by setting up $\delta = 0.05$ and $\kappa = \text{Number of data}/4$ respectively. The detailed procedures are described in APPENDIX I.F. The minimum outlier of the host-degree distribution is set to the *MaxTh*. The *MinTh* is computed as the average of the host-degrees without the outliers (Section 3.2).

4.3.1.4 ROC curves and AUC

In what follows, we produce the ROC curves and the corresponding AUCs for each selected threshold and compare the efficiency of the malicious node detection algorithms using the adaptive threshold versus using the fixed (static) thresholds. We use the Beta reputation, introduced in Section 2.4.2, to calculate the trust values for the static thresholds and we use the modified Beta reputation, proposed in Section 3.2, for the adaptive threshold method. The latter method incorporates two thresholds, *MinTh* and *MaxTh*. Since this experiment is designed for comparing the efficiency of the adaptive threshold to the static threshold, we minimize the influence of the trust management scheme by not accumulating the trust values over the 14 days, but detect the malicious nodes based on the trust value daily.

$$NBR_i := \begin{cases} 1 & , BV < StaticTh \\ 0 & , BV \geq StaticTh \end{cases} \quad (23)$$

$$MBR_i := \begin{cases} 1 & , BV \geq StaticTh \\ 0 & , BV < StaticTh \end{cases} \quad (24)$$

$$T_i := \frac{NNB_i + 1}{NNB_i + NMB_i + 2} \quad (25)$$

The trust values of each node for the static thresholds (Section 4.3.1.1 and 4.3.1.2) were calculated by using Equations (23), (24), and (25) with each threshold. The Normal Behavior Rate of node i (NBR_i) is set to 1 if the Behavior (BV) of node i does not exceed the static threshold while the Malicious Behavior Rate of node i (MBR_i) is

set to 1 if the BV of node i exceeds the threshold. The Trust of node i (T_i) is calculated by using the NBR_i and MBR_i .

$$NBR_i := \begin{cases} 1 & , BV \leq MinTh \\ 0 & , BV \geq MaxTh \\ 1 - \frac{BV_i - MinTh}{MaxTh - MinTh} & , otherwise \end{cases} \quad (26)$$

$$MBR_i := \begin{cases} 1 & , BV \geq MaxTh \\ 0 & , BV \leq MinTh \\ \frac{BV_i - MinTh}{MaxTh - MinTh} & , otherwise \end{cases} \quad (27)$$

$$T_i := \frac{NNB_i + 1}{NNB_i + NMB_i + 2} \quad (28)$$

The trust values of each node for the adaptive threshold (Section 4.3.1.3) were calculated by using (26), (27), and (28). The NBR_i is set to 1 if the BV of node i is less than or equal to the minimum threshold while the MBR_i is set to 1 if the BV of node i is greater than or equal to the maximum threshold. When the BV of node i is placed between the two thresholds, the NBR_i and MBR_i will be set by the equations in (26) and (27). The Trust of node i (T_i) is computed using the NBR_i and MBR_i .

The calculated trust values for each threshold were evaluated and compared by using ROC curves and the corresponding AUCs. The TPR and FPR for the ROC curves were calculated with the trust threshold from 0.0 to 1.0 with 0.001 intervals.

4.3.2 Experimental results

In this section, we present the results of 2 days out of 14 days because they showed very similar results. The host-degree and port-degree were analyzed separately, and the ROC curves and AUC were used for the comparisons.

Figure 23. Host-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 1

Figure 24. Host-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 14

Figure 25. Host-degree analysis and the AUC during the 14 days (Line plots)

Figure 26. Host-degree analysis and the AUC during the 14 days (Box plots)

	TPR	FPR
Static-2	0.948171	0.504942
Static-5	0.783537	0.207009
Static-7	0.670732	0.154375
Static-9	0.615854	0.12538
Static-134	0.167683	0.00743
Adaptive	0.783537	0.207009

Table 9. The highest TPR of each threshold with the corresponding FPR in the Host-degree analysis on Day 1

	TPR	FPR
Static-2	0.942568	0.486698
Static-5	0.827703	0.187601
Static-7	0.756757	0.137017
Static-9	0.699324	0.109715
Static-134	0.260135	0.006452
Adaptive	0.858108	0.235536

Table 10. The highest TPR of each threshold with the corresponding FPR in the Host-degree analysis on Day 14

In Figure 23 and Figure 24, the results of the host-degree analysis are presented. Each ROC curve represents the efficiency of each threshold, and the points of the highest TPR and the corresponding FPR are presented in Table 9 and Table 10. The AUCs rank high in the order of using adaptive, static 5, static 7, static 9, static 2, and static 134 host-degree thresholds.

The results clearly show the risks of selecting the degree threshold which is too low or too high. The static threshold 2 results in the highest TPR while it also shows the highest FPR during the 14 days, implying that many normal nodes can be labeled as malicious by the security system that employs the low host-degree threshold. On the other hand, the static threshold 134 results in the lowest TPR and the lowest FPR during the 14 days, implying that only a small number of normal nodes can be labeled as malicious, but also many malicious nodes can be misclassified as normal, adversely affecting the system.

The static threshold of 5, which was the best choice among the static threshold deduced from the host-degree threshold-based analysis (Section 4.3.1.1), results in relatively stable results. The TPR is greater than for other static thresholds but the

static threshold of 2, and the FPR is much lower than the static threshold of 2. We observed that in general the TPR increases and the FPR decreases for higher static threshold values (see in Table 9 and Table 10).

The results obtained for the adaptive threshold are similar to the results of using the static threshold of 5 in terms of the TPR and the FPR on the first day and a bit better on the last day, and much better for other values of static thresholds for all days (see Figure 25 and Figure 26). The detailed TPRs and FPRs are listed in APPENDIX II.C.

Figure 27. Port-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 1

Figure 28. Port-degree analysis and the ROC curves for each static threshold and the adaptive threshold on Day 14

Figure 29. Port-degree analysis and the the AUC during the 14 days (Line plots)

Figure 30. Port-degree analysis and the AUC during the 14 days (Box plots)

	TPR	FPR
Static-2	0.935976	0.40396
Static-5	0.77439	0.202861
Static-12	0.557927	0.111695
Static-13	0.542683	0.10423
Static-126	0.140244	0.01296
Adaptive	0.77439	0.202861

Table 11. The highest TPR of each threshold with the corresponding FPR in the Port-degree analysis on Day 1

	TPR	FPR
Static-2	0.918919	0.364921
Static-5	0.790541	0.173059
Static-12	0.614865	0.0938
Static-13	0.594595	0.088407
Static-126	0.236486	0.010931
Adaptive	0.790541	0.173059

Table 12. The highest TPR of each threshold with the corresponding FPR in the Port-degree analysis on Day 14

The ROC curves for the port-degree analysis are presented in Figure 27 and Figure 28. The results are similar to the host-degree analysis; the adaptive threshold-based method is the most accurate, and the static threshold-based detection accuracy ranks in the order of threshold values of 5, 12, 13, 2, and 126. However, the host-degree analysis resulted in slightly better performance than the port-degree analysis.

Again, the results support the risks of choosing the threshold that is too low or too high. The static threshold of 2 shows the highest TPR with the highest FPR, thereby misclassifying many normal nodes as malicious. On the other hand, the static threshold of 126 results in the lowest TPR and the lowest FPR during the 14 days, thereby misclassifying many malicious nodes as normal.

The static threshold 5, which was the best choice of the static threshold deduced from the port-degree threshold-based analysis (Section 4.3.1.2), shows the competitive TPR and FPR compared to the other static thresholds. The relationships between the threshold, TPR, and FPR in the port-degree analysis is similar to the one observed for the host-degree analysis. The adaptive threshold-based method shows the best detection performance resulting in higher AUCs during the 14 days than the other

static thresholds (see Figure 29 and Figure 30.). The detailed TPR and FPR of the intersections are listed in APPENDIX II.D.

In conclusion, in this section we show that the trust management scheme with the adaptive threshold provides the superior detection performance compared to the trust management scheme with the static thresholds.

4.3.3 Static threshold for clustering coefficient

As we discussed in Section 2.5, the clustering coefficient does not follow the heavy-tailed distribution, so we cannot apply the adaptive threshold method for this analysis, and needed to set a static threshold. To find the best static threshold, we composed the same experiments described in Section 4.3.2.

Figure 31. Clustering coefficient analysis and the AUC during the 14 days (Line plots)

Figure 32. Clustering coefficient analysis and the AUC during the 14 days (Line plots)

Figure 31 and Figure 32 show that the results of the clustering coefficient-based malicious node detection. Each ROC curve represents the efficiency of each threshold. We have tested the static thresholds every 0.05 from 0.05 to 1.0. As shown in the figures, the AUC increases as the static threshold increased. Among the static threshold candidates, we select the 0.95 for the static threshold of clustering coefficients for further experiments.

4.4 Detecting attacks using trust management scheme

In previous sections, we showed that the trust management scheme could be used to accumulate node behaviors over time (Section 4.2), and in combination with the proposed adaptive threshold selection could provide superior detection performance

compared to static threshold selection (Section 4.3). In this Section, we analyze the European and URI datasets (Section 1.3) by using the trust management scheme, which includes the modified Beta reputation with the proposed adaptive threshold (Section 3.2), Predictability trust (Section 3.3), and Overall trust (Section 3.4).

This section focuses on three main analyses:

- **Daily malicious behavior analysis:** We have analyzed four different types of behaviors on each day using graph-based characteristics including: the host-degree, port-degree, betweenness centrality, and clustering coefficient discussed in Section 2.3. We evaluate the node behaviors using the adaptive thresholds (Section 2.5 and 3.2), and the modified Beta reputation for the host-degree, port-degree, and betweenness centrality, characteristics that follow heavy-tail distribution. We evaluate the node behaviors using the clustering coefficient with a static threshold 0.95 since it does not follow the heavy-tailed distribution as discussed in Section 4.3.
- **Multi-day malicious behavior analysis:** The main purpose of this analysis is to identify the persistent malicious nodes. Each node in a network is identified by an IP address, and the daily malicious behavior analysis is performed based on the IP address. However, there exist many methods in which an attacker can hide or change their IP addresses (Section 2.6.2). Thus, it could be dangerous classifying a malicious node with a few behaviors because it would affect the normal user who has used or will use the IP address. This dissertation claims that an ADS needs to be cautious to evaluate a node and should not classify a malicious node with a few malicious behaviors. It would be ideal if a secure

system could store and observe the entire behaviors of each node and analyze them, but it is unattainable due to the limited storage and processor speed. Therefore, we employ the modified Beta reputation that provides a cumulative behavior evaluation method (Section 3.2) and observe each node for several days. The trust values are used for detecting the malicious nodes that show several malicious behaviors over the observation period.

- **Aggregation of multiple trust values:** An attacker may perform more than one type of attack over the observation period. It would be difficult to detect a malicious node by analyzing only one of the behaviors if an attacker performed hostscan on the first observation day, portscan on the second day, and any other types of attacks on the following days. Thus, it is important for security systems to have an ability to aggregate the results of multiple analyses into one to detect such an attacker. This research proposed two different trust values for the aggregation of multiple trust values; the predictability trust (Section 3.3) and the overall trust (Section 3.4). Since each behavior may affect the calculation of the predictability trust as shown in Section 3.3, it can reflect the behavior pattern of the node. Also, we present four different types of aggregation method in Section 3.4 that can be implemented in other methods.

4.4.1 Experiment design

This dissertation focus is on the analyses the European and URI datasets and detection of the hostscan, portscan, hybrid-scan, and any types of attacks that do not respect the community boundaries by using the proposed methods in Chapter 3. The

reasons we focus on these two datasets are the following: 1) the European dataset contains the ground truth in a form of DShield Logs that can be used for validation, however 2) the European dataset is limited to 14-days of network flows and contains no node attributes, 3) the URI dataset contains 90-days of network flows with nodes that can be classified as URI clients, URI servers, and non-URI nodes, but it does not have a ground truth for evaluation.

4.4.1.1 Settings for the Predictability trust and Overall trust

As discussed in Section 2.4.3, the predictability trust was developed for detecting On-off patterns. Since there is a possibility that an attacker performs the hostscan, portscan, and the other types of attacks alternatively, the attacker may disturb the trust management schemes. We consider and integrate the trust of the host-degree, port-degree, betweenness centrality, and clustering coefficient for the predictability trust. However, we do not consider the behaviors when nodes are labeled as suspicious, having node characteristics between the *MaxTh* and *MinTh*. The pseudocode is shown in APPENDIX I.A. If one of the characteristics exceeds the *MaxTh*, we increase the *MPB* while all characteristics need to be lower than the *MinTh* to increase the *NPB*. By using the obtained *NPB* and *MPB*, we calculate the predictability trust values for each node with (5).

The predictability trust requires two more parameters, which are a minimum window size and a maximum window size, for the sliding windows. We set the parameters to 5 and 72 for the minimum and maximum window sizes, respectively. The parameters make the sliding windows for the Predictability Trust big enough to

detect up to $5N-1M$ On-off attacks [68]. The window size of the dynamic sliding windows can be calculated by equation (29). Thus, the dynamic sliding window can change the observation size depending on the current trust.

$$Size := -67 \times T_i + 72 \quad (29)$$

, where the T_i represents the current overall trust value of node i .

When a node is unreliable ($T_i = 0.0$), the size will be 72 while the node is fully reliable ($T_i = 1.0$), the size will be 5. If the node is suspicious ($T_i = 0.5$), the observation size will be 39. However, we need a large dataset to fully test the predictability trust because the European dataset has only 14 days. The dataset would never fill the dynamic sliding windows when it requires the maximum size observation, and we would not be able to observe long-term redemption, which allows a second chance to a malicious node based on a long-term observation.

The overall trust is defined to integrate the various trust values into one to quickly detect malicious behaviors of each node. There are many ways that one can implement the overall trust, such as an average of all the trust values, maximum trust value, or minimum value as discussed in Section 3.4. We use the minimum value method in this research for the quick detection of a malicious node. If any of the characteristics exhibit a malicious behavior, it will be directly projected to the overall trust.

4.4.1.2 European dataset analysis scenario

The European dataset has been analyzed through several steps as shown in Figure 33. In this section, we describe each step from the data collection to evaluation.

Figure 33. Data Analysis Scenario of the European dataset

A. Collect raw data: As described in Section 1.3, the dataset has been collected by European Internet Service Provider in 2007 for 14 days, and it was sampled 1:1000. The dataset was collected with DShield Logs that contain the IP addresses of malicious nodes.

B. Infer node characteristics from the network flows: In graph theory, the degree represents the number of edges from the vertex and to the vertex. In network analysis, it shows the number of connections from a source node to the destination nodes. This can be very valuable information that could indicate that a source node is an attacker or that a destination node has been attacked by an adversary that usually shows high degrees (i.e. DDoS, probing, and etc.) [6], [58]. In this dissertation, we used the host-degrees, port-degrees, betweenness centrality, and clustering coefficient to evaluate the behaviors of each node (Section 2.3). This stage infers node characteristic information from the European network flows datasets. The most intuitive way to identify super spreaders or attack nodes, such as hostscan or portscan attackers, is to count the number of flows generated by each host within a measurement period [6], [92]. We counted the number of destination IP addresses from each source node for the host-degree information and counted the number of destination ports from each source node for the port-degree information on each day. If the host-degree or port-degree of a node is greater than a threshold, there is a high possibility that the node was an attack node that has scanned multiple hosts or ports. Besides, as we described in Section 2.3, if nodes a and b share at least one common destination, they are connected in the bipartite network projection [58]. Ding et al. formed a projection graph to identify communities in the network and utilized it to find malicious nodes, which do not respect community boundaries, by using the betweenness centrality and clustering coefficient in [6]. This dissertation also employs the betweenness centrality and

clustering coefficient to identify the nodes that do not respect community boundaries, that are calculated from the projection graph by using the MatlabBGL package in Matlab [93].

C. Trust Calculation: This dissertation proposes and employs the modified Beta reputation with heavy-tailed outlier-based adaptive threshold for the trust computation (Section 3.3). By using the heavy-tailed outlier detection method, we set the *MaxTh* and *MinTh* to determine normal, malicious, and suspicious behaviors for the modified Beta reputation. By using the adaptive thresholds, we calculate the trust values for the host-degree, port-degree, and betweenness centrality for each node on each day. We use the static thresholds for clustering coefficient that does not follow the heavy-tailed distribution.

D. Evaluation: We search for malicious behaviors in the European dataset network flows during the 14 days based on the trust values. We select the sample malicious nodes in Section 4.4.2 by sorting the latest trust values in the ascending order and look for the attack signatures in Section 2.6.1.

E. Comparison: The European dataset uses the DShield Logs as the ground truth dataset. Using the proposed methods, we detect malicious nodes and check whether the detected nodes are listed in the DShield Logs. Through this step, we verify the proposed methods as well as the DShield Logs.

4.4.1.3 URI dataset analysis scenario

The University of Rhode Island (URI) Network Dataset has been analyzed through several steps as shown in Figure 34. In this section, we describe each step from the data collection to evaluation.

Figure 34. Data Analysis Scenario of the URI dataset

A. Collect raw data: As described in Section 1.3, the dataset has been collected by University of Rhode Island Information Technology Service (URI-ITS) for 90 days, and it sampled 1:100. This data does not contain the ground truth of attacks, but it does contain a list of URI servers that has been formed by performing port scanning on the network of University of Rhode Island (URI).

B. Clean up the network flows of DNS servers: A Domain Name System (DNS) server runs a service that translates more readily memorized domain names to the numerical identification of addressing component. For example, DNS servers provide responses to the queries of human-memorable domain names and hostnames with the corresponding numeric Internet Protocol (IP) addresses. Since most of the connections from the outside of the university access to the URI-DNS server to search the matching IP addresses, the server showed extremely high out-degree and in-degree. More specifically, the top hierarchy of the DNS is served by the root name servers maintained by delegation by the Internet Corporation for Assigned Names and Numbers (ICANN). Below the root, Internet resources are organized into a hierarchy of domains, administered by the respective registers and domain name holders [94]. If a name server does not have information that can answer the query it may recursively query name servers higher up in the hierarchy. Because of this nature of DNS servers, the servers generally have many destinations that is enough to distract the degree based statistical analysis. Thus, we have cleaned up the network flows of the URI-DNS servers in this research for a more accurate statistical analysis and evaluation.

C. Infer node characteristics from the network flows: As discussed in Section 4.4.1.1, the degree represents the number of edges from the vertex and to the vertex, and this information can be very valuable information for detecting attacks such as DDoS, probing, and etc. [6], [58]. We characterize each node by using the host-degrees, port-degrees, betweenness centrality, and clustering coefficient. This stage infers these node characteristics from the URI network flows datasets. Specifically, the host-degrees and port-degrees are computed by counting the number of connections from each source node from the bipartite graph, and the betweenness centrality and clustering coefficient that used for identifying the nodes that do not respect community boundaries are computed from the projection graph by using the MatlabBGL [93].

D. Separate the nodes into URI and non-URI: Since we are more interested in detecting attacks and/or malicious behaviors from/to the university network. Thus, we separate the dataset into two parts; URI and non-URI nodes based on the IP addresses that are assigned to the University of Rhode Island. For our convenience, we defined a few terms for the flows.

Figure 35. Definitions of terms of the network flows

- a. **URI nodes, From flows:** The source IP addresses belong to the URI Network and the destination IP addresses do not belong to the URI Network.
- b. **non-URI nodes, To flows:** The source IP addresses do not belong to the URI Network, and the destination IP addresses belong to the URI Network.

E. Separate the nodes into server and client: We have also separated the URI nodes into two parts; URI servers (or simply, servers) and URI clients (or simply, clients) because we assumed that server and client nodes have different host-degree and port-degree characteristics. To collect the information of the server and client nodes, we have composed a port scanning technique to the URI network and searched the service information on each URI nodes. This research expectation is that the server nodes would have different outliers from the client nodes group because the servers generally get connection requests rather than requesting connections. However, there are three facts that we need to consider. First, the port

scanning technique does not guarantee it detects all the open ports because some systems employ protection techniques that do not reveal the information. Second, even if any ports are opened on a node, it does not mean that the node is a server, but it can be used as a server. Third, all the scanned ports are not opened for providing services as a server. Thus, this dissertation considers the ports that provide popular services as shown in Table 13.

Service	Port Number
File Transfer Protocol (FTP)	21
Secure Shell (SSH)	22
Telnet remote login service	23
Simple Mail Transfer Protocol (SMTP)	25
Domain Name System (DNS) service	53
Hypertext Transfer Protocol (HTTP)	80
Post Office Protocol (POP3)	110
Internet Message Access Protocol (IMAP)	143
HTTP Secure (HTTPS)	443
SMTP Secure (SMTPS)	465

Table 13. List of the Well-Known Ports

F. Trust Calculation: We set the *MaxTh* and *MinTh* to determine normal, malicious, and suspicious behaviors for the modified Beta reputation by using the heavy-tailed outlier method and calculated the trust values for the host-degree, port-degree, clustering coefficient, and betweenness centrality for each node on each day.

G. Evaluation: We search malicious behaviors patterns in the URI network flows during the 90 days based on the trust values. We present examples of a few malicious nodes in Section 4.4 and Section 4.5 and summarize the results for the whole URI data in Section 4.6 and 4.7.

4.4.2 Experimental results

In this section, we will show how the trust management scheme can be used to identify the malicious nodes. The detected malicious nodes in the European dataset will be compared to the DShield Logs. We will verify the proposed methods as well as the DShield Logs. The URI dataset will show how the trust management scheme works on a large dataset including the long-term redemption. Since there does not exist ground truth of the attack nodes for the URI datasets, we rely on the collected raw data and human expert to verify the relationships between the trust and malicious behaviors. Therefore, we have built a database system that can retrieve the raw data by using SQL, and we will use the raw data to confirm the malicious behaviors.

By using the calculated trust values, we will take a second look at the behaviors and evaluate the trust values node by node. We illustrate behaviors of a few nodes similar to a hostscan, portscan, hybrid-scan, or other types of attacks. We also use for illustration the nodes that show the behaviors of one of the attacks but are not listed in the DShield Logs to demonstrate that these logs are not a perfect representation of all malicious nodes.

4.4.2.1 Normal nodes

(a). The host-degrees of a normal node

(b). The port-degrees of a normal node

(c). The betweenness centrality of a normal node

(d). The clustering coefficient of a normal node

(e). The trust changes of a normal node
Figure 36. A normal node that has not been classified as malicious

Figure 36 shows the behaviors and the corresponding trust changes of a normal node that is not listed in the DShield Logs, and the trust management scheme has classified this as a normal node. The host-degree, port-degree, and betweenness centrality never exceeded the *MinTh* as shown in Figure 36.(a), (b), and (c) respectively. Figure 36.(d) shows that the clustering coefficient also never went below the static threshold 0.95. This means that the node has executed neither hostscan nor portscan signatures/behaviors, and so it is expected to have the high *HT*, *PT*, *CCT*, *BCT*, *PRT*, and *OT* values as shown in Figure 36.(e).

4.4.2.2 Hostscan attacks

(a). The host-degrees of a hostscan attack node

(b). The port-degrees of a hostscan attack node

(c). The betweenness centrality of a hostscan attack node

(d). The clustering coefficient of a hostscan attack node

(e). The trust changes of a hostscan attack node
Figure 37. A hostscan attack node that has been classified as malicious

Figure 37 shows each behavior and the corresponding trust changes of a node that has been listed in the DShield Logs, and the trust management scheme classified as a hostscan attack node. The red lines in Figure 37.(a), (b), (c), and (d) represent the node's behaviors during the 14 days, and the trust values have been calculated based on the behaviors. We could observe the host-degrees exceeded the *MaxTh* (blue line) every day, the port-degrees slightly exceeded the *MinTh* (green line) except on the 5th day, the betweenness centrality exceeded the *MinTh* only once on the 4th day but never exceeded the *MaxTh*, and the clustering coefficient were below the static threshold 0.95 every day. Therefore, we can expect that the Host Trust (*HT*), Betweenness Centrality Trust (*BCT*), Clustering Coefficient Trust (*CCT*), Predictability Trust (*PRT*), and Overall Trust (*OT*) will be low while the Port Trust (*PT*) will be high. The blue line in Figure 37.(e) represents *PT* over the 14 days, and it showed the high trust

values while the *HT* (red line), *CCT* (green line), *BCT* (yellow line), *PRT* (purple line), and *OT* (cyan line) showed low values as we expected.

The hostscan attack nodes tend to show high host-degrees and low port-degrees as we described in Section 2.6.1.1. Therefore, we can specify the attack nodes that showed the hostscan patterns during the 14 days by searching the low *HT* and high *PT*. Besides, we could observe that the nodes with low *HT* also showed low *CCT* and low *BCT* because a node with high port-degrees would have high possibility to be connected to many communities. Thus, the hostscan attack node also showed a type of behaviors that do not respect the community boundaries (Section 2.3.4 and 2.3.5). We also were able to classify the node as malicious with the low *PRT* and *OT*. These results shows that the proposed trust management scheme could cautiously detect a hostscan attack node in a dynamic environment by decreasing each trust gradually and integrate them together.

(a). The host-degrees of a hostscan attack node in the non-URI flows

(b). The port-degrees of a hostscan attack node in the non-URI flows

(c). The betweenness centrality of a hostscan attack node in the non-URI flows

(d). The clustering coefficient of a hostscan attack node in the non-URI flows

(e). The trust changes of a hostscan attack node in the non-URI flows
Figure 38. A hostscan attack node that has been showed in the non-URI flows

We were also able to observe the same attack in the URI dataset. Figure 38 shows the behaviors and the corresponding trust changes of a non-URI node that was suspected as a hostscan attack node. The node also showed higher host-degrees than the *MaxTh* except for the 9 times on April 16th, 17th, and 18th and May 14th, 15th, 19th, 20th, 22nd, and 25th during the 90 days, so we could expect the low *HT*, *PRT*, and *OT* values. We also could expect the high *PT* values because the node tried to access to only 1 port every day. This node tried to access to the port 0 (invalid port number), which frequently has been used by attackers, on 5,268 different destination IP addresses that belong to URI network. The results also illustrate the need of Intrusion Detection System (IDS) because the firewall, which is the first line of a security system (Section 1.1), could not detect and allowed the hostscan attacks.

The betweenness centrality and clustering coefficient in Figure 38.(c) and (d) also showed malicious behaviors every day during the 90 days. The high betweenness centrality represents that the node worked as a bridge between the communities, which share the destination nodes with other nodes, and the node belonged to many communities. Besides, the low clustering coefficient illustrated that the node was not densely connected with the neighbors. Thus, the *BCT* and *CCT* showed low values in Figure 38.(e), and the values represented the node did not respect the community boundaries [6].

(a). The host-degrees of a hostscan attack node in the URI-server flows

(b). The port-degrees of a hostscan attack node in the URI- server flows

(c). The betweenness centrality of a hostscan attack node in the URI- server flows

(d). The clustering coefficient of a hostscan attack node in the URI- server flows

(e). The trust changes of a hostscan attack node in the URI- server flows

Figure 39. A hostscan attack node that has been showed in the URI- server flows

These results shows that the network flows might need to be analyzed separately depending on the location of nodes, such as URI and non-URI nodes. Figure 39 shows the behaviors and the corresponding trust changes of a node in the URI-server flows that showed a hostscan attack pattern. This node had 52 open ports and got connection requests to 66 different ports from 3,699 different source IP addresses. The most popular service was provided by the port 80 (HTTP), and 3,312 different source IP addresses tried to access to this port. Besides, the port 25 (SMTP) and 23 (TELNET) were accessed by 125 and 48 source IP addresses respectively. Thus, the node was a high probability of being a server.

The host-degrees of the node exceeded the *MaxTh* every day in Figure 39.(a), so we could expect low *HT* values with the low *PRT* and *OT* values. On the other hand, the port-degree exceed the *MinTh* only 18 times but never exceeded the *MaxTh* during the 90 days, so the high *PT* was expected. Therefore, we could suspect this node might be a hostscan node because these results match the signature of the hostscan in Section 2.6.1.1. The node tried to access to port 0 (invalid port number), 25 (SMTP), and 443 (HTTPS) on 1,888 different destination IP addresses. The port 0 is not a valid port number, but the packets can be formed and frequently used for searching valid hosts by attackers. Thus, this node was highly suspicious. By using the proposed trust management scheme, we could observe the malicious behaviors over 90 days without looking at the raw network flows closely. However, this node might not be classified as hostscan node in the client node group because the *MinTh* and *MaxTh* for the host-degree in this figure were much lower than the values in Figure 38.

4.4.2.3 Portscan attacks

(a). The host-degrees of a portscan attack node

(b). The port-degrees of a portscan attack node

(c). The betweenness centrality of a portscan attack node

(d). The clustering coefficient of a portscan attack node

(e). The trust changes of a portscan attack node

Figure 40. A portscan attack node that has been classified as malicious

Figure 40 shows the behaviors and the corresponding trust changes of a node that has been listed in the DShield Logs, and the trust management scheme classified as a portscan attack node. The host-degrees of the node showed interesting behaviors that placed in the suspicious area, which is between the $MinTh$ and $MaxTh$, as shown in Figure 40.(a). Since the host-degrees were suspicious, we can expect that the HT will be lower than the normal nodes. Even though the host-degrees were placed in the suspicious area, they were closer to the $MinTh$ than the $MaxTh$, and it was enough to increase the HT rather than decreasing the values as shown in Figure 40.(e). If the behaviors were closer to the $MaxTh$ than the $MinTh$, we could expect that the HT was slowly getting decreased but not as much as a node that showed higher host-degrees than the $MaxTh$ during the 14 days. On the other hand, the port-degrees were above the $MaxTh$ during the 14 days except on the 6th, 11th, and 13th days in Figure 40.(b).

However, even the behaviors in the 3 days were very close to the *MaxTh*, and they were enough to decrease the *PT*. Besides, the node did not connect with the neighbors strongly enough to have high clustering coefficients as shown in Figure 40.(d). The clustering coefficient values were below the static threshold 0.95 during the 14 days, and it decreased the *CCT* as shown in Figure 40.(e). The betweenness centrality values showed a kind of On-off patterns as shown in Figure 40.(c). It was below the *MinTh* on the 1st, 3rd, 6th, 10th, 12th, and 13th days while it was above the *MaxTh* on the 2nd and 5th days. The behaviors on the 4th, 7th, 8th, 9th, 11th, and 14th were placed in the suspicious area. The behaviors were enough to increase the *BCT* as shown in Figure 40.(e), but it could decrease the *PRT* even if the six out of fourteen behaviors were classified as normal. The *NPB* would be 3, and the *MPB* would be 2 in (5) on the 14th day, so the *PTR* would be 0.57 if we assumed that the other behaviors were classified as normal, and only the betweenness centrality behaved as shown in Figure 40.(c). It would be much lower than the *BCT* shown in Figure 40.(e) that was 0.7, and it shows the *PTR* would work as we expected that was described in Section 2.4.3 and 3.3.

The portscan attack nodes tend to show low host-degrees and high port-degrees as we described in Section 2.6.1.2. Therefore, we can specify the nodes that showed portscan patterns during the 14 days by searching the high *HT* and low *PT*. The low *CCT* and *BCT* showed the possibility that the node also performed a kind of behaviors that did not respect the communities' boundaries. The *PRT* and *OT* also could be a clue that the trust management scheme successfully detected the malicious behaviors.

(a). The host-degrees of a portscan attack node in the URI-server flows

(b). The port-degrees of a portscan attack node in the URI-server flows

(c). The betweenness centrality of a portscan attack node in the URI-server flows

(d). The clustering coefficient of a portscan attack node in the URI-server flows

(e). The trust changes of a portscan attack node in the URI- server flows
Figure 41. A portscan attack node that has been showed in the URI- server flows

The trust management scheme was able to complement a weakness of the outlier-based adaptive threshold. Figure 41 shows the behaviors and the corresponding trust changes of a URI-server node that showed portscan attack pattern during the 90 days. This node had 33 open ports and got connection requests to 43 different ports from 137 different source IP addresses. The most popular port was 23 (TELNET), and 38 nodes tried to access to this port. The second popular port was 5,000 (UPNP-EVNT) that is used by Universal Plug N' Play devices, and both ports could provide services as a server.

This node has tried to access to 17 different ports including 0 on 155 different destination IP address, and the node showed one suspicious behavior pattern, which tried to access from port 26,116 to 26,196 by increasing 10 on one target IP address.

Therefore, the port-degrees in Figure 41.(b) showed relatively high values while the host-degrees in Figure 41.(a) exceed the *MinTh* only twice on February 23rd and May 20th but never exceeded the *MaxTh* during the 90 days, so we could expect the high *HT* and low *PT*. Since the port-degrees exceeded 81 times during the 90 days, the low *PRT* and *OT* also could be expected. The results showed similar patterns to the node in Figure 40 that has been classified as malicious by DShield Logs and the trust management scheme.

While the trust management scheme was able to detect the malicious behavior patterns, Figure 41.(b) also showed a weakness of the outlier-based adaptive threshold, which is the relatively high *MaxTh*. For example, the node showed exactly the same behaviors on May 11th and 12th, but the same behaviors were evaluated in different ways. The port-degree on May 12th was classified as malicious behavior while the behavior on May 11th was classified as suspicious due to the high *MaxTh*. Even though the behaviors were evaluated in different ways, the trust management scheme was able to keep the *PT* in low values due to the previously observed malicious behaviors.

(a). The host-degrees of a portscan attack node in the non-URI flows

(b). The port-degrees of a portscan attack node in the non-URI flows

(c). The betweenness centrality of a portscan attack node in the non-URI flows

(d). The clustering coefficient of a portscan attack node in the non-URI flows

(e). The trust changes of a portscan attack node in the non-URI flows
Figure 42. A portscan attack node that has been showed in the non-URI flows

The node in Figure 42 showed even more obvious portscan behaviors than the nodes in Figure 40 and Figure 41. The node tried to access to 8,992 different ports from 49,152 to 65,531 by increasing 1 or 2 on a single node in the URI network. Thus, the node showed low host-degrees in Figure 41.(a) and high port-degrees in Figure 41.(b). Our trust management scheme increased the *HT* while decreasing the *PT*, *PRT*, and *OT* in Figure 41.(e) based on the behaviors. We were able to detect the malicious behavior patterns with the low *PRT* and *OT* and specified the portscan patterns with the high *HT* and low *PT*. Besides the betweenness centrality in Figure 41.(c) and clustering coefficient in Figure 41.(d) showed that the node was not a type of attack that did not respect the community boundaries, so the node had high *BCT* and *CCT*.

The node was not active from February 18th to 25th and March 11th, and the Figure 41.(e) showed that the inactive dates could not affect the trust calculations as

we designed in Section 3.3. Besides, the Figure 41.(b) showed a vulnerability of the adaptive threshold that was observed from May 18th to 28th. The behaviors were not much different from the other days that have been classified as malicious, but the behaviors were classified as suspicious due to the relatively high outliers on the days. The trust management scheme was also able to keep the low *PT* values with the previously observed malicious behaviors.

4.4.2.4 Hybrid-scan attacks

(a). The host-degrees of a hybrid-scan attack node

(b). The port-degrees of a hybrid-scan attack node

(c). The betweenness centrality of a hybrid-scan attack node

(d). The clustering coefficient of a hybrid-scan attack node

(e). The trust changes of a hybrid-scan attack node

Figure 43. A hybrid-scan attack node that has been classified as malicious

Figure 43 shows the behaviors and the corresponding trust changes of a node that has been listed in the DShield Logs, and the trust management scheme classified as a hybrid-scan node. The port-degrees and clustering coefficient values clearly showed malicious behaviors in Figure 43.(b) and (d), the host-degrees were placed above the *MaxTh* except on the 5th day as shown in Figure 43.(a), and the betweenness centrality showed higher values than the *MaxTh* except on the 3rd day in Figure 43.(c). Therefore, we cannot expect high *HT*, *PT*, *CCT*, *BCT*, *PRT*, or *OT*, and they were shown in Figure 43.(e).

As we described in Section 2.6.1.3, the hybrid-scan attack nodes tend to show high host-degrees and port-degrees. Thus, we can specify the nodes that showed hybrid-scan patterns during the 14 days by searching the low *HT* and *PT*. The *CCT* and *BCT* showed the low value, this is one type of the attacks that do not respect community boundaries. The *PRT* and *OT* can also be used for detecting malicious nodes.

(a). The host-degrees of a hybrid-scan attack node in the URI-client flows

(b). The port-degrees of a hybrid-scan attack node in the URI-client flows

(c). The betweenness centrality of a hybrid-scan attack node in the URI-client flows

(d). The clustering coefficient of a hybrid-scan attack node in the URI-client flows

(e). The trust changes of a hybrid-scan attack node in the URI-client flows
Figure 44. A hybrid-scan attack node that has been showed in the URI-client flows

Figure 44 shows the behaviors and the corresponding trust changes of a node in the URI-client flows that showed the hybrid-scan attack patterns, which was described in Section 2.6.1.3. The node tried to access to 4,376 different ports on 54,808 different destination IP addresses during the 90 days. The node showed higher host-degrees than the *MaxTh* every day, and the pot-degrees showed higher values than the *MaxTh* except on the May 11th, so we could expect the low *HT* and *PT* along with the low *PRT* and *OT*. The low *PRT* and *OT* could indicate the existence of attack patterns, and we could specify the type of the attacks as hybrid-scan with the low *HT* and *PT* values. Besides, we also could indicate that the type of attacks did not respect the communities' boundaries with the low *BCT* and *CCT*.

4.4.2.5 Misclassified nodes

In the previous section, we looked at a few nodes that both the DShield Logs and our trust management scheme agreed that the nodes were malicious. On the contrary, we will show a few nodes that the trust management scheme suspected they were malicious nodes while the DShield Logs failed to classify as malicious. The looked at a few nodes that have behaved like the nodes in the previous section, and they can also be classified as malicious with the definitions in Section 2.6.1.

(a). The host-degrees of a hostscan attack node

(b). The port-degrees of a hostscan attack node

(c). The betweenness centrality of a hostscan attack node

(d). The clustering coefficient of a hostscan attack node

(e). The trust changes of a hostscan attack node

Figure 45. A hostscan attack node that has not been classified as malicious

As we can see in Figure 45, the behaviors of the node showed very similar patterns to the node in Figure 37, which both the DShield Logs and our trust management scheme agreed the node was a hostscan attack node. Every host-degree was above the *MaxTh* except on the 6th day, and every port-degree was placed in the suspicious area close to the *MinTh* except on the 5th, 6th, and 13th days (Figure 45.(a) and (b)). These patterns match the signature of hostscan attack nodes that was presented in Section 2.6.1.1. However, the node has not been listed in the DShield Logs and classified as normal node while the proposed trust management scheme suspected that the node was a malicious node based on the low *HT*, *CCT*, *BCT*, *PRT* and *OT*, and we specified the node was a hostscan attack node based on the low *HT* and high *PT*. This shows that the DShield Logs failed to list all the hostscan attack nodes.

(a). The host-degrees of a portscan attack node

(b). The port-degrees of a portscan attack node

(c). The betweenness centrality of a portscan attack node

(e). The trust changes of a portscan attack node

Figure 46. A portscan attack node that has not been classified as malicious

The node in Figure 46 also showed even more obvious behavior patterns of the portscan than the node in Figure 40, which both the DShield Logs and our trust management scheme agreed the node was a portscan attack node. The node tried to connect to more than 700 ports on a single destination node during the 14 days as shown in Figure 46.(a) and (b). However, this node also has not been listed in the DShield Logs and classified as normal node like the node in Figure 36. Our trust management scheme suspected the node was a malicious node based on the low PT , PRT and OT , and we specified the node was a portscan attack node based on the high HT and low PT . This result also shows that there is a relationship among the host-degree, clustering coefficient, and betweenness centrality as shown in Figure 46.(c) and (d). Since every port-degree exceeded the $MaxTh$, the PRT and OT also had low values.

(a). The host-degrees of a hybrid-scan attack node

(b). The port-degrees of a hybrid-scan attack node

(c). The betweenness centrality of a hybrid-scan attack node

(d). The clustering coefficient of a hybrid-scan attack node

(e). The trust changes of a hybrid-scan attack node

Figure 47. A hybrid-scan attack node that has not been classified as malicious

Figure 47 shows the behaviors and the corresponding trust changes of a node that was suspected of performing hybrid-scan attacks during the 14 days. The node showed high host-degree and port-degree in Figure 47.(a) and (b) that exceeded the *MaxTh*. The behaviors were very similar to the node in Figure 43, which was classified by the DShield Logs and our trust management scheme. The behavior patterns also match the behavior signature of a hybrid-scan attack node as described in Section 2.6.1.3. However, the node has not been listed in the DShield Logs and considered as a normal node. Our trust management scheme was able to detect the attacking behaviors with the low *HT*, *PT*, *CCT*, *BCT*, *PRT* and *OT*, and we specified the node was a hybrid-scan attack node based on the low *HT* and *PT*.

To sum up, section 4.4 showed that the proposed trust management scheme successfully detected the hostscan, portscan, hybrid-scan, and any types of attacks that do not respect the community boundaries. Each trust value represented the behaviors of the host-degree, port-degree, clustering coefficient, and betweenness centrality respectively and used for classifying the type of attacks. Also, the experiments showed that the predictability trust and overall trust successfully integrated the multiple trust values into one. The trust management scheme also carefully evaluated each node by providing a cumulative evaluation method and gradually decreased the trust value as the system observed more malicious behaviors. Besides, we showed that the DShield Logs cannot be a perfect representation of all malicious nodes in section 4.4.2.5 by presenting a few examples that showed the same behavior patterns of scanning attacks but were not classified as malicious nodes.

4.5 The trust management scheme in special cases

The trust management was employed for providing a method to avoid erroneously labeling nodes with a few malicious behaviors in a system (Section 2.4.3 and 2.6.2) and providing second chances to the nodes in a system with trust redemption method (Section 2.4.3). Thus, the trust management scheme allows redemption of the trust as the node behaves normally. However, some smart attacker might use the redemption method and could disturb the evaluation method, so the Predictability trust (*PRT*) has been developed to detect on-off attacks that the modified Beta reputation method was not able to detect as we discussed in Section 3.3. In this section, we will provide a few examples that the trust management scheme considered as a temporary error or suspected as malicious. Besides, this section presents the long-term redemption of the *PRT* that allows a second chance to a malicious node based on a long-term observation (Section 2.4.3.2).

4.5.1 Evaluation of one temporary error

(a). The betweenness centrality of a temporary error node

(b). The clustering coefficient of a temporary error node

(c). The trust changes of a temporary error node
Figure 48. A node that showed a temporary error

Figure 48 shows the behaviors and the corresponding trust changes of a node that the trust management scheme believed that the malicious behavior was a temporary error or a mistake. The node's betweenness centrality and clustering coefficient showed only one malicious behavior on the 12th day as shown in Figure 48.(a) and (b), so the *BCT* and *CCT* were lowered once in Figure 48.(c). As the malicious behavior was observed, the *PRT* was lowered on the same day. However, it was the only one malicious behavior of the node, so the trust management scheme wanted to give a second chance by recovering the *BCT* and *CCT* from the 13th day, which the node showed normal behaviors again. Since the current size of the *NBW* was 5 and the current size of the *MBW* was 24 due to the *OT* value, which was 0.714 on the 14th day, the *PRT* will start recovering the value after pushing the malicious behavior out from the *NBW*, which needs to observe three more normal behaviors.

The secure system might produce another victim that has been stolen the identity by attackers if the system blocks based on a few malicious behaviors (Section 2.6.2). As we can see in this case, the trust management scheme can provide a second chance to the node because the malicious behavior could be a temporary error while it also does not easily recover the trust as shown in the case of *PRT*.

4.5.2 Evaluation of on-off attacks

(a). The clustering coefficient of an on-off attack node

(b). The trust changes of an on-off attack node
Figure 49. A suspected on-off attack node

The trust management scheme suspected that the node in Figure 49 could be an On-off attack node. The *PRT* and *OT* were lowered based on the malicious behaviors on the 3rd, 7th, 8th, and 11th days because the clustering coefficient showed malicious behaviors on the same days in Figure 49.(a). The behaviors in Figure 49.(a) showed a kind of On-off attack patterns that show normal and malicious behaviors alternatively, so the increase and decrease of the *CCT* repeatedly appeared in Figure 49.(b). As a result of the patterns, the *CCT* was bounded between 0.57 and 0.66 while the *PRT* gradually lowered the value. The Figure 49.(b) showed that the *PRT* has successfully detected the On-off patterns by extending the size of the sliding window.

4.5.3 Long-term redemption

(a). The host-degrees of a hostscan attack node in the URI-client flows

(b). The trust changes of a hostscan attack node in the URI-client flows

Figure 50. A hostscan attack node that has been showed in the URI-client flows

We were able to observe that the trust management successfully specified the hostscan attack patterns of a node and the trust redemption in these results. Figure 50 shows each behavior and the corresponding trust changes of a node in the URI-client flows that showed hostscan attack patterns, and we detected the patterns with our trust management scheme. The host-degrees in Figure 50.(a) exceeded every day, and we could expect the low *HT* because the behaviors matched the attack signature of hostscan in Section 2.6.1.1. Since the trust management scheme observed the malicious behaviors from the host-degrees, the *PRT* was decreased with the *OT*. Therefore, we were able to detect a pattern of attacks with the *OT* and specified the attack as hostscan.

The node has tried to access to the port 80(HTTP) and 445(Microsoft Directory Services) on 19,926 different destination IP address. Especially, the port 445 has been targeted by many attackers because it has the great vulnerability originally created by Windows file sharing, which may allow the attackers could gain remote access to the contents of hard disk directories or drives [95]. After the first 17 days, the node has been inactive until the last observed day due to either blocked by URI firewall or changed the identity. There is also a possibility that the firewall blocked the IP address and the attack node changed the identity for the other attacks. The trust management scheme can provide a second chance to the IP address by pushing out the previous malicious behaviors from the sliding windows because it would be a waste of the resources if the firewall keeps blocking the IP address. So, our trust management scheme started recovering the *PRT* from May 7th gradually as shown in Figure 50.(b). However, if a malicious behavior happens while it was recovering the trust, it would

drop the trust immediately with the previously observed malicious behaviors that have not been pushed out from the sliding windows yet.

4.6 URI summarized results

In this section, we show the summarized analysis results of the URI dataset. In statistical analysis-based ADS, the systems detect malicious behaviors in low probability regions while the normal behaviors occur in high probability regions. Malicious nodes cannot be detected if a threshold is too high while normal nodes can be classified as malicious if a threshold is too low as we discussed in Section 2.5 and 4.3.1. The main advantage of employing the adaptive thresholds method is that the system can set an appropriate threshold in a dynamic environment. We found that the adaptive thresholds maintained a similar pattern of the detection ratio of each normal, suspicious, and malicious nodes in a dynamic environment. Also, we compared the detection rates and the two thresholds of the adaptive thresholds method for different periods, which are during the semester, break, and final exam periods.

This section only summarized the observations in the URI dataset, and the more detailed report for the observation will be presented in the future manuscripts. Please note that this section did not include the clustering coefficients. Since the values do not show the heavy-tailed distribution as we showed in Section 4.3.3, we did not apply the adaptive threshold for the clustering coefficients.

4.6.1 Host degrees

On average, over the 90 days, the nodes showed higher host degrees in the order of URI clients (20.43), URI servers (15.95), and non-URI nodes (2.23). Therefore, the [Average MinTh (95% Confidence Interval) / Average MaxTh (95% Confidence Interval)] of the adaptive thresholds were different for each node type. The client nodes showed the averages [10.40 (± 0.98) / 41.53 (± 3.65)], the server nodes showed the averages [2.82 (± 0.21) / 9.10 (± 1.08)], and the non-URI nodes showed the averages [1.43 (± 0.02) / 3.88 (± 1.08)].

Figure 51. The host degree-based detection rates of the client nodes

In general, the thresholds of the client nodes during the semester [11.09 (± 1.13) / 43.82 (± 4.13)] were greater than during the break [8.38 (± 1.80) / 34.87 (± 7.16)] and smaller than during the final exam [11.87 (± 1.87) / 47.33 (± 7.53)]. Figure 51 shows

that the detection rates of the normal, suspicious, and malicious nodes have the similar patterns no matter in which period the data have been collected, even though different time periods have different *MinTh* and *MaxTh*. At the same time, the client nodes show pretty different patterns between the weekdays and weekends. In general, the thresholds of the client nodes in the weekdays $[12.93 (\pm 0.62) / 50.90 (\pm 2.31)]$ are greater than the thresholds in the weekends $[3.44 (\pm 0.14) / 15.75 (\pm 0.78)]$, and the detection rates of the malicious nodes on the weekends are greater than the detection rates on the weekdays.

Figure 52. The host degree-based detection rates of the server nodes

In general, the thresholds of the server nodes during the semester $[3.06 (\pm 0.24) / 10.46 (\pm 1.25)]$ are greater than during the break $[2.06 (\pm 0.19) / 5.36 (\pm 0.99)]$ and smaller than the thresholds during the final exam $[2.37 (\pm 0.33) / 7.33 (\pm 2.61)]$. Figure 52 also shows that the detection rates of the normal, suspicious, and malicious nodes

have similar patterns no matter in which period the data have been collected, even though different time periods have required different *MinTh* and *MaxTh*. Meanwhile, unlikely the client nodes, the server nodes display very similar patterns between the weekdays and weekends. On average, the thresholds of the server host-degrees on the weekdays $[3.23 (\pm 0.21) / 11.37 (\pm 1.05)]$ are greater than the thresholds on the weekends $[1.71 (\pm 0.09) / 3.33 (\pm 0.30)]$. Since some server nodes may work like the client nodes, the resulting host-degree *MinTh* and *MaxTh* are different on the weekdays and weekends.

Figure 53. The host degree-based detection rates of the non-URI nodes

While the URI nodes require relatively higher thresholds during the semester, the non-URI nodes have resulted in the lowest host-degree thresholds during the semester. In general, the thresholds during the semester $[1.45 (\pm 0.03) / 24 (\pm 5.48)]$ are smaller

than during the break $[1.37 (\pm 0.05) / 82.22 (\pm 27.05)]$ and smaller than during the final exams $[1.37 (\pm 0.09) / 79.00 (\pm 62.88)]$. Figure 53 shows that the normal nodes detection rates during the break are relatively greater than during the semester because the suspicious nodes detection rates have decreased during the break. However, the malicious nodes detection rates were similar no matter in which period the dataset has been collected. Meanwhile, the host-degree thresholds on the weekdays $[1.41 (\pm 0.03) / 39.86 (\pm 11.82)]$ and on the weekends $[1.47 (\pm 0.05) / 36.17 (\pm 15.06)]$ show similar values, implying that the behaviors of non-URI nodes are not affected by the weekdays and weekends. The detection rates on the weekdays and weekends also show similar patterns.

4.6.2 Port degrees

On average, over the 90 days, the nodes show higher port degrees in the order of server (2.83), client (2.63), and non-URI nodes (1.17), and we have expected that the values [*Average MinTh (95% Confidence Interval) / Average MaxTh (95% Confidence Interval)*] of the adaptive thresholds to be placed in the same order. However, the adaptive thresholds have resulted in a different order. The client nodes have resulted in the average thresholds of $[1.90 (\pm 0.06) / 5.90 (\pm 0.54)]$, the non-URI nodes have resulted in the averages of $[1.07 (\pm 0.00) / 2.92 (\pm 0.06)]$, and the server nodes have resulted in the averages of $[1.41 (\pm 0.01) / 1.84 (\pm 0.11)]$. This supports the fact that the port degrees of the server nodes are relatively evenly distributed on each day.

Figure 54. The port degree-based detection rates of the client nodes

In general, the thresholds of the client nodes during the semester [$1.95 (\pm 0.07)$ / $6.25 (\pm 0.63)$] are greater than during the break [$1.79 (\pm 0.14)$ / $4.87 (\pm 0.98)$] and similar to during the final exam [$2.04 (\pm 0.07)$ / $6.67 (\pm 0.65)$]. Figure 54 shows that the detection rates of the normal, suspicious, and malicious nodes result in similar patterns no matter in which period the data have been collected. Meanwhile, the rates of the normal and suspicious nodes are very different on the weekdays and weekends. The thresholds of the client nodes in the weekdays [$2.07 (\pm 0.04)$ / $7.21 (\pm 0.40)$] were greater than the thresholds in the weekends [$1.46 (\pm 0.04)$ / $2.29 (\pm 0.19)$]. If we look only at the detection rates on the weekdays, the detection rates of the normal nodes are pretty stable with the average of 73.71% and the standard deviation 4.34%. At the same time, the detection rates of the normal nodes on the weekends were varied with

the average of 65.78% and the standard deviation 17.24%. However, the detection rates of the malicious nodes are similar every day.

Figure 55. The port degree-based detection rates of the server nodes

In general, the thresholds of the server nodes during the semester [$1.42 (\pm 0.02)$ / $2.00 (\pm 0.00)$] were similar to during the break [$1.35 (\pm 0.02)$ / $2.00 (\pm 0.00)$] and similar to during the final [$1.39 (\pm 0.02)$ / $2.00 (\pm 0.00)$]. The detection rates of the normal, suspicious, and malicious nodes changed many times as shown in Figure 55. However, the detection rates of the malicious nodes were relatively stable compare to the normal and suspicious nodes with the average 6.94% and standard deviation 3.59%. Meanwhile, the thresholds in the weekdays [$1.44 (\pm 0.01)$ / $2.00 (\pm 0.0)$] and in the weekends [$1.31 (\pm 0.01)$ / $2.00 (\pm 0.00)$] did not affect a lot to the detection rates of the

malicious nodes; average 4.90% and standard deviation 3.79% in the weekdays and average 7.68% and standard deviation 3.24% in the weekends.

Figure 56. The port degree-based detection rates of the non-URI nodes

We observed that the periods of the data collection do not affect a lot to the port degree-based detection of the normal, suspicious, and malicious nodes in non-URI nodes. In general, the thresholds of the non-URI nodes during the semester [$1.08 (\pm 0.00)$ / $2.99 (\pm 0.03)$] were similar to during the break [$1.07 (\pm 0.01)$ / $2.74 (\pm 0.22)$] and similar to during the final [$1.08 (\pm 0.01)$ / $3.00 (\pm 0.00)$]. Meanwhile, the thresholds of the nodes in the weekdays [$1.07 (\pm 0.00)$ / $2.94 (\pm 0.07)$] were also similar to the thresholds in the weekends [$1.08 (\pm 0.01)$ / $2.88 (\pm 0.14)$]. As shown in Figure 56, the detection rates of the normal, suspicious, and malicious nodes maintained similar patterns for the 90 days no matter in which period the dataset has been collected.

4.6.3 Betweenness centrality

On average, over the 90 days, the nodes showed higher betweenness centrality values in the order of non-URI (59273.84), server (3163.00), and client (2609.06). We also expected that the $[Average\ MinTh\ (95\%\ Confidence\ Interval) / Average\ MaxTh\ (95\%\ Confidence\ Interval)]$ of the adaptive threshold will be higher in the order of the average betweenness centrality values. However, the non-URI nodes showed the averages $[66299.43\ (\pm 6330.96) / 489598.11\ (\pm 107872.60)]$, the client nodes showed the averages $[1131.25\ (\pm 58.29) / 4087.00\ (\pm 161.99)]$, and the server showed the averages $[264.90\ (\pm 41.51) / 976.75\ (\pm 159.97)]$. These values also showed that the betweenness centrality values of the server nodes were relatively evenly distributed on each day.

Figure 57. The betweenness centrality-based detection rates of the client nodes

In general, the thresholds of the client nodes during the semester [$1172.13 (\pm 66.20)$ / $4194.34 (\pm 176.12)$] were greater than during the break [$1012.17 (\pm 110.25)$ / $3774.31 (\pm 348.37)$] and slightly smaller than during the final [$1142.43 (\pm 211.72)$ / $4287.33 (\pm 559.24)$]. Figure 57 shows that the detection rates of the normal, suspicious, and malicious nodes showed the similar patterns no matter in which period the dataset has been collected with the different *MinTh* and *MaxTh*. Meanwhile, the thresholds of the client nodes in the weekdays [$1272.24 (\pm 42.55)$ / $4453.63 (\pm 133.18)$] were greater than the thresholds in the weekends [$743.55 (\pm 31.50)$ / $3078.76 (\pm 106.71)$]. The detection rates of the malicious nodes in the weekdays, which showed the average 14.60% with standard deviation 0.79% , showed more stable detection rates than in the weekends, which showed the average 18.27% with the standard deviation 6.59% .

Figure 58. The betweenness centrality-based detection rates of the server nodes

In general, the thresholds of the server nodes during the semester $[310.11 (\pm 49.07) / 1198.24 (\pm 189.42)]$ were greater than during the break $[124.97 (\pm 34.22) / 477.23 (\pm 143.65)]$ and greater than during the final $[158.74 (\pm 91.43) / 678.87 (\pm 511.77)]$. As shown in Figure 58, the detection rates of the normal, suspicious, and malicious nodes also showed the similar patterns no matter in which period the dataset has been collected with the different *MinTh* and *MaxTh*. Meanwhile, the thresholds of the client nodes in the weekdays $[342.02 (\pm 41.20) / 1130.38 (\pm 155.87)]$ were greater than the thresholds in the weekends $[40.56 (\pm 8.87) / 125.61 (\pm 28.60)]$. The detection rates of the malicious nodes in the weekdays, which showed the average 13.48% with standard deviation 0.76%, showed more stable detection rates than in the weekends, which showed the average 16.34% with the standard deviation 2.17%.

Figure 59. The betweenness centrality-based detection rates of the non-URI nodes

In general, the thresholds of the server nodes during the semester $[67997.48 (\pm 7442.53) / 535542.23 (\pm 139805.04)]$ were greater than during the break $[61352.92 (\pm 12028.11) / 355760.87 (\pm 95888.34)]$ and greater than during the final $[48243.67 (\pm 2272.85) / 200748.90 (\pm 21810.49)]$. Figure 59 shows that the detection rates of the normal, suspicious, and malicious nodes showed the similar patterns no matter in which period the dataset has been collected with the different *MinTh* and *MaxTh*. Meanwhile, the thresholds of the client nodes in the weekdays $[71832.59 (\pm 7721.01) / 519506.80 (\pm 135633.33)]$ were greater than the thresholds in the weekends $[51083.21 (\pm 8097.82) / 407349.18 (\pm 156022.24)]$. Figure 59 also shows that the detection rates were not affected by the weekdays and weekends.

4.7 Summary

Chapter 4 evaluated the proposed ADS with multiple experiments. Section 4.1 showed that the graph representation can reduce the network flows dataset drastically, and Section 4.2 showed the trust management scheme can represent the results of statistical analysis methods. Section 4.3 compared the adaptive threshold to static thresholds and showed the adaptive threshold works better than the static thresholds in a dynamic environment. Section 4.4 showed that the converted trust values were able to detect the hostscan, portscan, hybrid-scan, and the types of attacks that do not respect the community boundaries. The section also showed that the proposed method can detect the attacks that even the DShield Logs [7] or Firewall could not detect. Section 4.5 presented how the trust management scheme deals with three special cases; 1) there was a temporary error looked like an attack, 2) if an attack node

composed on-off attack to disturb the trust management scheme, and 3) how the trust management scheme provides a second chance to a node. Section 4.6 summarized the analysis results of the URI dataset and showed the detection rates of the URI nodes, especially the client nodes, were affected by the weekdays and weekends due to the different number of active nodes on campus while the non-URI maintained similar detection rates with the different thresholds.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTIONS

In this final chapter, the main achievements of this research are summarized. Moreover, there exist several directions that would help in further improving and advancing the efficiency of the system.

This chapter will summarize the main achievements of this research from five perspectives: the graph representation of network flows, characterization of each node by using statistical analysis methods, representation of the results by using a trust management scheme, the adaptive threshold, and the trust management scheme for providing a method to avoid erroneously labeling nodes with a few malicious behaviors. A summary of the experimental results will be presented, and this dissertation closes with a brief discussion of future directions.

This dissertation showed that the graph representation can reduce the dataset by removing duplicated relationships and the statistical analysis on the graph can characterize each node's behaviors. Also, we showed that how the trust management can overcome the disadvantages of the statistical analysis-based IDS. The results that were produced by statistical analyses were not intuitive because it was not clear whether the bigger number meant normal or malicious. For instance, the high numbers in the host-degree, port-degree, and betweenness centrality represent the malicious behaviors while the low numbers in the clustering coefficient would be malicious. By representing the results in the trust management scheme, we were able to produce

intuitive results, which the low trust value represents more malicious. Moreover, the trust management scheme made it possible to integrate various results of statistical analysis methods into one by converting the results in the same range, between 0 and 1. We showed that the adaptive threshold worked better than static thresholds in a dynamic environment because if a static threshold is too low, the rate of false positive will be excessive, and if the threshold is too high, detection rate will be low. In addition to the efficiency, the trust management scheme with the adaptive threshold was able to represent how much the behavior closer to the normal or malicious behavior. Moreover, a secure system needs to carefully evaluate each node before blocking the suspicious nodes because there exist many methods that can hide the identity of the attackers, and many normal users can be affected. The proposed ADS could evaluate each node cautiously by providing cumulative evaluation by using the trust management scheme with the adaptive threshold and could also detect the attacks that even the DShield Logs or Firewall could not detect.

In our further work, we will show more analysis results of the URI dataset to show the usability of the adaptive thresholds. We also would like to analyze more than the host-degree, port-degree, betweenness centrality, and clustering coefficient. Since we have collected more than the IP addresses and port information of the source and destination nodes, the statistical analysis methods can be applied to the other information, such as the countries, size of the packets, the location of the nodes in the university, the types of application, etc. The datasets may be considered as a big data, and there exist many big data analysis methods. In the future research, we will apply big data analysis methods and detect malicious behaviors in the network flows.

APPENDIX I

PSUEDOCODEDS

A. Pseudocode for the predictability trust

```
( NPB, MPB ) ← getPRTStatus ( bv_1, bv_2, bv_3, bv_4, NPB, MPB )
```

```
( bv_1.GB, bv_1.BB ) ← CALL getBVStatus ( bv_1 )
( bv_2.GB, bv_2.BB ) ← CALL getBVStatus ( bv_2 )
( bv_3.GB, bv_3.BB ) ← CALL getBVStatus ( bv_3 )
( bv_4.GB, bv_4.BB ) ← CALL getBVStatus ( bv_4 )

IF bv_1.BB == 1 || bv_2.BB == 1 || bv_3.BB == 1 || bv_4.BB == 1
    NPB ← 0;
    MPB ← 1;
ELIF bv_1.GB == 1 && bv_2.GB == 1 && bv_3.GB == 1 && bv_4.GB == 1
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.GB == 1 && bv_2.BB == 0 && bv_3.BB == 0 && bv_4.BB == 0
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.BB == 0 && bv_2.GB == 1 && bv_3.BB == 0 && bv_4.BB == 0
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.BB == 0 && bv_2.BB == 0 && bv_3.GB == 1 && bv_4.BB == 0
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.BB == 0 && bv_2.BB == 0 && bv_3.BB == 0 && bv_4.GB == 1
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.GB == 1 && bv_2.GB == 1 && bv_3.BB == 0 && bv_4.BB == 0
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.GB == 1 && bv_2.BB == 0 && bv_3.GB == 1 && bv_4.BB == 0
    NPB ← 1;
    MPB ← 0;
bv_1.GB == 1 && bv_2.BB == 0 && bv_3.BB == 0 && bv_4.GB == 1
    NPB ← 1;
    MPB ← 0;
ELIF bv_1.BB == 0 && bv_2.GB == 1 && bv_3.GB == 1 && bv_4.BB == 0
    NPB ← 1;
    MPB ← 0;
```

```

    ELIF bv_1.BB == 0 && bv_2.GB == 1 && bv_3.BB == 0 && bv_4.GB == 1
        NPB ← 1;
        MPB ← 0;
    ELIF bv_1.BB == 0 && bv_2.BB == 0 && bv_3.GB == 1 && bv_4.GB == 1
        NPB ← 1;
        MPB ← 0;
    ELIF bv_1.GB == 1 && bv_2.GB == 1 && bv_3.GB == 1 && bv_4.BB == 0
        NPB ← 1;
        MPB ← 0;
    ELIF bv_1.GB == 1 && bv_2.GB == 1 && bv_3.BB == 0 && bv_4.GB == 1
        NPB ← 1;
        MPB ← 0;
    ELIF bv_1.BB == 0 && bv_2.GB == 1 && bv_3.GB == 1 && bv_4.GB == 1
        NPB ← 1;
        MPB ← 0;
    ELIF bv_1.GB == 1 && bv_2.BB == 0 && bv_3.GB == 1 && bv_4.GB == 1
        NPB ← 1;
        MPB ← 0;
    ELIF bv_1.GB == 0 && bv_1.BB == 0 && bv_2.GB == 0 && bv_2.BB == 0 &&
    bv_3.GB == 0 && bv_3.BB == 0 && bv_4.GB == 0 && bv_4.BB == 0
        NPB ← 0;
        MPB ← 0;
    ELSE
        DISP('ERROR')
    END IF
    RETURN NPB, MPB

```

```

( GB, BB ) ← getBVStatus ( bv )

```

```

    IF bv.bv >= bv.MaxTh
        GB ← 0
        BB ← 1
    ELIF bv.bv <= bv.MinTh
        GB ← 1
        BB ← 0
    ELSE
        GB ← 0
        BB ← 0
    END IF
    RETURN GB, BB

```

B. Pseudocodes for the Everyday malicious pattern-based ROC curves

```
( TPR, FPR )  $\leftarrow$  getEverydayROC ( ip_list, dataset )
```

```
TPR  $\leftarrow$   $\emptyset$ 
```

```
FPR  $\leftarrow$   $\emptyset$ 
```

```
degree_threshold = minimum degree
```

```
WHILE degree_threshold  $\neq$  ( maximum degree + 1 )
```

```
    ( ret )  $\leftarrow$  isEveryday ( ip_list, dataset, degree_threshold )
```

```
    ( TMP_TPR, TMP_FPR )  $\leftarrow$  calcEverydayROC ( ip_list, ret )
```

```
    TPR  $\leftarrow$  [ TPR, TMP_TPR ]
```

```
    FPR  $\leftarrow$  [ FPR, TMP_FPR ]
```

```
    INCREMENT degree_threshold by 1
```

```
END WHILE
```

```
RETURN TPR, FPR
```

```
( RET )  $\leftarrow$  isEveryday ( ip_list, dataset, degree_threshold )
```

```
RET  $\leftarrow$  initialize list with TRUEs
```

```
FOR each day
```

```
    FOR each behavior of each node
```

```
        IF behavior [ node_id ] < degree_threshold
```

```
            RET [ node_id ]  $\leftarrow$  FALSE
```

```
        END IF
```

```
    END FOR
```

```
END FOR
```

```
RETURN RET
```

```
( TMP_TPR, TMP_FPR ) = calcEverydayROC ( ip_list, isEveryday )
```

```
TP  $\leftarrow$  0
```

```
TN  $\leftarrow$  0
```

```
FP  $\leftarrow$  0
```

```
FN  $\leftarrow$  0
```

```
FOR each isEveryday of the matching ip_list
```

```
    IF isEveryday [ node_id ] == TURE AND listed in DShield Logs
```

```
        TP  $\leftarrow$  TP + 1
```

```
    ELIF isEveryday [ node_id ] == FALSE AND listed in DShield Logs
```

```
        FN  $\leftarrow$  FN + 1
```

```
    ELIF isEveryday [ node_id ] == TRUE AND not listed in DShield Logs
```

```
        FP  $\leftarrow$  FP + 1
```

```
    ELIF isEveryday [ node_id ] == FALSE AND not listed in DShield Logs
```

```
        TN  $\leftarrow$  TN + 1
```

```
    END IF
```

```
END FOR
```

```

TMP_TPR  $\leftarrow$  ( TP ) / ( TP + FN )
TMP_FPR  $\leftarrow$  ( FP ) / ( TN + FP )

```

```

RETURN TMP_TPR, TMP_FPR

```

C. Pseudocodes for the malicious Average degree pattern-based ROC curves

```

( TPR, FPR )  $\leftarrow$  getAverageROC ( ip_list, dataset )

```

```

TPR  $\leftarrow$   $\emptyset$ 
FPR  $\leftarrow$   $\emptyset$ 
degree_threshold = minimum degree
WHILE degree_threshold  $\neq$  ( maximum degree + 1 )
    ( ret )  $\leftarrow$  getAverage ( ip_list, dataset )
    ( TMP_TPR, TMP_FPR )  $\leftarrow$  calcAverageROC ( ip_list, ret, degree_threshold )
    TPR  $\leftarrow$  [ TPR, TMP_TPR ]
    FPR  $\leftarrow$  [ FPR, TMP_FPR ]
    INCREMENT degree_threshold by 1
END WHILE
RETURN TPR, FPR

```

```

( RET )  $\leftarrow$  getAverage ( ip_list, dataset )

```

```

RET  $\leftarrow$  initialize list with 0s
FOR each behavior of each node
    SUM  $\leftarrow$  0
    FOR each day
        SUM  $\leftarrow$  SUM + behavior [ node_id ]
    END FOR
    RET [ node_id ]  $\leftarrow$  SUM / Number of Days
END FOR
RETURN RET

```

```

( TMP_TPR, TMP_FPR ) = calcAverageROC ( ip_list, averages, degree_threshold )

```

```

TP  $\leftarrow$  0
TN  $\leftarrow$  0
FP  $\leftarrow$  0
FN  $\leftarrow$  0
FOR each average of the matching ip_list
    IF averages [ node_id ]  $\geq$  degree_threshold AND listed in DShield Logs
        TP  $\leftarrow$  TP + 1
    ELIF averages [ node_id ] < degree_threshold AND listed in DShield Logs
        FN  $\leftarrow$  FN + 1

```

```

    ELIF averages [ node_id ]  $\geq$  degree_threshold AND not listed in DShield Logs
        FP  $\leftarrow$  FP + 1
    ELIF averages [ node_id ]  $<$  degree_threshold AND not listed in DShield Logs
        TN  $\leftarrow$  TN + 1
    END IF
END FOR
TMP_TPR  $\leftarrow$  ( TP ) / ( TP + FN )
TMP_FPR  $\leftarrow$  ( FP ) / ( TN + FP )

RETURN TMP_TPR, TMP_FPR

```

D. Pseudocodes for the At Least Once malicious pattern-based ROC curves

```

( TPR, FPR )  $\leftarrow$  getAtLeastOnceROC ( ip_list, dataset )

```

```

TPR  $\leftarrow$   $\emptyset$ 
FPR  $\leftarrow$   $\emptyset$ 
degree_threshold = minimum degree
WHILE degree_threshold  $\neq$  ( maximum degree + 1 )
    ( ret )  $\leftarrow$  isAtLeastOnce ( ip_list, dataset, degree_threshold )
    ( TMP_TPR, TMP_FPR )  $\leftarrow$  calcAtLeastOnceROC ( ip_list, ret )
    TPR  $\leftarrow$  [ TPR, TMP_TPR ]
    FPR  $\leftarrow$  [ FPR, TMP_FPR ]
    INCREMENT degree_threshold by 1
END WHILE
RETURN TPR, FPR

```

```

( RET )  $\leftarrow$  isAtLeastOnce ( ip_list, dataset, degree_threshold )

```

```

RET  $\leftarrow$  initialize list with FALSEs
FOR each day
    FOR each behavior of each node
        IF behavior [ node_id ]  $\geq$  degree_threshold
            RET [ node_id ]  $\leftarrow$  TRUE
        END IF
    END FOR
END FOR
RETURN RET

```

```

( TMP_TPR, TMP_FPR ) = calcAtLeastOnceROC ( ip_list, isAtLeastOnce )

```

```

TP  $\leftarrow$  0
TN  $\leftarrow$  0
FP  $\leftarrow$  0

```

```

FN ← 0
FOR each isAtLeastOnce of the matching ip_list
  IF isAtLeastOnce [ node_id ] == TURE AND listed in DShield Logs
    TP ← TP + 1
  ELIF isAtLeastOnce [ node_id ] == FALSE AND listed in DShield Logs
    FN ← FN + 1
  ELIF isAtLeastOnce [ node_id ] == TRUE AND not listed in DShield Logs
    FP ← FP + 1
  ELIF isAtLeastOnce [ node_id ] == FALSE AND not listed in DShield Logs
    TN ← TN + 1
  END IF
END FOR
TMP_TPR ← ( TP ) / ( TP + FN )
TMP_FPR ← ( FP ) / ( TN + FP )

RETURN TMP_TPR, TMP_FPR

```

E. Pseudocodes for the trust-based ROC curves

```

( TPR_LIST, FPR_LIST ) ← startTrustROC ( ip_list, dataset )
TPR_LIST ← ∅
FPR_LIST ← ∅
trust_threshold ← 0.1
WHILE trust_threshold ≠ 1.0 )
  ( TPR, FPR ) ← getTrustROC ( ip_list, dataset, trust_threshold )
  TPR_LIST ← [ TPR_LIST; TPR ]
  FPR_LIST ← [ FPR_LIST; FPR ]
  INCREMENT trust_threshold by 0.1
END WHILE
RETURN TPR_LIST, FPR_LIST

```

```

( TPR, FPR ) ← getTrustROC ( ip_list, dataset, trust_threshold )
TPR ← ∅
FPR ← ∅
degree_threshold ← minimum degree
WHILE degree_threshold ≠ ( maximum degree + 1 )
  ( trust ) ← calcTrust ( ip_list, dataset, degree_threshold )
  ( TMP_TPR, TMP_FPR ) ← calcTrustROC ( ip_list, trust, trust_threshold )
  TPR ← [ TPR, TMP_TPR ]
  FPR ← [ FPR, TMP_FPR ]
  INCREMENT degree_threshold by 1
END WHILE
RETURN TPR, FPR

```

```

( TRUST )  $\leftarrow$  calcTrust ( ip_list, dataset, degree_threshold )
TRUST  $\leftarrow$  initialize list with 0s
gb  $\leftarrow$  initialize list with 0s
bb  $\leftarrow$  initialize list with 0s
FOR each day
    FOR each behavior of each node
        IF behavior [ node_id ] < degree_threshold
            gb [ node_id ]  $\leftarrow$  gb [ node_id ] + 1
        ELSE
            bb [ node_id ]  $\leftarrow$  bb [ node_id ] + 1
        END IF
    END FOR
END FOR
TRUST [ node_id ]  $\leftarrow$  ( gb [ node_id ] + 1 ) / ( gb [ node_id ] + bb [ node_id ] + 2 )
RETURN TRUST

```

```

( TMP_TPR, TMP_FPR ) = calcTrustROC ( ip_list, trust, trust_threshold )
TP  $\leftarrow$  0
TN  $\leftarrow$  0
FP  $\leftarrow$  0
FN  $\leftarrow$  0
last_day_trust  $\leftarrow$  last column of the trust
FOR each trust in last_day_trust
    IF last_day_trust [ node_id ]  $\leq$  trust_threshold AND listed in DShield Logs
        TP  $\leftarrow$  TP + 1
    ELIF last_day_trust [ node_id ] > trust_threshold AND listed in DShield Logs
        FN  $\leftarrow$  FN + 1
    ELIF last_day_trust [ node_id ]  $\leq$  trust_threshold AND not listed in DShield Logs
        FP  $\leftarrow$  FP + 1
    ELIF last_day_trust [ node_id ] > trust_threshold AND not listed in DShield Logs
        TN  $\leftarrow$  TN + 1
    END IF
END FOR
TMP_TPR  $\leftarrow$  ( TP ) / ( TP + FN )
TMP_FPR  $\leftarrow$  ( FP ) / ( TN + FP )

RETURN TMP_TPR, TMP_FPR

```

F. Pseudocodes for the outlier-based adaptive threshold

```
( MAX_TH, MIN_TH ) ← getThresholds ( dataset )
```

```
delta ← 0.05
sorted_dataset ← sort the dataset in descend order
kappa ← number of data / 4
k ← 1
WHILE k < Number of data
    alpha ← CALL calcModifiedHill ( sorted_dataset, k, kappa )
    r_i ← ( 1 - ( 1 - delta ) ^ ( 1 / k ) ) ^ ( -1 / ( alpha * k ) )
    R_i ← sorted_dataset [ k ] / sorted_dataset [ k + 1 ]
    IF R_i > r_i
        MAX_TH ← sorted_dataset [ k + 1 ]
        refined_data ← remove the outliers from the sorted_dataset
        MIN_TH ← mean ( refined_data )
        RETURN MAX_TH, MIN_TH
    END IF
    INCREMENT k by 1
END WHILE
RETURN FAIL
```

```
( ALPHA ) ← calcModifiedHill ( sorted_dataset, k, kappa )
```

```
X_k ← sorted_dataset [ k ]
X_kappa ← sorted_dataset [ kappa ]

log_sum ← 0
log_index ← k
WHILE log_index ≠ ( kappa + 1 )
    log_sum ← log_sum + log ( sorted_dataset [ log_index ] )
    INCREMENT log_index by 1
END WHILE

ALPHA ← ( ( ( k - 1 ) / ( kappa - ( k - 1 ) ) ) * log ( X_k ) - ( ( kappa - 1 ) / ( kappa - ( k - 1 ) ) ) * log ( X_kappa ) + ( 1 / ( kappa - ( k - 1 ) ) ) * log_sum ) ^ ( -1 ) )

RETURN ALPHA
```

G. NPB and MPB for the predictability trust

```
( NPB, MPB ) ← getEvaluation ( host_degree, port_degree,  
                                h_max, h_min, p_max, p_min )
```

```
NPB ← 0  
MPB ← 0  
day ← 1  
WHILE day < (Number of days + 1)  
  IF host_degree [ day ] >= h_max  
    h_bb ← 1  
    h_gb ← 0  
  ELIF host_degree [ day ] <= h_min  
    h_bb ← 0  
    h_gb ← 1  
  ELSE  
    h_bb ← 0  
    h_gb ← 0  
  END IF  
  
  IF port_degree [ day ] >= p_max  
    p_bb ← 1  
    p_gb ← 0  
  ELIF port_degree [ day ] <= p_min  
    p_bb ← 0  
    p_gb ← 1  
  ELSE  
    p_bb ← 0  
    p_gb ← 0  
  END IF  
  
  IF ( h_bb == 1 && p_bb == 1 ) || ( h_bb == 1 && p_bb == 0 ) ||  
    ( h_bb == 0 && p_bb == 1 )  
    NPB ← NPB + 0  
    MPB ← MPB + 1  
  ELIF h_gb == 1 && h_bb == 1  
    NPB ← NPB + 1  
    MPB ← MPB + 0  
  ELSE  
    NPB ← NPB + 0  
    MPB ← MPB + 0  
  END IF  
  INCREMENT day by 1  
END WHILE  
RETURN NPB, MPB
```

APPENDIX II

TABLES

A. Comparisons of the amount of dataset being analyzed in the European dataset

Day	1	2	3	4	5	6	7
Original	2033520	2582552	2658580	2591969	1857031	1986025	2864999
Graph	338919	479894	476489	426867	242404	268617	489119
Saved	83.3%	81.4%	82.1%	83.5%	86.9%	86.5%	82.9%
Day	8	9	10	11	12	13	14
Original	2983068	2841339	2884174	2639961	1990954	1970922	2762730
Graph	513891	523543	523104	458629	264846	283987	523031
Saved	82.8%	81.6%	81.9%	82.6%	86.7%	85.6%	81.1%

a) Comparisons of the amount of IP addresses to be analyzed between the original and graph represented in the European dataset

Day	1	2	3	4	5	6	7
Original	2033520	2582552	2658580	2591969	1857031	1986025	2864999
Graph	449164	569886	561409	532058	366307	408868	593437
Saved	77.9%	77.9%	78.9%	79.5%	80.3%	79.4%	79.3%
Day	8	9	10	11	12	13	14
Original	2983068	2841339	2884174	2639961	1990954	1970922	2762730
Graph	576659	570111	583733	537062	381646	385751	565519
Saved	80.7%	79.9%	79.8%	79.7%	80.8%	80.4%	79.5%

b) Comparisons of the amount of ports to be analyzed between the original and graph represented in the European dataset

B. Comparisons of the amount of dataset being analyzed in the URI dataset

Day	1	2	3	4	5	6	7
Original	243790	341427	280573	244027	239039	143410	144945
Graph	152678	180531	176662	142146	150214	80086	84414
Saved	37.4%	47.1%	37.0%	41.8%	37.2%	44.2%	41.8%
Day	8	9	10	11	12	13	14
Original	267088	271934	276411	268633	236520	144232	160232
Graph	169553	169272	173777	167295	150696	90969	96034
Saved	36.5%	37.8%	37.1%	37.7%	36.3%	36.9%	40.1%
Day	15	16	17	18	19	20	21
Original	283563	271568	297094	292063	248845	153801	173922
Graph	175528	173179	188807	186227	157337	93485	103471

Saved	38.1%	36.2%	36.5%	36.2%	36.8%	39.2%	40.5%
Day	22	23	24	25	26	27	28
Original	297167	285278	290520	309071	49516	87645	236083
Graph	182704	175760	176831	199923	34103	50921	146806
Saved	38.5%	38.4%	39.1%	35.3%	31.1%	41.9%	37.8%
Day	29	30	31	32	33	34	35
Original	221660	139798	149500	293320	295133	294104	317198
Graph	144689	86955	89171	184082	185555	185840	183723
Saved	34.7%	37.8%	40.4%	37.2%	37.1%	36.8%	42.1%
Day	36	37	38	39	40	41	42
Original	266514	168503	171537	314527	310587	321669	305103
Graph	174452	106587	103356	194957	188491	190148	181163
Saved	34.5%	36.7%	39.8%	38.0%	39.3%	40.9%	40.6%
Day	43	44	45	46	47	48	49
Original	279160	200273	189714	348489	320224	304232	291077
Graph	158559	93825	94090	202633	181879	181016	177446
Saved	43.2%	53.2%	50.4%	41.9%	43.2%	40.5%	39.0%
Day	50	51	52	53	54	55	56
Original	258292	168184	171474	302885	314768	327262	369205
Graph	157300	92751	90253	176488	177188	186313	192132
Saved	39.1%	44.9%	47.4%	41.7%	43.7%	43.1%	48.0%
Day	57	58	59	60	61	62	63
Original	315119	189703	195809	317470	308328	314982	299310
Graph	170140	93570	94352	185025	184978	191340	181010
Saved	46.0%	50.7%	51.8%	41.7%	40.0%	39.3%	39.5%
Day	64	65	66	67	68	69	70
Original	268504	181580	177183	345831	232169	160125	283543
Graph	153877	90026	87145	185174	133289	103928	177553
Saved	42.7%	50.4%	50.8%	46.5%	42.6%	35.1%	37.4%
Day	71	72	73	74	75	76	77
Original	252928	181854	148693	251745	258205	262348	246732
Graph	166431	98794	95722	162970	170592	171897	165830
Saved	34.2%	45.7%	35.6%	35.3%	33.9%	34.5%	32.8%
Day	78	79	80	81	82	83	84
Original	231042	148227	211136	335357	271628	261328	281513
Graph	157776	95782	96560	156789	158284	156270	175134
Saved	31.7%	35.4%	54.3%	53.3%	41.7%	40.2%	37.8%
Day	85	86	87	88	89	90	
Original	264349	171762	150282	168142	331850	317282	

Graph	164315	99023	82552	96148	178288	160511	
Saved	37.8%	42.4%	45.1%	42.8%	46.3%	49.4%	

a) Comparisons of the amount of IP addresses to be analyzed between the original and graph represented in the URI dataset

Day	1	2	3	4	5	6	7
Original	243790	341427	280573	244027	239039	143410	144945
Graph	51137	70860	57723	54160	49252	36872	38488
Saved	79.0%	79.3%	79.4%	77.8%	79.4%	74.3%	73.5%
Day	8	9	10	11	12	13	14
Original	267088	271934	276411	268633	236520	144232	160232
Graph	51476	50711	50192	48844	47054	37168	40263
Saved	80.7%	81.4%	81.8%	81.8%	80.1%	74.2%	74.9%
Day	15	16	17	18	19	20	21
Original	283563	271568	297094	292063	248845	153801	173922
Graph	51602	51352	55454	56533	50321	38386	41865
Saved	81.8%	81.1%	81.3%	80.6%	79.8%	75.0%	75.9%
Day	22	23	24	25	26	27	28
Original	297167	285278	290520	309071	49516	87645	236083
Graph	55260	53263	51917	55061	10904	22935	47875
Saved	81.4%	81.3%	82.1%	82.2%	78.0%	73.8%	79.7%
Day	29	30	31	32	33	34	35
Original	221660	139798	149500	293320	295133	294104	317198
Graph	45951	34570	36403	54196	53479	53691	53663
Saved	79.3%	75.3%	75.7%	81.5%	81.9%	81.7%	83.1%
Day	36	37	38	39	40	41	42
Original	266514	168503	171537	314527	310587	321669	305103
Graph	51474	38008	39634	54101	54374	55391	52845
Saved	80.7%	77.4%	76.9%	82.8%	82.5%	82.8%	82.7%
Day	43	44	45	46	47	48	49
Original	279160	200273	189714	348489	320224	304232	291077
Graph	49524	36848	38360	54292	55414	53412	54683
Saved	82.3%	81.6%	79.8%	84.4%	82.7%	82.4%	81.2%
Day	50	51	52	53	54	55	56
Original	258292	168184	171474	302885	314768	327262	369205
Graph	49866	38251	36051	54035	54371	56143	54397
Saved	80.7%	77.3%	79.0%	82.2%	82.7%	82.8%	85.3%
Day	57	58	59	60	61	62	63
Original	315119	189703	195809	317470	308328	314982	299310

Graph	50512	36271	39530	56385	56927	55849	54704
Saved	84.0%	80.9%	79.8%	82.2%	81.5%	82.3%	81.7%
Day	64	65	66	67	68	69	70
Original	268504	181580	177183	345831	232169	160125	283543
Graph	50862	37335	36893	57515	42333	35329	54775
Saved	81.1%	79.4%	79.2%	83.4%	81.8%	77.9%	80.7%
Day	71	72	73	74	75	76	77
Original	252928	181854	148693	251745	258205	262348	246732
Graph	52090	39892	39720	55448	55962	56075	55283
Saved	79.4%	78.1%	73.3%	78.0%	78.3%	78.6%	77.6%
Day	78	79	80	81	82	83	84
Original	231042	148227	211136	335357	271628	261328	281513
Graph	51767	38159	44680	65098	65982	71880	75169
Saved	77.6%	74.3%	78.8%	80.6%	75.7%	72.5%	73.3%
Day	85	86	87	88	89	90	
Original	264349	171762	150282	168142	331850	317282	
Graph	76600	62297	54744	55458	75284	73781	
Saved	71.0%	63.7%	63.6%	67.0%	77.3%	76.8%	

b) Comparisons of the amount of ports to be analyzed between the original and graph represented in the URI dataset

C. The intersections between the adaptive and each static threshold in the host-degree analysis ROC curves

	Static-5		Static-7		Static-9	
	TPR	FPR	TPR	FPR	TPR	FPR
Day 1	0.783537	0.207009	0.670732	0.154375	0.615854	0.125380
Day 2	0.786996	0.192407	0.710762	0.143458	0.643498	0.115227
Day 3	0.83683	0.197340	0.745921	0.144571	0.650350	0.115227
Day 4	0.78836	0.202299	0.708995	0.149422	0.640212	0.12152
Day 5	0.77	0.271609	0.693333	0.207444	0.6	0.167518
Day 6	0.791304	0.259221	0.692754	0.194712	0.608696	0.158796
Day 7	0.729282	0.185759	0.646409	0.134352	0.576427	0.107686
Day 8	0.765531	0.184705	0.689379	0.134365	0.607214	0.108346
Day 9	0.769231	0.191047	0.665992	0.14098	0.603239	0.113393

Day 10	0.740654	0.193691	0.635514	0.143678	0.565421	0.116319
Day 11	0.751351	0.198135	0.686486	0.146777	0.640541	0.118154
Day 12	0.763158	0.268392	0.674342	0.202611	0.595395	0.165226
Day 13	0.744681	0.257072	0.659574	0.192075	0.606383	0.157579
Day 14	0.827703	0.187601	0.756757	0.137017	0.699324	0.109715

D. The intersections between the adaptive and each static threshold in the port-degree analysis ROC curves

	Static-5		Static-7		Static-9	
	TPR	FPR	TPR	FPR	TPR	FPR
Day 1	0.77439	0.202861	0.557927	0.111695	0.542683	0.10423
Day 2	0.780269	0.193573	0.562780	0.11059	0.542601	0.105166
Day 3	0.822844	0.187947	0.564103	0.104642	0.550117	0.098941
Day 4	0.78836	0.1894	0.579365	0.104382	0.558201	0.099076
Day 5	N/A	N/A	0.513333	0.149677	0.5	0.141064
Day 6	N/A	N/A	0.527536	0.160788	0.498551	0.153034
Day 7	0.725599	0.185973	0.506446	0.103004	0.480663	0.097799
Day 8	0.761523	0.172533	0.54509	0.092423	0.527054	0.86809
Day 9	0.748988	0.176256	0.514170	0.095122	0.483806	0.08977
Day 10	0.738318	0.178390	0.502336	0.097056	0.485981	0.091386
Day 11	0.751351	0.183936	0.548649	0.101758	0.535135	0.095109
Day 12	N/A	N/A	0.529605	0.148967	0.513158	0.141639
Day 13	N/A	N/A	0.56383	0.148294	0.542553	0.140574
Day 14	0.790541	0.173059	0.614865	0.0938	0.594595	0.088407

BIBLIOGRAPHY

- [1] T. File and C. Ryan, “Computer and Internet Use in the United States: 2013,” 2014.
- [2] “PANDALABS REPORT Q2 2015,” 2015.
- [3] E. D. Kolaczyk, “Statistical Analysis of Network Data: Methods and Models,” *Springer Ser. Stat.*, p. 386, 2009.
- [4] C. Schluter and M. Trede, “Identifying multiple outliers in heavy-tailed distributions with an application to market crashes,” *J. Empir. Financ.*, vol. 15, no. 4, pp. 700–713, 2008.
- [5] A. Jøsang and R. Ismail, “The beta reputation system,” in *Proceedings of the 15th bled electronic commerce conference*, 2002, pp. 41–55.
- [6] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, “Intrusion as (anti)social communication,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12*, 2012, p. 886.
- [7] “DShield.org.” [Online]. Available: <http://dshield.org/>.
- [8] J. Anderson, “Computer security threat monitoring and surveillance,” 1980.
- [9] D. E. DE Denning, “An intrusion-detection model,” *Softw. Eng. IEEE Trans.*, no. 2, pp. 222–232, 1987.
- [10] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, “Comprehensive approach to intrusion detection alert correlation,” *Dependable Secur. Comput. IEEE Trans.*, vol. 1, no. 3, pp. 146–169, 2004.

- [11] Z. Zhang and J. Li, "HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," *Proc. IEEE Work. ...*, 2001.
- [12] K. Scarfone and P. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS), 2007," *NIST Spec. Publ.*, pp. 800–894, 2012.
- [13] M. Aydın, A. Zaim, and K. Ceylan, "A hybrid intrusion detection system design for computer network security," *Comput. Electr. Eng.*, 2009.
- [14] M. Tavallaei, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.
- [15] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, "A survey of intrusion detection techniques in cloud," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 42–57, 2013.
- [16] J. M. Kizza, "System intrusion detection and prevention," in *Guide to Computer Network Security*, Springer, 2015, pp. 273–298.
- [17] J. Zhang and M. Zulkernine, "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection," in *Communications, 2006. ICC '06. IEEE International Conference on*, 2006, vol. 5, pp. 2388–2393.
- [18] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Ann. Des Télécommunications*, vol. 55, no. 7–8, pp. 361–378.
- [19] W. W. Cohen, "Fast effective rule induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123.

- [20] E. Eskin, W. Lee, and S. J. Stolfo, "Modeling system calls for intrusion detection with dynamic window sizes," in *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, 2001, vol. 1, pp. 165–175.
- [21] D. Adamu Teshome and V. S. Rao, "A Cost Sensitive Machine Learning Approach for Intrusion Detection," *Glob. J. Comput. Sci. Technol.*, vol. 14, no. 6, 2014.
- [22] Z. Yu and J. J. P. Tsai, "A Framework of Machine Learning Based Intrusion Detection for Wireless Sensor Networks," in *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*, 2008, pp. 272–279.
- [23] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," *Recent Adv. Intrusion Detect.*, 2003.
- [24] T. Stibor, J. Timmis, and C. Eckert, "A comparative study of real-valued negative selection to statistical anomaly detection techniques," *Artif. Immune Syst.*, 2005.
- [25] R. Patel, A. Thakkar, and A. Ganatra, "A survey and comparative analysis of data mining techniques for network intrusion detection systems," *Int. J. Soft Comput. Eng. ISSN*, pp. 2231–2307, 2012.
- [26] M. S. Narayana, B. Prasad, A. Srividhya, and K. P. R. Reddy, "Data Mining Machine Learning Techniques--A Study on Abnormal Anomaly Detection System," *Int. J. Comput. Sci. Telecommun.*, vol. 2, no. 6, 2011.

- [27] S. Brugger, "Data mining methods for network intrusion detection," *Univ. Calif. Davis*, 2004.
- [28] M. Rafsanjani and Z. Varzaneh, "Intrusion Detection By Data Mining Algorithms: A Review," 2013.
- [29] L. M. L. de Campos, R. C. L. de Oliveira, M. Roisenberg, and L. de Campos, "Network intrusion detection system using data mining," in *Engineering Applications of Neural Networks*, Springer, 2012, pp. 104–113.
- [30] S. Parekh, B. Madan, and R. Tugnayat, "Approach For Intrusion Detection System Using Data Mining," *J. Data Min. Knowl. Discov.*, vol. 3, no. 2, p. 83, 2012.
- [31] W. Wang, X. X.-H. Guan, and X. X.-L. Zhang, "Modeling program behaviors by hidden Markov models for intrusion detection," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, 2004, vol. 5, pp. 2830–2835.
- [32] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognit.*, vol. 36, no. 1, pp. 229–243, Jan. 2003.
- [33] N. Ye, Y. Zhang, and C. Borrer, "Robustness of the Markov-chain model for cyber-attack detection," *Reliab. IEEE Trans.*, 2004.
- [34] H. S. Javitz and A. Valdes, "The SRI IDES statistical anomaly detector," *Res. Secur. Privacy, 1991. Proceedings., 1991 IEEE Comput. Soc. Symp.*, pp. 316–326, 1991.

- [35] H. S. H. Javitz, A. Valdes, and C. NRaD, "The NIDES statistical component: Description and justification," *Contract*, vol. 39, no. 92–C, p. 15, 1993.
- [36] Y. F. Jou *et al.*, "Design and implementation of a scalable intrusion detection system for the protection of network infrastructure," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, 2000, vol. 2, pp. 69–83.
- [37] H. Choi, H. Lee, and H. Kim, "Fast detection and visualization of network attacks on parallel coordinates," *Comput. Secur.*, vol. 28, no. 5, pp. 276–288, Jul. 2009.
- [38] S. Staniford, J. Hoagland, and J. McAlerney, "Practical automated detection of stealthy portscans," *J. Comput. Secur.*, 2002.
- [39] K. Wang, J. J. J. Parekh, and S. J. S. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," in *Recent Advances in Intrusion Detection*, 2006, pp. 226–248.
- [40] A. Patcha and J. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Networks*, 2007.
- [41] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, no. 1, pp. 18–28, 2009.
- [42] D. Ippoliti, *Automated network anomaly detection with learning, control and mitigation*. University of Colorado at Colorado Springs, 2014.

- [43] G. Cormode and S. Muthukrishnan, “An improved data stream summary: the count-min sketch and its applications,” *J. Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.
- [44] P. A. Porras and P. G. Neumann, “EMERALD: Event monitoring enabling response to anomalous live disturbances,” in *Proceedings of the 20th national information systems security conference*, 1997, pp. 353–365.
- [45] R. Vargiya and P. K. P. Chan, “Boundary detection in tokenizing network application payload for anomaly detection,” 2003.
- [46] K. Wang, G. Cretu, and S. J. S. Stolfo, “Anomalous payload-based worm detection and signature generation,” in *Recent Advances in Intrusion Detection*, 2006, vol. 3858, pp. 227–246.
- [47] K. Wang and S. J. S. Stolfo, “Anomalous payload-based network intrusion detection,” in *Recent Advances in Intrusion Detection*, 2004, pp. 203–222.
- [48] C. Krügel, T. Toth, and E. Kirda, “Service specific anomaly detection for network intrusion detection,” in *Proceedings of the 2002 ACM symposium on Applied computing*, 2002, pp. 201–208.
- [49] M. Klassen and N. Yang, “Anomaly based intrusion detection in wireless networks using Bayesian classifier,” in *Advanced Computational Intelligence (ICACI), 2012 IEEE Fifth International Conference on*, 2012, pp. 257–264.
- [50] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, “Bayesian event classification for intrusion detection,” in *Computer Security Applications Conference, 2003. Proceedings. 19th Annual*, 2003, pp. 14–23.

- [51] F. Jemili, M. Zaghdoud, and M. Ben Ahmed, "A Framework for an Adaptive Intrusion Detection System using Bayesian Network," in *2007 IEEE Intelligence and Security Informatics*, 2007, pp. 66–70.
- [52] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Min. Knowl. Discov.*, vol. 29, no. 3, pp. 626–688, 2014.
- [53] S. Staniford-Chen *et al.*, "GrIDS-a graph based intrusion detection system for large networks," in *Proceedings of the 19th national information systems security conference*, 1996, vol. 1, pp. 361–370.
- [54] H. Sun, J. Huang, J. Han, H. Deng, P. Zhao, and B. Feng, "gskeletonclu: Density-based network clustering via structure-connected tree division or agglomeration," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 2010, pp. 481–490.
- [55] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: a structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 824–833.
- [56] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Neighborhood formation and anomaly detection in bipartite graphs," in *Data Mining, Fifth IEEE International Conference on*, 2005, p. 8--pp.
- [57] J.-L. Guillaume and M. Latapy, "Bipartite structure of all complex networks," *Inf. Process. Lett.*, vol. 90, no. 5, pp. 215–221, 2004.

- [58] E. Kolaczyk and G. Csrdi, *Statistical Analysis of Network Data with R*. Springer Publishing Company, Incorporated, 2014.
- [59] M. Burgess, G. Canright, and K. Engø-Monsen, “A graph-theoretical model of computer security,” *Int. J. Inf. Secur.*, vol. 3, no. 2, pp. 70–85, 2004.
- [60] S. Fortunato, “Community detection in graphs,” *Phys. Rep.*, vol. 486, no. 3, pp. 75–174, 2010.
- [61] P. Symeonidis and C. Perentis, “Link Prediction in Multi-modal Social Networks,” in *Machine Learning and Knowledge Discovery in Databases*, Springer, 2014, pp. 147–162.
- [62] L. Akoglu, M. McGlohon, and C. Faloutsos, “Oddball: Spotting anomalies in weighted graphs,” in *Advances in Knowledge Discovery and Data Mining*, Springer, 2010, pp. 410–421.
- [63] F. Moradi, T. Olovsson, and P. Tsigas, “Overlapping Communities for Identifying Misbehavior in Network Communications,” in *Advances in Knowledge Discovery and Data Mining*, Springer, 2014, pp. 398–409.
- [64] M. McGlohon, L. Akoglu, and C. Faloutsos, “Statistical properties of social networks,” in *Social network data analytics*, Springer, 2011, pp. 17–42.
- [65] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.
- [66] Y. Chae, L. Cingiser Dipippo, and Y. L. Sun, “Trust Management for Defending On-off Attacks,” *Parallel Distrib. Syst. IEEE Trans.*, vol. PP, no. 99, p. 1, 2014.

- [67] P. Michiardi and R. Molva, "Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," 2002, p. 107.
- [68] Y. Chae and L. DiPippo, "Redeemable Reputation based Secure Routing Protocol for Wireless Sensor Networks," University of Rhode Island, 2012.
- [69] S. Buchegger and J. Y. Le Boudec, "Performance analysis of the CONFIDANT protocol," 2002, pp. 226–236.
- [70] S. Buchegger and J. Y. Le Boudec, "Self-policing mobile ad hoc networks by reputation systems," *Commun. Mag. IEEE*, vol. 43, no. 7, pp. 101–107, 2005.
- [71] L. C. DiPippo, Y. Sun, and K. R. Jr., "Secure Adaptive Routing Protocol for Wireless Sensor Networks," 2010. [Online]. Available: http://www.cs.uri.edu/wordpress/wp-content/uploads/2010/03/sarp_tech_report.pdf.
- [72] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2–3, pp. 293–315, 2003.
- [73] X. Mao and J. McNair, "Effect of on/off misbehavior on overhearing based cooperation scheme for MANET," in *MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010*, 2010, pp. 1086–1091.
- [74] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *International Conference on Pervasive Computing*, 2010, pp. 138–155.
- [75] A. J sang, R. Ismail, C. Boyd, A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618–644, 2007.

- [76] R. Clark, “State Estimation Schemes for Instrument Fault Detection,” 1989.
- [77] M. Tavallaee, W. Lu, S. A. Iqbal, A. Ghorbani, and others, “A novel covariance matrix based approach for detecting network anomalies,” in *Communication Networks and Services Research Conference, 2008. CNSR 2008. 6th Annual*, 2008, pp. 75–81.
- [78] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” in *ACM sigmod record*, 2000, vol. 29, no. 2, pp. 93–104.
- [79] B. Neethu, “Adaptive Intrusion Detection Using Machine Learning,” *IJCSNS*, vol. 13, no. 3, p. 118, 2013.
- [80] L. Hamel, *Knowledge discovery with support vector machines*. 2011.
- [81] F. E. Grubbs, “Procedures for detecting outlying observations in samples,” *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.
- [82] G. S. Maddala and K. Lahiri, *Introduction to econometrics*, vol. 2. Macmillan New York, 1992.
- [83] J. Nair, A. Wierman, and B. Zwart, “The Fundamentals of Heavy-tails: Properties, Emergence, and Identification,” *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 387–388, Jun. 2013.
- [84] B. M. Hill and others, “A simple general approach to inference about the tail of a distribution,” *Ann. Stat.*, vol. 3, no. 5, pp. 1163–1174, 1975.
- [85] I. Paredes-Oliva, P. Barlet Ros, J. Solé Pareta, and X. Dimitropoulos, “Addressing practical challenges for anomaly detection in backbone networks,” 2013.

- [86] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time.," *USENIX Secur. Symp.*, 2002.
- [87] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith, "Nymble: Anonymous IP-address blocking," in *Privacy Enhancing Technologies*, 2007, pp. 113–133.
- [88] T. Baba and S. Matsuda, "Tracing network attacks to their sources," *IEEE Internet Comput.*, vol. 6, no. 2, p. 20, 2002.
- [89] P. Srisuresh and K. Egevang, "Traditional IP network address translator (Traditional NAT)." RFC 3022, January, 2001.
- [90] "Measures of Central Tendency," *Laerd Statistics*. [Online]. Available: <https://statistics.laerd.com/statistical-guides/measures-central-tendency-mean-mode-median.php>. [Accessed: 04-Jan-2016].
- [91] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.
- [92] N. Kamiyama, T. Mori, and R. Kawahara, "Simple and adaptive identification of superspreaders by flow sampling," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, 2007, pp. 2481–2485.
- [93] G. David, "MatlabBGL," 2007. [Online]. Available: https://www.cs.purdue.edu/homes/dgleich/packages/matlab_bgl/.
- [94] "Name server." [Online]. Available: https://en.wikipedia.org/wiki/Name_server. [Accessed: 27-Jul-2015].
- [95] "Port Authority Database Port 445." [Online]. Available: https://www.grc.com/port_445.htm.