2011

Feature modeling and cluster analysis of malicious Web traffic

Ana Dimitrijevikj
West Virginia University

Feature modeling and cluster analysis of malicious Web traffic

by

Ana Dimitrijevikj

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

Katerina Goseva-Popstojanova, Ph.D., Chair
James D. Mooney, Ph.D.
Arun A. Ross, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2011

Keywords: Web applications, honeypots, heavy-tails, cluster analysis, vulnerability scans, attacks

Abstract

Feature modeling and cluster analysis of malicious Web traffic

by

Ana Dimitrijevikj

Many attackers find Web applications to be attractive targets since they are widely used and have many vulnerabilities to exploit. The goal of this thesis is to study patterns of attacker activities on typical Web based systems using four data sets collected by honeypots, each in duration of almost four months. The contributions of our work include cluster analysis and modeling the features of the malicious Web traffic. Some of our main conclusions are: (1) Features of malicious sessions, such as Number of Requests, Bytes Transferred, and Duration, follow skewed distributions, including heavy-tailed. (2) Number of requests per unique attacker follows skewed distributions, including heavy-tailed, with a small number of attackers submitting most of the malicious traffic. (3) Cluster analysis provides an efficient way to distinguish between attack sessions and vulnerability scan sessions.

Acknowledgements

First, I would like to thank my committee chair and advisor, Dr. Katerina Goseva-Popstojanova, for her guidance, support and encouragement throughout my graduate studies. Also, I would like to thank Dr. James Mooney and Dr. Arun Ross for being my graduate committee members. I am grateful for the support and advice from all my graduate committee members and I am thankful for their collaboration.

I would like to acknowledge that my work has been funded by the National Science Foundation under CAREER grant CNS-0447715. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

I also want to thank and acknowledge Risto Pantev, Brandon S. Miller, J. Alex Baker, Jonathan Lynch, and David Krovich for their collaboration in the research project.

I want to thank all my friends for their help and support.

Finally, I would like to express my deepest gratitude to my mother, father, and brother. They all motivated and encouraged me to pursue this degree. They were always supporting me and that means the world to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many companies use Web Applications to run their businesses. These Web Applications have many vulnerabilities that can be exploited by attackers. A report from SANS [45] concludes that 60% of the total attack attempts observed on the Internet were against Web applications. The harm of having the Web application compromised can have a huge impact on the company's business since it could potentially lose clients as their information is being stolen by attackers. [46] explains that there are two types of trends to compromise a Web server. The first trend is brute force password guessing attacks against Microsoft SQL, FTP, and SSH servers, and the second trend consists of the three most popular attacks against Web sites: SQL Injection, Cross-site Scripting and PHP File Include attacks. According to [45], SQL injection and Cross-Site Scripting flaws in open-source applications account for more than 80% of the vulnerabilities being discovered.

The second generation of the World Wide Web (WWW) or Web 2.0 is a generation in which dynamic and sharable content Web pages were introduced. Examples of such Web-applications are: Wikipedia, BlogSpot, Facebook, and so on. However, these Web 2.0 applications face many security problems and one in particular that stands out is spam. According to [11], spam is done by automatically posting random comments or promoting commercial services to blogs, wikis and so on. In fact, any web application that accepts and displays hyperlinks is potentially a target to spam. A report by Sophos in 2008 concluded that every 3 seconds new spam-related webpage is created [37].

The fact that most attackers find Web Applications attractive targets to attack motivates

us to analyze attacker activities on Web-based systems. In this thesis, the goal is to learn patterns and characteristics of attacker activities on typical Web based systems using data collected by honeypots [38], [35] which appear to be legitimate servers, but were actually collecting information on attacker activity. The main purpose of honeypots is to collect information. We use honeypots to install Web Applications and collect malicious traffic. The Web Applications on our honeypots are configured in a three-tier architecture consisting of a web server, application server and a back-end database.

Thus, having a Web-based system with meaningful functionality running on our honeypots makes more sense than having just independent applications installed. Older versions of the applications were installed, each with a known set of vulnerabilities. The assumption is that the older versions of the applications along with the type of applications will make the honeypots attractive targets for attackers.

The Web applications which are installed on our honeypots are:

- phpMyAdmin is a type of a Web based database administration application used to handle the administration of a database over the Web. phpMyAdmin [40] is an open source tool which is written in PHP and is used as a Web based front-end to MySQL.

- WordPress [49] is Web software used to create websites or blogs. Thus, it is a PHP-based open source blogging software. Blogs are web sites where users post their thoughts, opinions and they follow the principles of Web 2.0.

- Mediawiki [32] is an PHP-based open source wiki software that is widely used across the Internet. Wikis are Web applications in which users collaborate to generate an online encyclopedia such as Wikipedia.

The data collected through the application traffic logs from our honeypots is grouped into four data sets: WebDBAdmin I, Web 2.0 I, WebDBAdmin II and Web 2.0 II. The data is in a form of *Web Sessions*, each defined as a sequence of requests issued from the same user during a single visit to the Web System. In order to characterize attacker behavior, we do formal inferential statistical analysis and cluster analyzes of the malicious Web sessions observed on our honeypots. In particular, we model the tails of Web session features with

heavy-tailed distributions since most of the feature values are small, and only a few are big. We observed and modeled the phenomenon of having few attackers generating most of the malicious traffic on our honeypots. Finally, we do cluster analysis of Web sessions in order to explore whether we can separate attacks from vulnerability scans.

This work is a part of larger effort aimed at Improving Web Quality through an Integrated Approach together with [38] and [35]. The main contributions of this thesis are as follows:

- We carried out statistical analysis of the attackers activities, including several features of malicious Web sessions as well as analyzing the number of sessions and request per unique source attacker. Unlike the statistical characterization of the network traffic which has a long tradition (see for example [19], [22]), only very few attempts were made to statistically model some aspects of malicious traffic, such as [24], [6]. Neither of these studies included Web 2.0 applications and analysis of malicious Web sessions.

- We applied cluster analysis of malicious Web traffic in order to separate Attacks from Vulnerability scans sessions. Cluster analysis of malicious traffic collected from honeypots in the past was considered only in [9]. However, the work in [9] is different from ours since it is focused on distinguishing only between attacks aimed at port 445 and using only 3 clusters. In our work, we cluster sessions aimed at Web 2.0 applications as well as phpMyAdmin applications, and the purpose of the analysis is to distinguish between attacks and vulnerability scans Web sessions.

This thesis is organized as follows. Chapter 2 presents the related work. The experimental approach and data description is explained in Chapter 3. The statistical characterization of Web session features is presented in Chapter 4, which is followed by cluster analyzes of Web sessions in Chapter 5. Finally, conclusions are given in Chapter 6.

Chapter 2

Related Work & Our Contributions

In this section, first we review papers that discuss modeling of some aspects of malicious traffic. Then, we review papers that use clustering approaches for Internet traffic classification, i.e. clustering non-malicious traffic. Next, we review papers that use clustering based anomaly detection techniques i.e. clustering malicious and non-malicious traffic, and finally we address papers that cluster only malicious traffic.

2.1 Related Work

2.1.1 Modeling some aspects of malicious traffic

In this subsection, we review papers which discuss modeling of some aspects of malicious traffic.

In [24], the authors discuss modeling of attacks based on the data collected from the honeypots deployed. The authors account the attacks observed on 14 honeypot platforms (the total number of attacks 816476). The analysis presented in [24] included using linear regression to model the number of attacks per unit of time as a function of attacks originating from a single country, and fitting a mixture of exponential and Pareto distributions to model the time between two consecutive attacks.

The work presented in [6] compared the data collected by Leurre.com and two high-interaction honeypots which ran several unrelated applications. The analysis was based on the traffic at network layer and included distribution fitting of the time between the first

packet exchanges from reappearing IPs.

In [6], the authors also showed that a small number of attackers for each given network are responsible for a very large amount of the malicious traffic.

2.1.2 Clustering approaches for Internet Traffic Classification

In this subsection, we review papers in which cluster analysis was used to separate traffic into a number of network applications; that is, classes. The main types of network applications analyzed were HTTP, P2P, SMTP, POP3, Telnet, DNS and FTP, among others. These papers discuss several types of approaches used to identify an application: port-based, payload-based, and machine learning-based approaches. Port-based method is a traditional method that relies on mapping a known port number to a specific application. For example, HTTP traffic uses port 80, FTP port 21, and so on. However, P2P applications that are popular nowadays use dynamic port numbers to disguise their traffic or masquerade as well-known applications. In payload-based approach, packet payloads are analyzed to see if they contain specific signatures of known applications. This method fails to detect the application traffic when payloads are being encrypted. For example, P2P applications use encryption. Also, this method can only identify applications whose specific signatures are known. We review papers that use machine-learning approach for application identification, specifically unsupervised techniques to identify applications using flow features. The idea behind this approach is that applications usually send data in some pattern that can be used for identification. In order to find these patterns, network flow statistics (features) were being extracted. Clustering was used for classifying traffic using only transport layer statistics.

The data used in [50], was collected by a high-performance network monitor at a research facility. The goal was to classify different network flows and specify their application types. Flow was defined as series of packet exchanges between two hosts, identified by a 5-tuple (source IP address, source port, destination IP address, destination port, application protocol). Some of the flow features used were: the number of total packets-b-a (where 'a' is the client and 'b' is server), the number of actual data bytes-b-a, the number of pushed data

packets-a-b, size of the mean IPpackcet-a-b etc. The classes, that is, the applications identified, were WWW, mail, P2P, database, multimedia, etc. Since the number of features used to describe a flow can be several hundred, in [50] several feature selection and search techniques were used. Thus, algorithms [20] such as Correlation-based Feature Selection (CFS), Consistency-based subset evaluation, Information gain attribute evaluation were used for feature selection. Furthermore, algorithms such as backward and forward greedy search, Best First, and Ranker for searching were used for feature search techniques. Features that were frequent in the selected feature subsets were believed to be better at discriminating the classes. K-means algorithm was used to classify network traffic by application. The performance metrics used were recall and the overall accuracy. The results showed that the overall accuracy of K-means was up to 80%, and after a log transformation, the accuracy was improved to 90% or more. Also, the results showed that after log transformation, the recall value increased. The recall value of each class increased considerably; for example, 7 out of 11 classes had recall above 80%. The overall conclusion was that K-means performed better on log transformed data rather than on original data.

In [12], the authors analyzed data from two empirical traces: one publicly available Internet traffic trace from the University of Auckland and the other trace collected from the University of Calgary. The goal was to use clustering algorithms to identify groups of traffic based on transport layer statistics. Some of the statistical flow characteristics used were number of packets, mean packet size, number of bytes transferred, mean inter-arrival time of packets and so forth, whereas the classes were DNS, FTP, HTTP, IRC, LIMEWIRE, NNTP, POP3, etc. The following clustering algorithms were compared: K-means, DBSCAN [15] and Autoclass [7]. The approach was based on two phases. In the first phase, the clustering algorithm clusters the data. Then, these set of clusters were labeled to represent the classification model. A cluster is labeled by a traffic class that makes up the majority of its connections. The number of correctly classified connections in a cluster was refereed to as the True Positives (TP). Then overall accuracy was determined by the portion of the total TP for all clusters out of the total number of connections.

The overall accuracy of K-Means was approximately 49% for the Auckland IV data sets and 67% for the Calgary data sets. However, with a higher number of clusters, the accuracy

was increased. For example, when K was 500 the overall accuracy was 80%. One problem with this is the possibility of over-fitting the data. The overall accuracy of the DBSCAN algorithm was from 59.5% to 75.6% for the Auckland IV data sets and from 32.0% to 72.0% for the Calgary data sets. Finally, AutoClass was 92.4% and 88.7% accurate on the Auckland IV and Calgary data sets, respectively. Also, precision values were calculated for the three algorithms using the Calgary data set. The results showed that seven out of nine traffic classes had average precision values over 95%.

The analysis presented in [14] was very similar to the analysis in [12]. In [14], the authors proposed a semi-supervised learning method for traffic classification that relied on the flow statistics to classify traffic. The clustering algorithm used was K-means. As in [12], clustering of the flows was done first, followed by mapping of the clusters to the classes (applications). The data set in the first experiment was unlabeled. Then, after the clusters were created, a few flows from each clusters were randomly selected and labeled and these labeled flows were the basis for mapping clusters to applications. The results showed that for K=400 and two labeled flows per cluster, the attained flow accuracy was 94%. The second experiment included labeled flows and these labeled flows were mixed with a varying number of unlabeled flows. The results showed that by increasing the number of unlabeled flows in a data set together with the fixed labeled flows, the precision was also increased to above 90%.

The work in [31] used data of 6-hour Auckland - VI trace. The packets in this trace were divided into bi-directional flows. Flow features used were interarrival statistics, byte counts, connection duration, the number of transactions between transaction mode and bulk transfer mode, etc. The classes, that is, the applications used, were HTTP, FTP, SMTP, IMAP, NTP and DNS. EM (Expectation-Maximization) clustering algorithm was used to group traffic flows into clusters. The number of clusters was found automatically by cross-validation, so the best model (the number of clusters) was used. The algorithm separated the flows by their traffic type (bulk transfer, small transactions, multiple transactions, etc.). However, the algorithm was not successful in identifying individual applications as it was expected since individual applications behaved differently across different connections. Therefore, they did not estimate the accuracy of the classification.

The analysis presented in [13] was based on the Auckland IV and Auckland VI publicly

available data [3]. The features used for the connections were number of packets, mean packet size, flow duration, mean inter-arrival time of packets, etc. Some of the traffic classes were HTTP, SMTP, DNS, IRC, POP3, FTP, Limewire, etc. An EM clustering technique called AutoClass was used and the results from this technique were compared with a previously applied supervised machine learning approach, the Naive Bayes classifier. Three metrics were used to measure the effectiveness of the algorithms: precision, recall and overall accuracy. The results showed that for the Naive Bayes classifier, on average, the precision and recall for majority of the traffic classes was above 80%. As for the AutoClass approach, the precision and recall values were around 91%. The AutoClass had an overall accuracy of 91.2% while the Naive Bayes had an accuracy of 82.5%. Therefore, the results showed that the unsupervised technique outperformed the supervised by up to 9%.

In [5], the authors used one hour packet trace of TCP flows from an university network. The goal was to identify an application associated with a TCP flow. To do so, the information in the header of the first 5 packets of the flow was used, which captured the application negotiating phase. The applications (classes) used were: Edonkey, FTP, HTTP, Kazaa, POP3 etc. K-means clustering algorithm was used to identify the application associated with the TCP connection (flow). The results showed that more than 80 % of total flows of all of the applications were correctly identified by using the first five packets of each TCP flow.

The work presented in [51] used three 24-hour traffic traces, that is, Aukcland VI, Leipzig-II, and NZIX-II. Packets were classified into flows and flow characteristics such as packet inter-arrival time, packet mean length and variance, flow size in bytes and flow duration were computed. Feature selection techniques such as sequential forward selection (SFS) was used to examine the influence of different features. The results showed that packet length statistics was preferred over inter-arrival times statistics. The applications(classes) used were FTP, TELNET, SMTP, DNS, HTTP, AOL messenger, Napster, Half-life. An unsupervised machine learning technique called Autoclass (which is an unsupervised Bayesian classifier) was used to determine the best clusters set from the training data. The authors mapped each class to the application that was dominating the class. The results showed that the average accuracy across all traces was 86.5%. The false positive rate per application was also

calculated. The results showed that the FTP, TELNET and Web traffic have the highest percentage of false positives.

2.1.3 Cluster analysis for anomaly detection

In this subsection, we review papers that use clustering techniques on data sets that contain both normal and malicious traffic. The goal of these papers was to identify anomalies within the traffic such as malicious attack traffic.

In [27], Denial-of-Service attacks and Network Probe attacks were selected from DARPA 1998 [10] intrusion detection data. Also, portions of normal network traffic were used from the DARPA data sets. The goal was to describe a method for detecting these attacks using Principal Component Analysis (PCA). Therefore, four data sets were used to represent the attack types and three data sets to represent normal traffic. For each data set, 300 packets were collected and processed to extract the IP header information to produce 300 feature vectors. Each packet header was represented by a 12 dimensional feature vector. Principal Component Analysis was used to reduce the dimensionality of the feature vectors and also to provide visualization and analysis of the data. The first two principal components and their loading values were calculated along with other statistics. The principal component loadings are the coefficients of the principal components transformation and provide a summary of the influence of the original variables on the principal components. The highest loading values on the two components were used. The results showed that the loading values of the features on the first and second principal components can be used to identify an attack. For normal traffic, loading values appeared to be similar, while during an attack the loading values differed significantly for the first two principal components.

The analysis presented in [36] included flow records of both normal and anomalous traffic exported by routers and network monitors using Cisco Netflow protocol. The features of a flow were total number of packets and bytes transmitted at specific time intervals and the number of unique source destination pairs observed in the considered time interval. K-means clustering algorithm was used to divide the data into two clusters (K = 2); that is, normal and anomalous clusters. If the resulting centroids of the normal and anomalous clusters

were very close to each other, it was concluded that there were no traffic anomalies in the analyzed data. Therefore, the centroids of the clusters were used to detect anomalies in the data. For example, K-means was applied to services such as HTTP, FTP and SSH and the resulting centroids were very close to each other indicating that there were no anomalies in the analyzed data. When, UDP traffic was analyzed, the resulting centroids were not close to each other, and anomalous traffic was detected.

In [52] the data used was network traffic data sets DARPA 1998. 105 feature dimensions were used and scaled to be within [0,1] interval. Several centroid-based algorithms such as K-means, Mixture of Spherical Gaussians [4], Self-Organizing Maps (SOM) [25] and Neural-Gas [30] were used to demonstrate the advantages of clustering-based attack detection. To estimate the performance of the cluster algorithms, mean square error (mse), average purity and the runtime for each cluster algorithm was calculated. For evaluating the intrusion detection results, false positive rate, attack detection rate (recall) and overall accuracy were estimated too. For the whole data set consisting of 109,910 instances, the overall accuracy was 93.6%, false positive rate was 3.6% and the recall was 72%. The mse, average purity and run time for each clustering algorithm was calculated in case for 100 and 200 clusters. When k =100 the results showed that the kmo and Neural-Gas algorithms performed significantly better than the other methods in terms of mse and average purity. However, kmo achieved the same high quality with much less run time. Clustering results from 100 clusters and 200 clusters were compared and it was concluded that k-means and SOM were computationally efficient. The results also showed that for 100 clusters, the SOM was the worst for the false positive rates, and for 200 clusters, all algorithms performed well. For example, more than 50% of attacks were detected with a false positive rate of almost 0.

The data used in [28] was from two backbone networks (Abilene and Geant). Traffic features such as source address, destination address, source port and destination port were extracted from the header of the packet. The anomaly classes used were: DoS, port scan, network scan, worms etc. The goal was to find methods capable of detecting diverse set of network anomalies with high detection rate and low false alarm rate. Hierarchical agglomerative clustering and K-means clustering with Euclidian distance ware used to analyze and categorize anomalies. Clustering was used to get a better understanding of the nature of the

anomalies that have been detected and to possibly discover new anomaly types. The results of performing hierarchical clustering for 10 clusters on the 3-week Geant data set showed that the clusters were consistent, meaning that most of the points within a cluster had the same label.

2.1.4 Clustering malicious traffic

In this subsection, we review papers that use cluster analysis on malicious traffic only, which are most related to our work.

The work presented in [9] analyzed malicious activity that targeted port 445. The analyzes was done on data collected from two honeypots. Port scans and vulnerability scans were separated from the malicious activity, allowing distinction between attacks only. Therefore, their goal was to find characteristics that would separate attacks within the attack data. Attack characteristics used were: attack duration, number of packets, number of bytes, message lengths and so forth. Three classes of attacks were identified when analyzing the attack messages, and therefore K=3 was used for the K-means algorithm with Euclidian distance. The results showed that number of bytes was the best characteristic for separating attacks with only 2% of incorrectly clustered instances.

The data set used in [39] consisted of malware samples collected in a period of six months from different malware sources such as MWCollect [8], Malfease [42] and commercial malware feeds. The authors proposed a novel behavioral malware clustering system that can discover similarities among different malware samples that may not be captured by the current anti virus programs. The idea was that even thought there are many variants of the same malware, they all exhibit similar malicious activities. Therefore, their goal was to cluster malware samples according to similarities in their malicious behavior. The output of the authors analysis is hoped to be the input to algorithms that generate network signatures. The authors proposed a cluster process consisting of Coarse-grained Clustering, Fine-grained Clustering and Cluster merging. In the first phase, the Coarse-grained Clustering clusters the malware samples based on the following features: total number of HTTP requests generated by the malware, the number of GET requests, the number of POST requests, average length

of URLs, average number of parameters in the request and so on. This phase splits the malware into "large" (coarse-grained) clusters. In the second phase, that is, the fine-grained clustering, the clusters were further split into smaller groups. Finally, the cluster merging phase merges the clusters of malware that exhibit similar HTTP behavior. In all three phases, a single-linkage hierarchial clustering was used. As for the cluster validity, the authors propose a new approach based on a measure of the cohesion of each cluster and the separation among different clusters. The results showed that the majority of the clusters were compact and well separated from each other. The authors also discussed how their method could help the automatic generation of network signatures. A signature set was extracted from each of the clusters and the ability to detect the future malware was measured. All the signature sets were able to detect around 20% to 53% of future malwares that the current anti virus scanners were not able to detect.

2.2 The Contributions of this Thesis

The contributions of this thesis are as follows:

- We perform statistical analysis of attacker activities, such as modeling several features of malicious Web sessions and the also modeling the arrival of sessions and requests originating from a unique attacker. It appears that there were only very few attempts to statistically model some aspects of malicious traffic, such as [24], and [6]. Neither of these studies included Web 2.0 applications and analysis of malicious Web sessions.

- In this thesis we perform cluster analysis on malicious Web traffic which is represented by malicious Web sessions collected from our honeypots, with specific focus on distinguishing between vulnerability scan sessions and attack sessions. None of the related work have focused on these aspects of the malicious traffic. The goal of the related work that discuss cluster analysis of normal traffic was to classify different network flows and specify their application, whereas the goal of the related works that discuss clustering techniques on data sets that contain both normal and malicious traffic was to identify anomalies within the traffic such as malicious attack traffic. Only a few

papers apply cluster analysis on malicious traffic only. The authors in [9] analyzed malicious activity that targeted port 445. Port scans and vulnerability scans were separated from the malicious activity and their goal was to find characteristics that would separate attacks within the attack data.

- In our work we compare the performance of the K-means algorithm when all session features are used and when features are selected using several feature selection methods. Only few related works that use cluster analysis use formal feature selection techniques. In [50] several feature selection and search techniques were used such as Information gain attribute evaluation. In [51], Sequential forward selection was used for feature selection.

- We calculate the following performance measures of the K-means algorithms: probability of detection, probability of false alarm, precision, balance, and overall accuracy. Most of the related work estimates a subset of these measures. Furthermore, none of the related work that discuss cluster analysis have addressed the balance measure. The balance measure is important since it shows the balance between the probability of false alarm and the probability of detection. Ideally, a learner would have 100% probability of detecting an attack and 0% probability of false alarm, a result which is rarely achieved in reality. This is why we calculate the balance, which tells us how close the learner is to the ideal spot (of having 0% probability of false alarm and 100% probability of detection of attack).

Chapter 3

Experimental Approach and Data Description

In this chapter, we describe the experimental set up used in our work and the data sets collected for the analysis.

3.1 Honeypots

A *honeypot* is a computer which is connected to a network but it is not intended to be used by anyone. If anyone attempts to use the machine, it is probably an attempt to attack the machine [1]. To analyze attacker behavior on Web based systems, we developed and deployed three different honeypot configurations (set-ups) [16], [18]. Each configuration was installed on two identical honeypots, one advertised and the other unadvertised. The honeypot was *advertised* by using a technique called "transparent linking". This technique involves placing an "invisible" link that points to the honeypot on a regular public Web page. Since the link is invisible, it cannot be accessed by a regular user surfing the Web page, but it can be reached by Web spiders crawling that page [21]. By having the honeypot indexed by search engines we allow for attacks based on search engines. The IP address of the *unadvertised* honeypot, as the name applies, was not advertised anywhere on the Web. The unadvertised honeypot could only be reached by an IP-based strategy when an attacker scans a range of IP addresses. In this thesis, the analysis is focused only on traffic

collected from the advertised honeypots since the traffic from the unadvertised honeypot was significantly smaller. The observation from our experiments is that the attackers use search-engines to find the types and versions of applications that are installed, instead of performing random scans, especially for honeypots running Web 2.0 applications [18].

To summarize, each honeypot ran a Web-based system with a three-tier architecture consisting of Web server as a front-end, an application server, and database server as a back-end. The three different configurations developed as a part of a wider project, described in details in [38], [35] are as follows.

Two (advertised and unadvertised) honeypots were running on Linux operating system with Apache Web Server, PHP as an Application Server, and phpMyAdmin as a Web Application used as a front-end of an MySQL database. In addition, OpenSSH server and client were installed to allow remote login, as it is typical for many Web systems. There was only one SSH user account, which was set up with a strong password. The MySQL server is the only other application with login information, but no user accounts in the MySQL server were accessible to remote systems. However, attackers could still attempt to login to the MySQL server via the phpMyAdmin Web page. WebDBAdmin I is the name of the data set consisting of malicious traffic collected from the advertised honeypot during the period of June 2 to September 28, 2008.

The configuration of the other two honeypots had Windows XP operating system installed with IIS Web Server, PHP as an Application Server, and phpMyadmin as a Web Application used as a front-end of an MySQL database. WebDBAdmin II is the name of the data set consisting of malicious traffic collected from this advertised honeypot during the period of August 17, 2009 to January 17, 2010.

Another two honeypots (advertised and unadvertised) were running on Windows XP operating system with Microsoft IIS Web Server, PHP as an Application Server and Blog and Wiki as Web 2.0 Applications, with MySQL database. For each Web application a random text generator was used to generate random content so that the attackers would think the applications were actually being actively used. Multiple users with strong password accounts were created for each application. In Wordpress, anonymous users were allowed to post comments to blog entries. In MediaWiki, anonymous users had the same permission

level as logged in users and were allowed to post and edit entries [35].From this advertised honeypot, we have two datasets Web 2.0 I and Web 2.0 II which were collected in different time periods. The malicious traffic in Web 2.0 I were collected from March 30 until July 26, 2009, whereas the traffic in Web 2.0 II was collected from August 17, 2009 until January 17, 2010.

All honeypot configurations have a honeywall which acts as a brigding firewall between the honeypots and the Internet. Any traffic going to or from the honeypots passes through the honeywall which logs all the packets using TCPDump and forwards the packets without modifing the hop count of the packets. This makes the honeywall undetectable. Besides the network traffic, the application logs of various components running on our honeypots are captured too. These logs are stored in a central data repository which runs on a separate physical host. It should be mentioned that we analyzed only incoming traffic since the outgoing traffic from the honeypots consisted only of responses to requests sent to the honeypots. Also, non-malicious traffic consisting of system management traffic and legitimate Web crawlers (such as Google) was removed from all honeypot data.

3.2 Definition of Web Session

The malicious traffic analyzed in this thesis is in form of Web sessions. A Web session is defined as a sequence of requests issued from the same user during a single visit to the Web Application. Thus, a Web session has multiple Web requests from the same source IP address within a given threshold. In order to analyze the Web sessions observed on the honeypots, scripts were developed by [19], [17], [35], [38] to extract data from Web server logs (Apache and IIS). The output of the tool was a csv file in which each row was a distinct Web Session extracted from Web server log.

Upon detailed investigation of the requests within a Web session, each Web session was labeled as an attack or vulnerability scan on a Web based system. A session is labeled as an *Attack* if and only if there is at least one request used to attempt to exploit a specific vulnerability in a system. A session is labeled as *Vulnerability Scan* if and only if all requests are used to check for specific vulnerabilities in a system.

3.3 Web Session Features

In this section we explain the features of a Web session. Let X be a data set consisting of n Web sessions observed on a particular Web System, X= $\{x_1, ..., x_n\}$, where x_i is a data point that represents a session. Furthermore, each Web session is characterized by 43 different features $x_i = (f_1, ..., f_{43})$. Scripts were developed for extracting the features of each Web session [38]. Also, more details regarding the features can be found in [38]. In this thesis, we only give a list of the features, with brief descriptions. Each Web session is characterized by the following features:

1. *Number of requests*
2. *Bytes transferred*
3. *Duration* in seconds
4. Mean time between requests
5. Median time between requests
6. Minimum time between requests
7. Maximum time between requests
8. Standard Deviation of Time Between Requests
9. *Number of requests with GET method type*
10. *Number of requests with POST method type*
11. *Number of requests with OPTIONS method type*
12. *Number of requests with HEAD method type*
13. *Number of requests with PROPFIND method type*
14. *Number of requests with other method types, such as: PUT, DELETE, TRACE, CONNECT*

15. Number of requests to *Picture files* (extensions like .jpeg, .jpg, .gif, .ico, .png, etc.)
16. Number of requests to *Video files* (extensions like .avi, .mpg, .wmv, etc)
17. Number of requests to *static Application files* (extensions like .html, .htm)
18. Number of requests to *dynamic Application files* (php, .asp, etc.)
19. Number of requests to *Text files* (extensions like .txt, .ini, .css, etc.)
20. Number of requests with 1xx status code *(Informational status)*
21. Number of requests with 2xx status code *(Success status)*
22. Number of requests with 3xx status code *(Redirection status)*
23. Number of requests with 4xx status code *(Client Error)*
24. Number of requests with 5xx status code *(Server Error)*

    The following features are based on the length (measured in number of characters) of the request substring from the Web server access log. One example of a request string is: "GET //phpMyAdmin//scripts/setup.php HTTP/1.1". The length of a substring of this request string which is used in our analysis is: "//phpMyAdmin//scripts/setup.php".

25. *Mean Length* of all request substrings (within a session)
26. *Median Length* of all request substrings
27. *Minimum Length* is the length of the shortest request substring
28. *Maximum Length* is the length of the longest request substring
29. *Standard Deviation of Length* of all the request substring

    The following five features are based on the request parameters. HTTP request parameters are usually used to pass data in web applications, that is, when data needs to be passed from the client to the server. One example of a request used to pass parameters is: "http://www.examplesite.com/login?username=foo&password=bar" Usually

the number of parameters passed are separated with ampersand "&" or semicolon ":". The number of parameters passed with this example request is 2.

30. *Mean Number of Parameters* passed to an application (within a session)

31. *Median Number of Parameters* passed to an application

32. *Minimum Number of Parameters* passed to an application

33. *Maximum Number of Parameters* passed to an application

34. *Standard Deviation of Number of Parameters* passed to an application

35. *robots.txt*: whether a robots.txt file was been accessed in any request

36. *Night*: whether the session was between 12am to 8am (local time)

37. *Remote Sites Injected* - whether there is a remote site injection in at least one request

38. *Semicolon Used* - whether a semicolon was used to divide the multiple passed attributes to an application in in some of the requests

    The following features are based on whether there is any specific characters within a request string which might be associated with malicious activity.

39. *Unsafe Characters* - whether there is a string containing suspicious encoding in some of the requests

40. *Reserved Characters* indicates whether a reserved character was used in some of the requests

41. *ASCII Control Characters* indicates weather an ASCII control character was used in some of the requests

42. *Non ASCII Control Characters* indicates weather a Non ASCII control character was used in some of the requests

43. *Invalid Characters* - whether invalid encoding was used in some of the requests

3.4 Types of attackers' activities

In this section, we present descriptive statistics of each data set. Table 3.1 presents what the attackers did in the Web sessions of each data set.

The WebDBAdmin I data set contains 214 Web Sessions, 185 (86.45%) of which were labeled as vulnerability scans and 29 (13.55%) were labeled as attacks. The most dominant types of vulnerability scans were sessions labeled as "phpMyAdmin" and "Static and php-Myadmin". Among the 29 sessions that were labeled as attacks, 18 were password cracking of phpMyAdmin user accounts.

The Web 2.0 I data set contains 1117 Web Sessions, 824 (73.77%) of which were labeled as vulnerability scans and 293, (26.23%) were labeled as attacks. Among the sessions that were labeled as vulnerability scans, the most dominant types of scans were labeled as "Wiki". Another dominant type of scan was when attackers were fingerprinting (accessing) Blog and/or Wiki directly or through the homepage. The most frequent type of attack observed in this data set was Spam on Wiki with 249 sessions out of 293 attack sessions.

The WebDBAdmin II data set contains 549 Web sessions, 513 (93.44%) of which were labeled as vulnerability scans and only 39 (6.56%) were labeled as attacks. Among the sessions that were labeled as vulnerability scans the most frequent were sessions further labeled as Static, phpMyAdmin and phpMyAdmin and Static, in which attackers were fingerprinting (accessing) the Static and/or phpMyAdmin page. Among the 36 sessions that were labeled as attacks, 35 were labeled as "other attacks". These other attacks are different from attacks such as RFI, XSS, SQL injection, and some of them are unknown and this is why they are classified as other attacks.

The Web 2.0 II data set contains 4785 Web sessions, 2059 (43.03%) of which were labeled as vulnerability scans and 2726 (56.97%) were labeled as attacks. Among the types of scans the most frequent were sessions in which browsed static content (html pages, pictures, and video files) and fingerprinting (accessing) Blog or Wiki. Among the 2726 sessions that were labeled as attacks, 1411 were Spam on Blog and 1055 were Spam on Wiki.

- As it can be seen from Figure 3.1, data set Web 2.0 II has the most sessions compared to the other data sets. There were more sessions labeled as vulnerability scans than

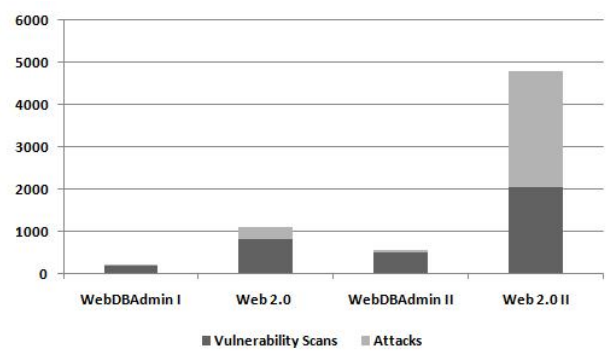

Figure 3.1: Web sessions in data sets

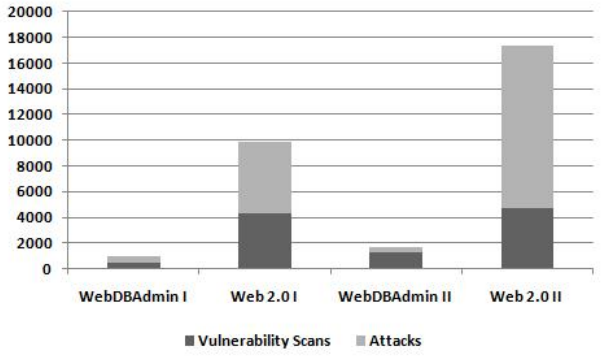

Figure 3.2: Web requests in data sets

attacks in all data sets, except for the Web 2.0 II data set.

- Figure 3.2 shows the distribution of the number of attack requests and vulnerability scan requests for each data set. All data sets have more attack requests than vulnerability scans except for the WebDBAdmin II data set. It is interesting to note that for the WebDBAdmin I and Web 2.0 I data sets, even though there are more attack requests than vulnerability scan requests, the number of vulnerability scan sessions are dominating. This is because a few attack sessions contain many number of requests. For example, for the WebDBAdmin I data set, 5 sessions from the e-mail harvesting category contain 245 attack requests, or 18 sessions labeled as password cracking phpmyadmin user accounts have 260 requests. Similarly, the 4 sessions from data set Web 2.0 I that are labeled as DoS attack have 3724 requests.

- The majority of attacks that we observed in data set WebDBAdmin I were Password cracking of phpMyAdmin user accounts.

- The majority of attacks that we observed in the Web 2.0 I data set were Spam on Wiki; that is, there were 249 Spam on Wiki sessions out of the 293 attack sessions. Spam on Wiki was also observed in 1055 sessions out of the 2726 attack sessions in the Web 2.0 II data set.

- The most frequent attacks in the Web 2.0 II data set were Spam on Blog (approximately 29%). This type of spam was not nearly as frequent as in the Web 2.0 I data set (only

	WebDBI		Web2.0I		WebDBII		Web2.0II	
	Sessions		Sessions		Sessions		Sessions	
Vulnerability scans: Total	**185**	**86.45%**	**824**	**73.77%**	**513**	**93.44%**	**2059**	**43.03%**
DFind	17	7.94%	24	2.15%	19	3.46%	20	0.42%
Other Fingerprint	14	6.54%	0	0.00%	3	0.55%	2	0.04%
Static	26	12.15%	181	16.20%	305	55.56%	327	6.83%
Blog	0	0.00%	107	9.58%	0	0.00%	690	14.42%
Wiki	0	0.00%	385	34.47%	1	0.18%	922	19.27%
Blog & Wiki	0	0.00%	73	6.54%	0	0.00%	77	1.61%
Static & Blog	0	0.00%	10	0.90%	0	0.00%	1	0.02%
Static & Wiki	0	0.00%	19	1.70%	0	0.00%	3	0.06%
Static & Blog & Wiki	0	0.00%	25	2.24%	0	0.00%	3	0.06%
phpMyAdmin	77	35.98%	0	0.00%	155	28.23%	11	0.23%
Static & phpMyAdmin	51	23.83%	0	0.00%	30	5.46%	3	0.06%
Attacks: Total	**29**	**13.55%**	**293**	**26.23%**	**36**	**6.56%**	**2726**	**56.97%**
DoS	0	0.00%	4	0.36%	0	0.00%	0	0.00%
Password cracking phpMyAdmin user accounts	18	8.41%	0	0.00%	0	0.00%	0	0.00%
Password cracking Blog user accounts	0	0.00%	9	0.81%	0	0.00%	1	0.02%
Password cracking Wiki user accounts	0	0.00%	0	0.00%	0	0.00%	71	1.48%
E-mail harvesting	5	2.34%	0	0.00%	0	0.00%	0	0.00%
Spam on Blog	0	0.00%	23	2.06%	0	0.00%	1411	29.49%
Spam on Wiki	0	0.00%	249	22.29%	0	0.00%	1055	22.05%
RFI	0	0.00%	4	0.36%	1	0.18%	5	0.10%
SQL injection	1	0.47%	2	0.18%	0	0.00%	0	0.00%
XSS	0	0.00%	2	0.18%	0	0.00%	11	0.23%
Other Attacks	5	2.34%	0	0.00%	35	6.38%	172	3.59%
Total	**214**	**100.00%**	**1117**	**100.00%**	**549**	**100.00%**	**4785**	**100.00%**

Table 3.1: Summary of Web Sessions for all data sets

around 2%).

- Posting *spam messages* dominated among the attack sessions on both Web 2.0 data sets. We observed the trend that Spam is a major problem for many servers that host Web 2.0 applications due to their interactive nature. In our case, on both Wiki and Blog, attackers posted text and often links to other Web sites. It is interesting to note that no spam ended on the unadvertised server which indicates the use of search-based

strategies.

- Denial of Service (DoS) attack was observed only in the Web 2.0 I data set. This attack was based on the Microsoft IIS vulnerability, but since this vulnerability was fixed in Windows XP SP1 which ran on our honeypots, the attack was unsuccessful. The attacker used IP-based strategy to reach the servers and although the number of sessions was small, it dominated the number of requests on both advertised and unadvertised servers.

- Attackers browsing static content such as html pages, pictures, and video files and fingerprinting (accessing) Blog and/or Wiki ware observed on both Web 2.0 I and Web 2.0 II data sets.

Chapter 4

Statistical characterization of Session Features

Modeling malicious traffic seems interesting since it can help us characterize attacker behavior. In this chapter, we analyze the following malicious Web session features: Duration, Number of Requests per session, and Bytes Transferred per session. The goal is to study whether these features follow heavy-tailed distributions. Also, we explore whether these features are correlated. Furthermore, we explore whether the number of request and sessions originated from unique source IP (attackers) follow heavy-tailed distributions. Finally, we test whether there is any difference in attackers' behavior observed in the four data sets of sessions.

4.1 Background on fitting a heavy-tailed distribution

Statistical characterization of network traffic has a long tradition. In [19], heavy-tailed distributions were used to describe some features of normal traffic. It appears that there were only very few attempts to statistically model aspects of malicious traffic, such as the distribution of the time between visits of reappearing IPs in [6].

Our interest is to see if the "tails" of malicious Web session features can be modeled with heavy-tailed distributions. Practically, this means that most of the feature values are small, however there are a few feature values that are much higher than the typical feature values.

These few values are making the tail to the right of the feature distribution. For example, most of the sessions on our honeypots have a small number of requests, however there are a few session with many requests. Or, most of the sessions are short in duration, but there are a few that last for hours.

In this section, we summarize the methods used to statistically characterize malicious Web session features. A random variable X [44] with cumulative distribution function $F(x)$ is said to be heavy-tailed if

$$1 - F(x) = x^{-\alpha}L(x) \tag{4.1}$$

where $L(x)$ is slowly varying as $x \to \infty$.

The simplest heavy-tailed distribution is a Pareto distribution with a continuous shape parameter $\alpha > 0$ and a continuous location (scale) parameter $b > 0$. The cumulative distribution function (CDF) for Pareto distribution is:

$$F(x) = P[X \leq x] = 1 - (\frac{b}{x})^{\alpha},\ \mathrm{b} \leqslant x < +\infty \tag{4.2}$$

If $1 < \alpha \leq 2$, the distribution has a finite mean and infinite variance; if $\alpha \leq 1$, the distribution has infinite mean and variance. Thus, if α is less then 2 the distribution has infinite variance meaning that the tails of the distribution go to zero slowly. In practical terms, a random variable that follows a heavy-tailed distribution can *give rise to extremely large values with non-negligible probability.*

To estimate the **tail index** α of a Pareto distribution we use the log-log complementary distribution (LLCD) plots and Hills estimator as in [19], [18], [16]. LLCD plot is a plot of the complementary cumulative distribution function $P[X > x] = 1 - F(x)$ on a log-log axes. Linear behavior for the upper tail is evidence of a heavy-tailed distribution. If this is the case, we select a value for x from the LLCD plot above which the plot seems to be linear. Then we estimate the slope, which is equal to $-\alpha$, using least-square regression.

Another alternative approach to estimate the tail index α is the Hill estimator [44]. It estimates α as a function of the k largest elements in the data set. Let $X_1, X_2, \ldots, X_n$ denote observed values of the random variable X and let $X_{(1)} \geq X_{(2)} \geq \ldots \geq X_{(n)}$ be the ordered

statistics of the data set. We pick $k < n$ and compute the Hill estimator

$$H_{k,n} = \frac{1}{k}\sum_{i=1}^{k} \log X_{(i)} - \log X_{(k+1)}. \tag{4.3}$$

For each value of k we estimate the tail index parameter as $\alpha_{k,n} = \frac{1}{H_{k,n}}$. These estimates are then plotted as a function of k, and if the estimator stabilizes to a constant value this provides an estimate of α.

The absence of such straight line behavior is an indication that the data are not consistent with Pareto-like distribution. In those cases, we fitted alternative distributions. In particular, we fitted the lognormal distribution which is a skewed distribution with two parameters: μ and σ. The lognormal distribution, unlike the Pareto distribution, has a finite variance. The cumulative distribution function (CDF) for two parameter Lognormal distribution is:

$$F(x) = \Phi(\frac{lnx - \mu}{\sigma}),\ 0 < x < +\infty \tag{4.4}$$

where Φ is the Laplace Integral.

4.1.1 Andreson-Darling goodness-of-fit test

The Anderson-Darling [2] is a goodness-of-fit test used to test if a sample of data came from a population with a specified continuous cumulative distribution function $F(x)$. This test is more powerful than the better known Kolmogorov-Smirnov and χ^2 tests, particularly for detecting deviations in the tail of a distribution. Thus, the Anderson-Darling test gives more weight to the tails than the Kolmogorov-Smirnov test.

The Anderson-Darling statistic A^2 is defined as

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)[lnF(X_i) + ln(1 - F(X_{n-i+1}))] \tag{4.5}$$

The null hypothesis is rejected at the chosen significance level α if the test statistic, A^2, is greater than the critical value obtained from a table. For the Pareto test, we use the asymptotic significance points from Table 4.1 that are given in [2].

Significance Level α	Critical Value (Significance Point)
0.1	1.933
0.05	2.492
0.01	3.857

Table 4.1: Significance levels and the critical values

4.2 Analyzing the features of malicious Web sessions

In this section, we present statistical analysis of the following features: Number of Requests per Session, Bytes Transferred per Session and Session Duration for all data sets. We first suspect that the best model for the features is a heavy-tailed distribution, that is, a Pareto distribution. For each data set, we observe a 3D scatter plot in which a point within the cube is a Web session characterized by Session Duration, Number of Requests, and Bytes Transferred. Figures 4.1, 4.2, 4.3, and 4.4, illustrate the Web sessions in 3D space from WebDBAdmin I, Web 2.0 I, WebDBAdmin II, and Web 2.0 II data set, respectively. We see that most of the points (sessions) are close to the origin where the values of all the three features are small; however, there are points such that at least one of the session's features has large values.

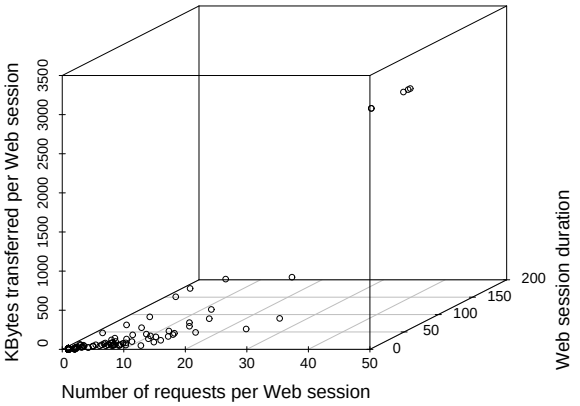

Figure 4.1: 3D scatter plot of Web sessions from the WebDBAdmin I data set

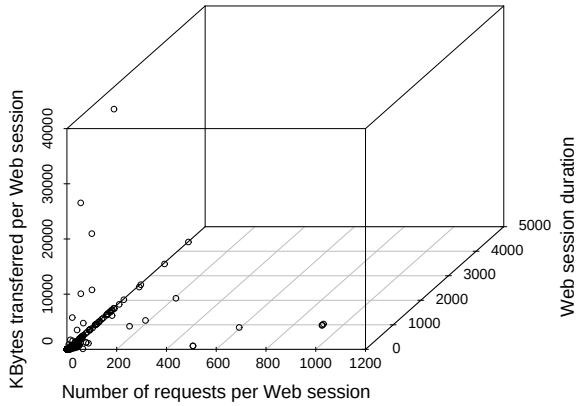

Figure 4.2: 3D scatter plot of Web sessions from the Web 2.0 I data set

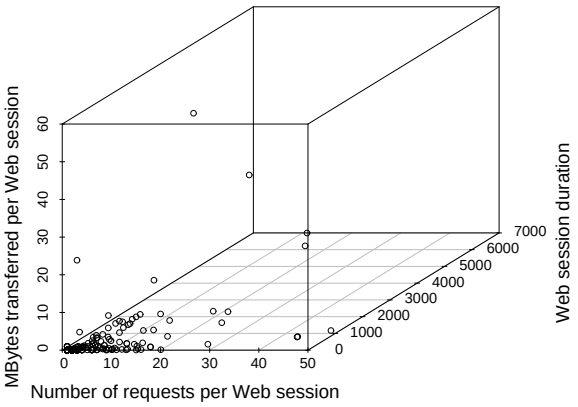

Figure 4.3: 3D scatter plot of Web sessions from the WebDBAdmin II data set

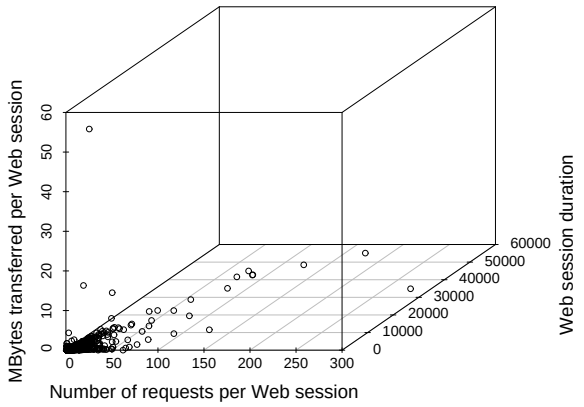

Figure 4.4: 3D scatter plot of Web sessions from the Web 2.0 II data set

4.2.1 Number of Requests per Web Session

The histogram of the Number of Requests per Web Session for the Web 2.0 I data set is shown in Figure 4.5. It can be noticed that most of the Web sessions observed on the Web 2.0 honeypot have a small number of requests, but there are a few of them that have many requests. The minimum, median and maximum for Number of Requests per Web Session for Web 2.0 I feature are: 1/2/1021. Thus, most of the sessions have 1,2,4, or 5 requests, however there were 3 sessions with 1021 requests each. These sessions were DoS attacks which did not have long duration (in time). Besides the DoS sessions, among the sessions in the tail of the Number of Requests were vulnerability scans in which attackers accessed Static content. Some of these sessions had 491 requests, but zero bytes transferred since attackers were looking for applications such as phpMyAdmin that were not deployed on our honeypots [18].

The LLCD plot of the Number of Requests per Session for the Web 2.0 I data set is shown in Figure 4.6. The estimate of α_{LLCD} is 0.73, with a high value of the coefficient of determination $R^2 = 0.9$, which indicates a good fit between the empirical and mathematical distribution.

Since the α_{Hills} is as a function of the k largest elements in the data set, we take the 100 sessions with largest Number of Request to represent the tail of the whole data set. From

the Hill plot of the Number of Requests per Session for the Web 2.0 I data set is shown in Figure 4.7, we observe that α is approximately 0.8 which is consistent with the estimate from the LLCD plot.

Thus, we fit the Pareto distribution in the tail. The Pareto parameters for this tail are: α = 0.8 and b=6. We use the Anderson-Darling goodness-of-fit test to test the null hypothesis that the tail indeed follows a Pareto Distribution. We estimated the A^2 statistics as described in the previous section and we got 1.655712. We take the significance level to be 0.05, thus from the Table 4.1, the corresponding critical value is 2.495. Since 1.655712 $<$ 2.495, we do not reject the null hypothesis that the Number of Request per Session follows a Pareto distribution.

Table 4.2 summarizes the models that we fitted to the tail of the Number of Requests per Web session for all data sets.

	WebDBAdmin I	Web 2.0 I	WebDBAdmin II	Web 2.0 II
min/median/max	1/1/49	1/2/1021	1/2/50	1/2/285
Distribution	Lognormal	Pareto	Lognormal	Pareto
Parameters	$\sigma = 0.583$	$\alpha = 0.8$	$\sigma = 0.512$	$\alpha = 1.6$
	$\mu = 2.495$	$b = 6$	$\mu = 2.556$	$b = 29$
A^2	2.36	1.655712	1.447	0.4695654

Table 4.2: Distribution fitting to the tail of Number of Requests per Session for all data sets

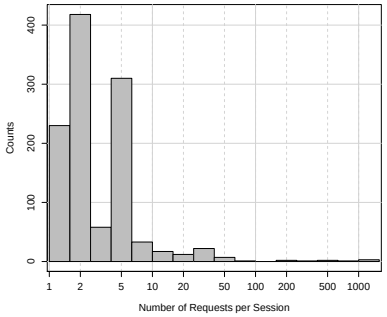

Figure 4.5: Histogram for Number of Requests for the Web 2.0 I data set

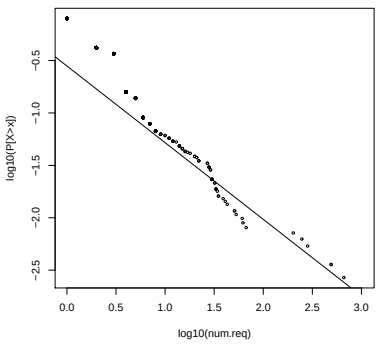

Figure 4.6: LLCD plot for Number of Requests for the Web 2.0 I data set

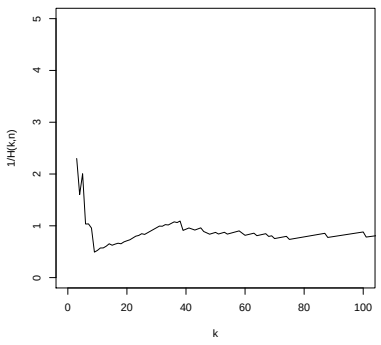

Figure 4.7: Hill plot for Number of Requests for the Web 2.0 I data set

The maximum number of requests per session in the Web 2.0 I and Web 2.0 II data sets were 1021 and 285, respectively, whereas in the WebDBAdmin I and WebDBAdmin II there were 49 and 50 requests, respectively. This means that the longest Web sessions in Web 2.0 honeypot had a much higher number of requests compared to the Number of Requests observed on a phpMyAdmin honeypots.

So, what did the attackers do in sessions with maximum number of requests in each data set? The session with 1021 Number of Requests from the Web 2.0 I data set was due to Denial of Service attack on Microsoft IIS, and the session with 285 Number of Requests from the Web 2.0 II data set was labeled as Spam on Wiki. The 50 requests within a session in the WebDBAdmin II data set were due to password cracking on phpMyAdmin, whereas the session with 49 requests in the WebDBAdmin I data set was labeled as e-mail harvesting.

As it can be seen from Table 4.2, the tails of Number of Requests per Session for both the Web 2.0 I and Web 2.0 II data set follow Pareto distribution with $\alpha < 1$ and $\alpha < 2$, respectively. The hypotheses that the tail of Number of Requests per session follows a Pareto distribution for the WebDBAdmin I and WebDBAdmin II data sets ware rejected. Instead, a lognormal distribution with parameters given in Table 4.2 was fitted into the tails of Number of Requests per session for both WebDBAdmin I and WebDBAdmin II data sets.

It follows that only the tails of Number of Requests per Session for both Web 2.0 I and Web 2.0 II exhibit a heavy-tail behavior.

4.2.2 Bytes Transferred per Web Session

The histogram of Bytes Transferred per Web Session for the Web 2.0 I data set is shown in Figure 4.8. The minimum, median and maximum for this feature are: 0 B /9445 B /37,319,140 B. This indicates a heavy-tailed behavior, since most of the sessions have a small number of bytes transferred and a few had a large number of bytes transferred. The three sessions with most bytes (i.e., 17 MB, 23 MB and 35 MB) were vulnerability scans labeled as Static. Thus, attackers accessing Static content such as videos and pictures contributed towards the large number of bytes transferred within a session. Among the sessions with a large number of bytes transferred were sessions in which attackers did password cracking

attacks on Blog user accounts, DoS sessions, and some vulnerability scans from the "Static & Blog & Wiki" category.

The LLCD plot of Bytes Transferred per Session for data set Web 2.0 I is shown in Figure 4.9. The estimate of α_{LLCD} is 0.56, with a high value of the coefficient of determination R^2 = 0.95.

The tail has 100 sessions that have the largest bytes transferred. From the Hill plot of the Bytes Transferred per Session for data set Web 2.0 I shown in Figure 4.10, we observe that α is approximately 0.56 which is approximately consistent with the estimate from the LLCD plot.

In the tail, we fitted a Pareto distribution with parameteres $\alpha = 0.6$ and b=36,035. The A^2 statistics was 0.8668536. We take the significance level to be 0.05, thus from the Table 4.1 the corresponding critical value is 2.495. Since $0.8668536 < 2.495$, we do not reject the null hypothesis that the Bytes Transferred per Session follows a Pareto distribution.

Table 4.3 summarizes the models that we fitted to the tails of the Bytes Transferred per Web session for all data sets.

	WebDBAdmin I	Web 2.0 I	WebDBAdmin II	Web 2.0 II
	304B/14KB/3MB	0B/9KB/35MB	0B/2.3KB/55MB	0B/20KB/55MB
Distribution	Lognormal (3P)	Pareto	Pareto	Pareto
Parameters	$\sigma = 3.625$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 1.2$
	$\mu = 8.748$	$b = 36035$	$b = 18747$	$b = 162272$
	$\gamma = 41751.0$			
A^2	1.3306	0.8668536	0.7167136	1.313801

Table 4.3: Distribution fitting to the tail of Bytes Transferred per Session for all data sets

The Web 2.0 I, Web 2.0 II and WebDBAdmin II data sets had a few sessions with many bytes transferred. For example, the maximum bytes transferred within a Web sessions for the Web 2.0 I, Web 2.0 II and WebDBAdmin II data sets was 35 MB, 55 MB and 55 MB, respectively. However, the maximum number of bytes transferred per session in the WebDBAdmin I data set was only 3MB which is much less compared to the other data sets. This is why we reject the hypothesis that Bytes Transferred per Web Session follows a Pareto

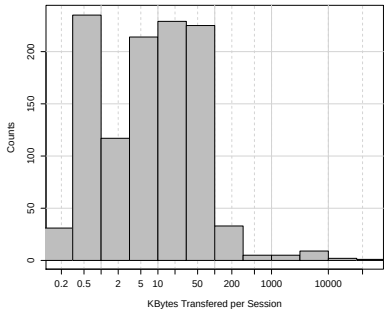

Figure 4.8: Histogram for KBytes Transferred for the Web 2.0 I data set

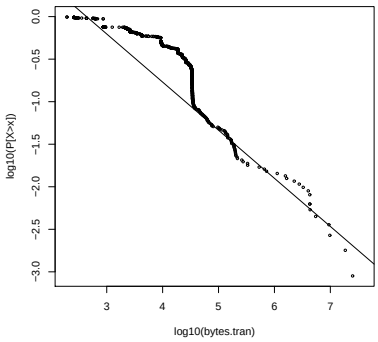

Figure 4.9: LLCD plot for Bytes Transferred for the Web 2.0 I data set

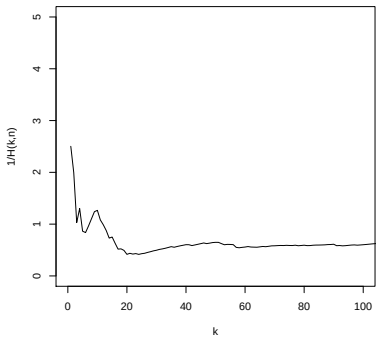

Figure 4.10: Hill plot for Bytes Transferred for the Web 2.0 I data set

distribution for the WebDBAdmin I data set, whereas for all the other data sets, we do not reject that hypothesis. We assume that the reason for smaller number of bytes transferred is the fact that the WebDBAdmin I data set was collected from a configuration with less static content such as figures, videos and so on.

Thus, Bytes Transferred per Web Session for the Web 2.0 I, Web 2.0 II and WebDBAdmin II data sets are modeled with Pareto distribution, whereas for data set WebDBAdmin I we fitted Lognormal distribution with parameters shown in Table 4.3.

It is interesting to see what the attacker activities were in the sessions that ended up with maximum bytes transferred for each data set. The session with 55 MB transferred in Web 2.0 II was labeled as Blog and Wiki, whereas the 35 MB transferred per session in Web 2.0 I was due to viewing Static content. The session with 55 MB transferred in data set WebDBAdmin II was labeled as Static and PhpMyAdmin, whereas the session with 3 MB transferred in data set WebDBAdmin I was due to e-mail harvesting activity.

4.2.3 Duration of a Web Session

The histogram of the Session Duration for the Web 2.0 I data set is shown in Figure 4.11. The minimum, median and maximum for this feature are: 0 /2 /4330 seconds. By looking at the histogram, we observe again that most of the sessions are short and only a few of them are long in duration. The longest session lasted 72 minutes and belonged to a

vulnerability scan in the category "Static & Wiki". The second longest session, which lasted 58 minutes, was classified as "Spam on the Wiki". Among the long sessions for the data set Web 2.0 I there were also vulnerability scans from the "Blog & Wiki" and "Static & Blog & Wiki" categories.

The LLCD plot of the Session Duration for data set Web 2.0 I is shown in Figure 4.12. The estimate of α_{LLCD} is 0.56, and the coefficient of determination $R^2 = 0.86$.

From the Hill plot of the Session Duration for data set Web 2.0 I shown in Figure 4.13, we observe that α is approximately 0.6 which is consistent with the estimate from the LLCD plot.

The Pareto parameters for this tail are: $\alpha = 0.6$ and b=14. Now, using the Anderson-Darling goodness-of-fit test, we test the null hypothesis that the tail follows a Pareto Distribution. The A^2 statistics is 2.61853. We take the significance level to be 0.01, since the corresponding critical value is 3.857. Since $2.61853 < 3.857$, we do not reject the null hypothesis that the Session Duration follows a Pareto distribution.

Table 4.4 summarizes the models that we fitted to the tails of Web session Duration for all data sets. As it can be observed from the Table 4.4, the tails of session duration for all data sets was modeled with Pareto distribution. This means that for each data set there are a few sessions that are very long compared to most of the sessions that were short in duration. The maximum Session Duration in the WebDBAdmin I, Web 2.0 I, WebDBAdmin II and Web 2.0 II data sets were 3.2 minutes, 1.2 hours, 1.9 hours and 14.8 hours, respectively. The attacker activities in these long sessions were labeled as CVE attack, Static and Wiki, Static and Spam on Wiki, respectively.

	WebDBAdmin I	Web 2.0 I	WebDBAdmin II	Web 2.0 II
min/median/max	0/0/194 sec	0/2/4330 sec	0/0/6985 sec	0/4/53383 sec
Distribution	Pareto	Pareto	Pareto	Pareto
Parameters	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 1.1$
	$b = 5$	$b = 14$	$b = 40$	$b = 1376$
A^2	1.6921	2.61853	2.490836	2.249344

Table 4.4: Distribution fitting to the tail of Session Duration for all data sets

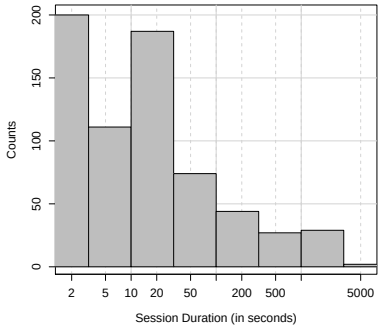

Figure 4.11: Histogram for Session Duration for the Web 2.0 I data set

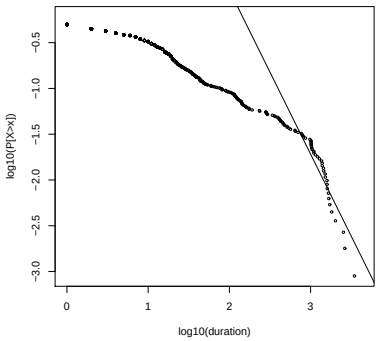

Figure 4.12: LLCD plot for Session Duration for the Web 2.0 I data set

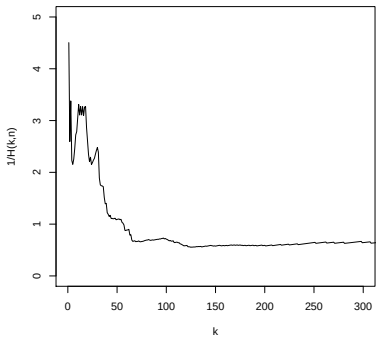

Figure 4.13: Hill plot for Session Duration for the Web 2.0 I data set

4.2.4 Summary of the distribution fitting to the tail of the session features

All three features (Number of Requests, Bytes Transferred, Duration) for sessions in the Web 2.0 I and Web 2.0 II data sets follow a heavy-tailed distribution. In particular, we fitted a Pareto distribution in the tails of each feature and showed that the hypothesis that the "tails" of the features follow a Pareto distribution cannot be rejected for sessions in the Web 2.0 I and Web 2.0 II data sets.

The duration feature for the sessions in WebDBAdmin I data set follows a heavy-tailed distribution. That is, we fitted a Pareto distribution in the tail of duration feature. Pareto distribution is not a good fit to the tail of the Number of Requests and Bytes Transferred features for WebDBAdmin I data set. Instead, the Lognormal distribution is a good fit.

Bytes Transferred and Duration features for sessions in WebDBAdmin II data set follow a heavy-tailed distribution. Specifically, we fitted a Pareto distribution in the tails of Bytes Transferred and Duration features and showed the hypothesis that the "tails" of these features follow a Pareto distribution cannot be rejected for sessions in Web 2.0 II data set. Pareto distribution is not a good fit to the tail of Number of Requests feature for WebDBAdmin II data set. Instead, the Lognormal distribution is a good fit.

In general, what we observed is that most of the malicious sessions are short, with small requests and small bytes transferred. This can be observed in Figures 4.1, 4.2, 4.3 and 4.4,

which show many sessions close to the origin. However, there are sessions that have at least one of these three features with large values. These large values make the tails of the features. This means that sessions can have feature(s) with significantly large values and these kinds of sessions happen with non-negligible probability. For example, most of the sessions are short in duration which can be considered as typical, however we also observed sessions that lasted for hours.

4.3 Correlation between Web session features

In this section, we explore the correlation coefficients between the pairs of session features in all data sets. We use the Spearman's rank correlation coefficient r_s since session features, as shown in the section 4.2, are not normally distributed. In addition, Spearman's correlation coefficient is rather robust to tails, which may consist of up to 20% of the data sample.

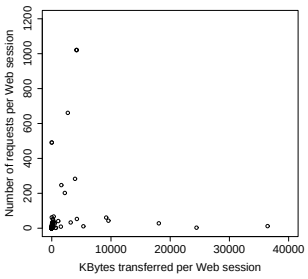

Figure 4.14: KBytes Transferred vs. Number of Requests for the Web 2.0 I data set

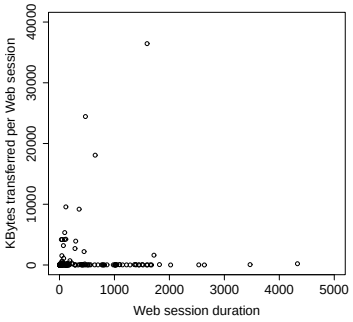

Figure 4.15: Duration vs KBytes Transferred for the Web 2.0 I data set

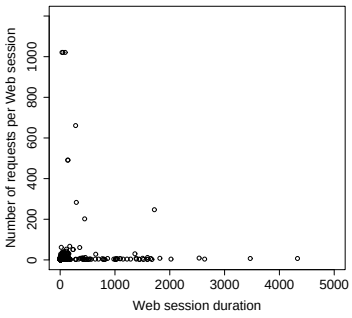

Figure 4.16: Duration vs. Number of Requests for the Web 2.0 I data set

Figures 4.14, 4.15, and 4.16 are 2D projections (i.e., 2D scatter plots for pairs of features) of Web 2.0 I sessions shown in the 3D scatter plot 4.2. The results for Web 2.0 I showed that the Number of Request and Session Duration features are positively correlated with $r_s = 0.76$. The Number of Request and Bytes Transferred features are also positively correlated with $r_s = 0.70$, as well as Bytes Transferred and Session duration with $r_s = 0.72$. All correlations are statistically significant at 0.05 significance level.

These pairs of features are positively correlated since many sessions have small number

of request, bytes transferred, and are short (Figure 4.2). However, we also observe some long sessions, which have very few requests and bytes transferred, or sessions with many bytes transferred which are short and do not have many requests. These sessions are a few and therefore do not affect the Spearman correlation coefficients.

Table 4.5 summarizes the correlation coefficients between the pairs of session features for all data sets. It can be concluded that the pairs of the session features are positively correlated for all data sets. In particular, the correlation values range from moderate, such as 0.41 to high, such as 0.90. For each data set, the pair of features Duration and Number or requests per session has the highest correlation compared to the other pairs.

	WebDBAdmin I	Web 2.0 I	WebDBAdmin II	Web 2.0 II
Bytes & Requests	0.81	0.70	0.66	0.44
Duration & Bytes	0.79	0.72	0.57	0.41
Duration & Requests	0.90	0.76	0.80	0.82

Table 4.5: Spearman correlation coefficient between Web session features for all data sets

4.4 Number of Requests and sessions per unique source IP

Next, we analyze whether Web sessions and requests per unique source IP follow a heavy-tailed behavior. For data set Web 2.0 I, the hypothesis that the number of sessions per unique attacker follows Pareto distribution was rejected. Instead, a lognormal distribution with parameters given in Table 4.6 was a good fit for the number of sessions per unique IP.

For the the Web 2.0 I data set, the Number of Requests per unique source IP (or attacker) follows Pareto distribution with parameters $\alpha = 1.3$ and b=26. Practically, this means that the large number of requests can originate from a small number of attackers with non-negligible probability. These events are rare but they generate the majority of the requests to the honeypot. For example, in the Web 2.0 I data set, 38% of the requests were from one IP address that launched a DoS attack on the Microsoft IIS Server.

Table 4.6 summarizes the distributions of the sessions and requests per unique IP for all data sets.

	WebDBAdmin I		Web2.0 I		WebDBAdmin II		Web2.0 II	
	Sessions	Requests	Sessions	Requests	Sessions	Requests	Sessions	Requests
min/median/max	1/1/24	1/2/98	1/1/58	1/3/3724	1/1/64	1/2/151	1/1/312	1/2/2541
Distribution	Lognormal	Lognormal(3P)	Lognormal	Pareto	Lognormal	Lognormal	Pareto	Pareto
Parameters	$\sigma = 0.675$	$\sigma = 1.15$	$\sigma = 0.70538$	$\alpha = 1.3$	$\sigma = 0.718$	$\sigma = 0.594$	$\alpha = 0.8$	$\alpha = 1.2$
	$\mu = 0.694$	$\mu = 1.63$	$\mu = 2.0034$	b=26	$\mu = 1.7372$	$\mu = 2.864$	b=7	b=22
		$\gamma = 5.77$						
A^2	2.25	0.606	1.5498	0.712	1.713	1.23	0.9	2.33

Table 4.6: Requests and sessions from unique source IP

4.5 Statistical comparison of Web session features across all data sets

In this section we explore whether there is any difference in the features of the malicious Web sessions observed in the four data sets: WebDBAdmin I, Web 2.0 I, WebDBAdmin II and Web 2.0 II. Furthermore, we want to investigate whether the differences of Web session features observed on different Web-based systems (WebDBAdmin II and Web 2.0 II) and collected during the same period of time are statistically significant, that is, are non random. Finally, we want to examine whether there is a significant difference in the features of Web sessions observed on the same Web-based systems (Web 2.0 I and Web 2.0 II) collected during different periods of time.

First, to compare the Web session features across all data sets, we use Kruskal-Wallis [26], a nonparametric one-way ANOVA, which compares medians across three or more groups of observations. As indicated in [26], the test assumes that observations are independent, that all those within a given sample come from a single population, and that the C populations are of approximately the same form. To do this test, we use a function that is implemented in R, which is also explained in [34]. This function gives the p-value and if the p-value is less then or equal to α, we reject the null hypothesis; whereas if the p-value is bigger than α, we do not reject the null hypothesis. The significance level that we use is 0.05.

Therefore, we use the Kurskal-Wallis to test the following hypothesis:

Hypothesis #1:

H_0^1 : There is no significant difference between the median value for Number of Requests per Session across all data sets.

H_A^1 : At least one of the medians is different.

Hypothesis #2:

H_0^2 : There is no significant difference between the median value for Bytes Transferred per Session across all data sets.

H_A^2 : At least one of the medians is different.

Hypothesis #3:

H_0^3 : There is no significant difference between the median value for Session Duration across all data sets.

H_A^3 : At least one of the medians is different.

Figure 4.17 shows the box plots of Web session features for all data sets. We see that the features have a few large values that make the tail to the right, and hence we cannot observe the median values. This is why we "zoom-in" this figure as in Figure 4.18 to see the differences across the medians.

From Figure 4.18 we observe that the median values for Number of Requests per Session are 1, 2, 2, 2, for WebDBAdmin I, Web 2.0 I, WebDBAdmin II, and Web 2.0 II, respectively. We visually see that not all median values are the same. This is confirmed by the test since the p-value obtained from the test is $p < 2.2e - 16$, (which is almost zero) based on a chi-square distribution with 3 degrees of freedom. Since the p-value is less then 0.05 we conclude that we can reject the null hypothesis H_0^1 that the there is no significant difference between the median value for Number of Requests per Session across all data sets.

From Figure 4.18 we observe the median values for Bytes Transferred feature are 4 KB, 9 KB, 2.3 KB, 20 KB for WebDBAdmin I, Web 2.0 I, WebDBAdmin II, and Web 2.0 II, respectively. The test yields a p-value$< 2.2e-16$ which is less then 0.05. This means that we can reject the null hypothesis H_0^2 that there is no significant difference between the median value for Bytes Transferred per Session across all data sets.

From Figure 4.18 we observe that the median values for Session Duration are 0, 2, 0, 4

for WebDBAdmin I, Web 2.0 I, WebDBAdmin II, and Web 2.0 II, respectively. The test yields a p-value$< 2.2e-16$ based on the chi-square distribution with 3 degrees of freedom. Since the p-value is less then 0.05, we can reject the null hypothesis H_0^3 that there is no significant difference between the median value for Session Duration across all data sets.

We can conclude that the differences between median Number of Requests per session across all data sets are non-random and are statistically significant. Also, the differences in median session duration across all data sets are statistically significant and are not due to random chance. Finally, the differences between median Bytes Transferred per session across all data sets are statistically significant and are non-random. Thus, we conclude that we have observed different attacker behavior across all four data sets, and these variations are statistically significant. This behavior may be expected by the fact that there are differences in honeypot configurations, as well as period of data collection.

Next, we examine whether there is a significant difference in the features of Web sessions observed on same Web-based systems (Web 2.0 I and Web 2.0 II) collected during different periods of time. When testing the differences among two groups, we use the *Mann-Whitney Test* [47].

Hypothesis #4:

H_0^4 : There is no significant difference between the median value for Number of Requests per Session between Web 2.0 I and Web 2.0 II data sets.

H_A^4 : There is a difference in the medians between these two data sets.

Hypothesis #5:

H_0^5 : There is no significant difference between the median value for Bytes Transferred per Session Web 2.0 I and Web 2.0 II data sets.

H_A^5 : There is a difference in the medians between these two data sets.

Hypothesis #6:

H_0^6 : There is no significant difference between the median value for Session Duration between Web 2.0 I and Web 2.0 II data sets.

H_A^6 : There is a difference in the medians between these two data sets.

For the hypotheses numbered 4, 5, and 6, the p-value was less then 0.05. Therefore, we can reject the null hypotheses H_0^4, H_0^5, and H_0^6. This means that on the same Web-based

system (Web 2.0 applications) during a different period of time, the attacker behaviors were different and non random.

Finally, we examine whether there is a significant difference in the features of Web sessions observed on different Web-based systems (WebDBAdmin II and Web 2.0 II) collected during the same period of time.

Hypothesis #7:

H_0^7 : There is no significant difference between the median value for Number of Requests per Session between WebDBAdmin II and Web 2.0 II data sets.

H_A^7 : There is a difference in the medians between these two data sets.

Hypothesis #8:

H_0^8 : There is no significant difference between the median value for Bytes Transferred per Session WebDBAdmin II and Web 2.0 II data sets.

H_A^8 : There is a difference in the medians between these two data sets.

Hypothesis #9:

H_0^9 : There is no significant difference between the median value for Session Duration between WebDBAdmin II and Web 2.0 II data sets.

H_A^9 : There is a difference in the medians between these two data sets.

For the hypothesis numbered 7, 8, and 9, the p-value was less then 0.05. Therefore, we can reject the null hypotheses H_0^7, H_0^8, and H_0^8. This means that on a different honeypot configurations (WebDBAdmin II and Web 2.0 II) running on the same period of time, we have different attackers behavior. These differences are statistically significant and non random.

4.5.1 Description of feature characteristics for different types of vulnerability scans and attacks

Motivated by the fact that there are significant differences between the features of Web sessions across all data sets, we further explore the three features that characterize different types of vulnerability scan sessions and attack sessions. For example, we explore whether the features of DoS attack sessions are different from the features of the other types of attack

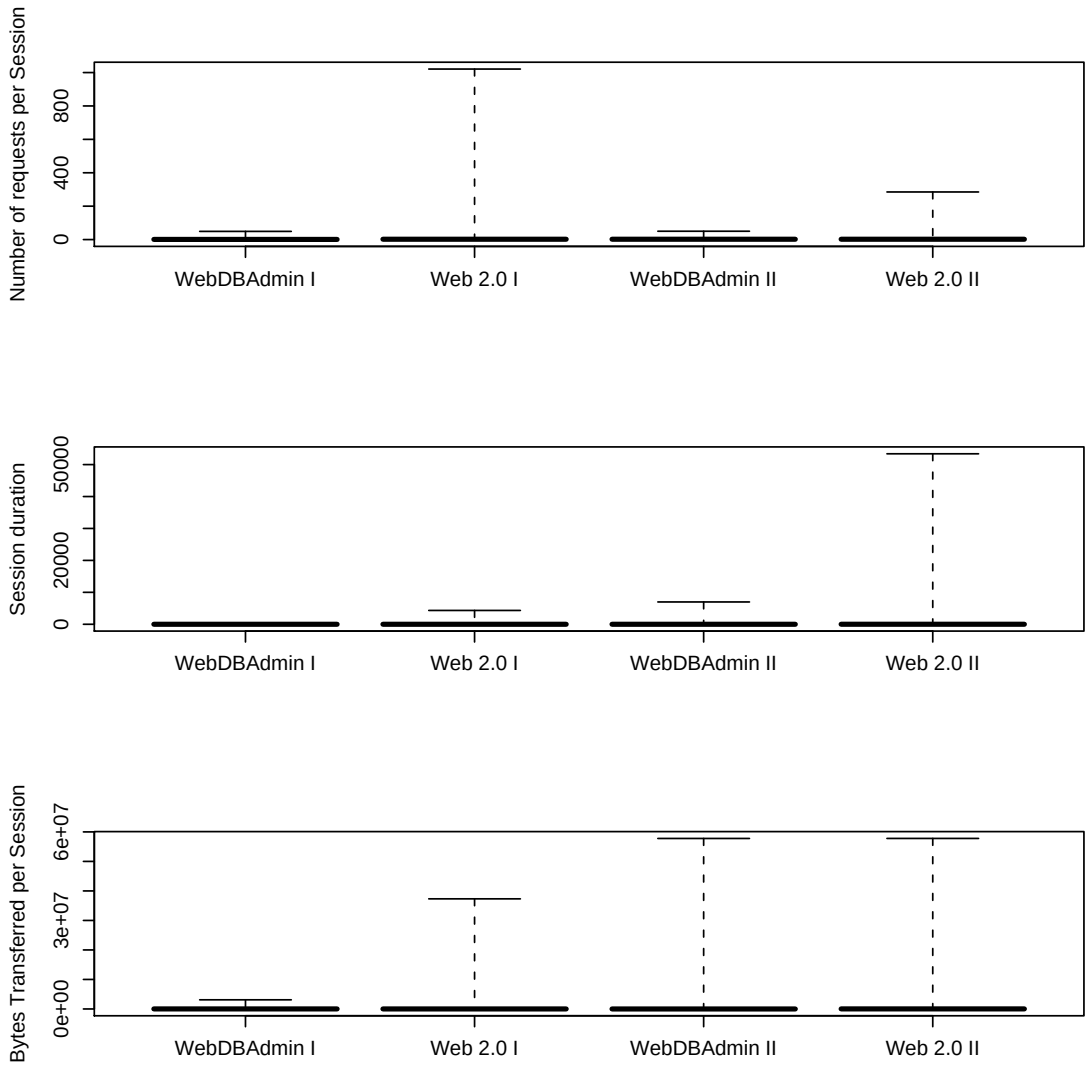

Figure 4.17: Box plots of Web session features

sessions.

Figure 4.19 shows the box plot of Web 2.0 sessions (the left side are sessions from the Web 2.0 I data set, the right side are sessions from the Web 2.0 II data set). As it can be seen from the left side of the Figure 4.19, relatively short sessions with large number of requests are due to DoS attacks, while the sessions with most number of bytes transferred are from the "Static (S)" category. For data set Web 2.0 II, long sessions with many number of requests are due to posting Spam on Wiki, while the sessions with the most number of

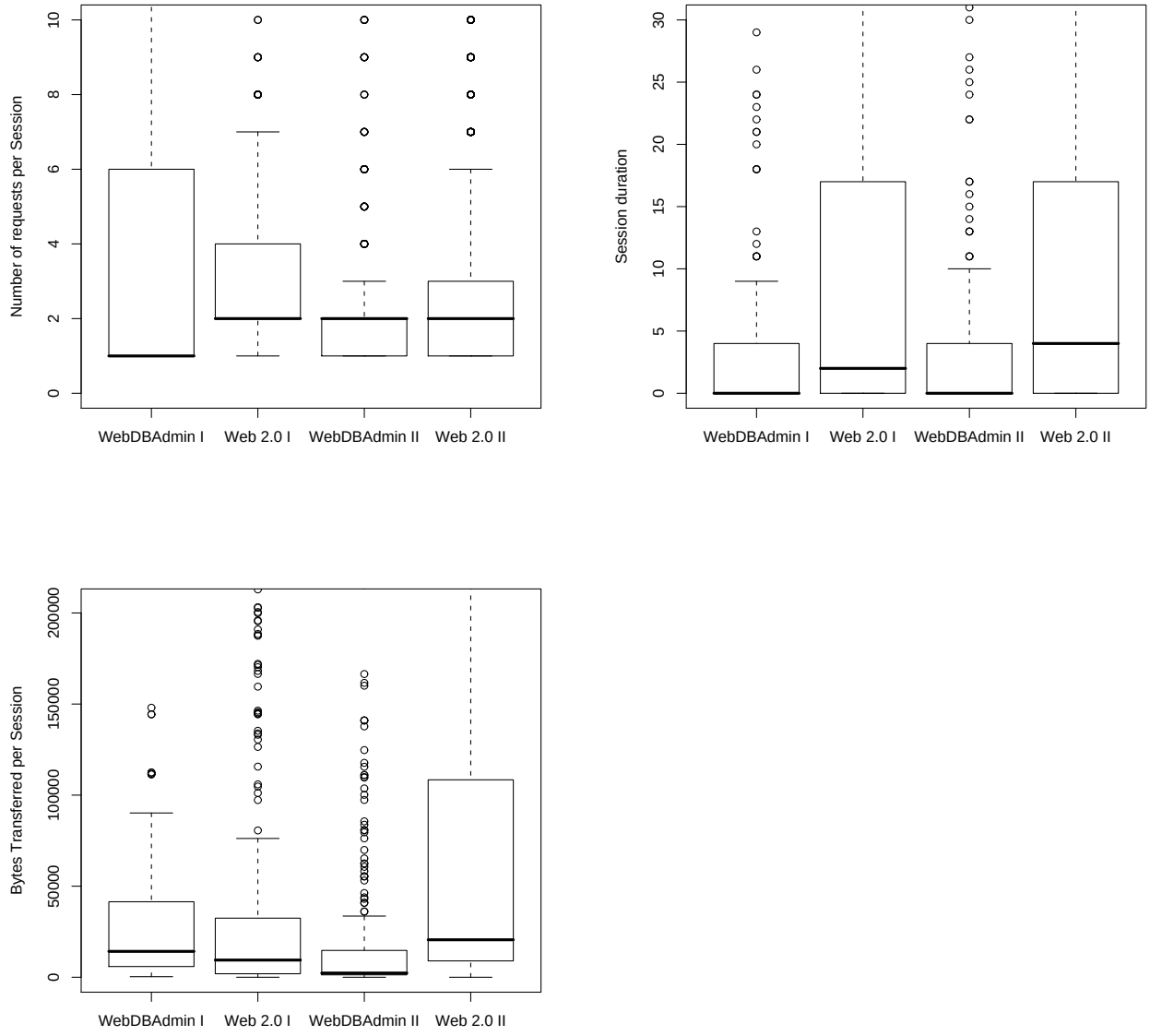

Figure 4.18: Box plots of Web session features

bytes transferred are from the "Blog and Wiki (B&W)" category. It should be noted that for both Web 2.0 I and II data sets, there were no sessions with all three features having significantly large values.

Figure 4.20 shows the box plot for WebDBAdmin sessions (the left side are sessions from WebDBAdmin I data set, the right side are sessions from WebDbAdmin II data set). As it can be seen from left side of Figure 4.19, sessions with most number of bytes transferred and the most number of requests are due to attacks labeled as "e-mail harvesting". Sessions with the longest duration with many request but with small bytes transferred are due to are due to password cracking on PhpMyAdmin user accounts. For WebDBAdmin II data set, the sessions with the most number of bytes transferred are from the "Static and PhpMyAdmin" category. It should be noted that for both WebDBAdmin I and II data sets, there were no sessions with all three features having significantly large values.

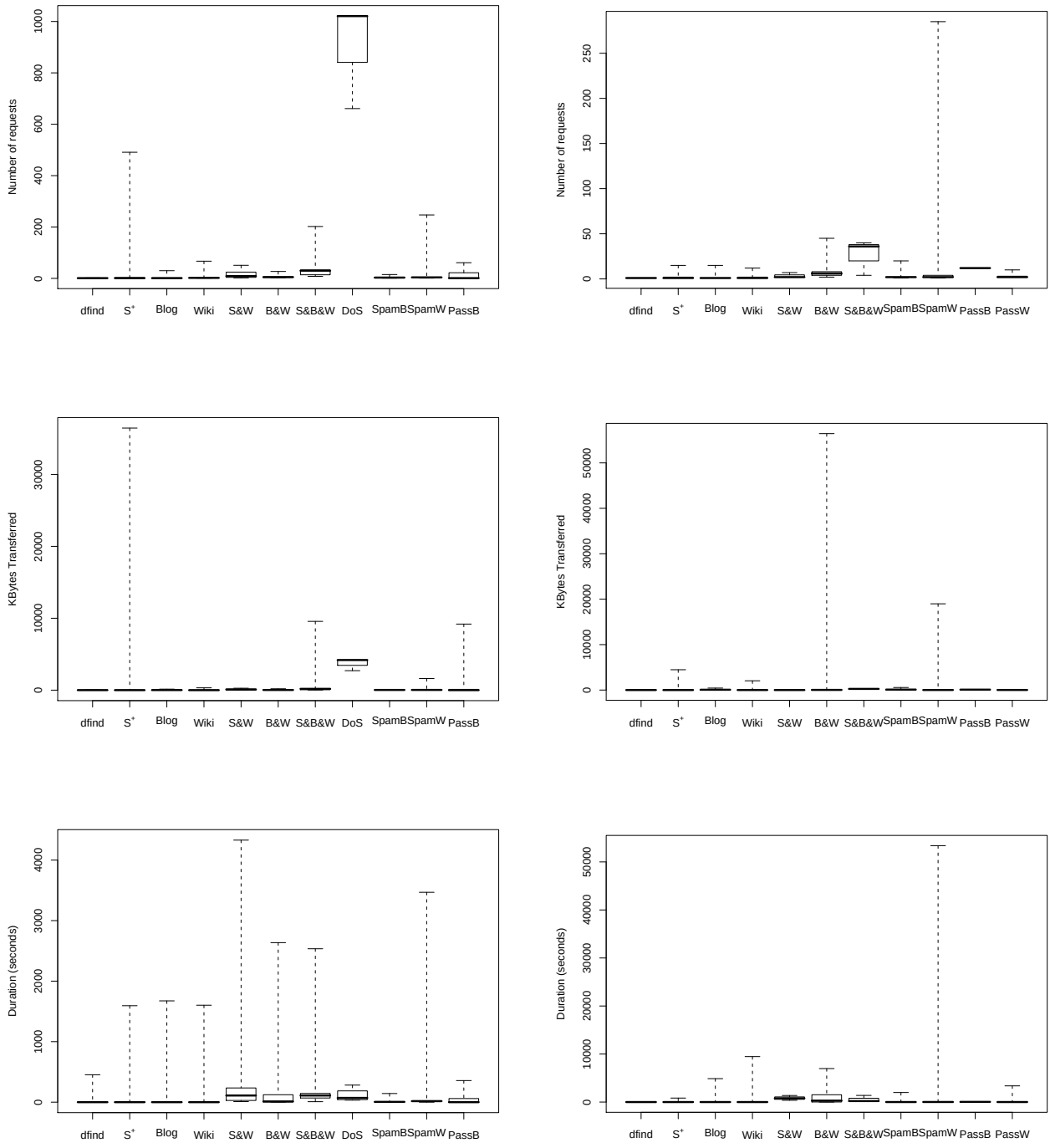

Figure 4.19: Box plots of Web session features: Left side Web 2.0 I, Right side Web 2.0 II

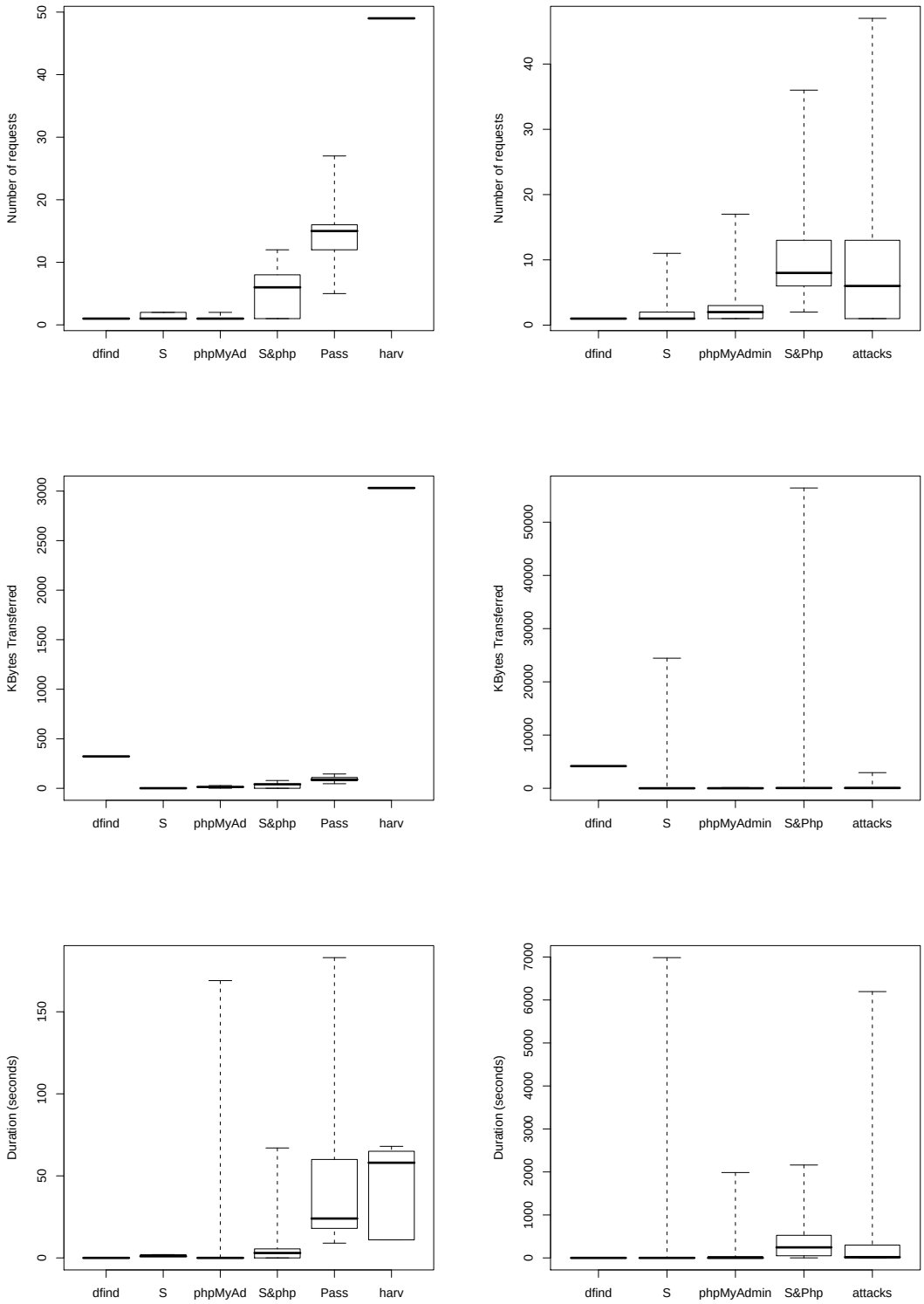

Figure 4.20: Box plots of Web session features: Left side WebDBAdmin I, Right side WebDBAdmin II

Chapter 5

Cluster Analysis of malicious Web Sessions

In this chapter, we present a cluster analysis for malicious Web sessions collected from our advertised honeypots. The purpose of a cluster analysis is to look for similarity groups called clusters in a data set. Objects belonging in the same cluster are similar to each other, whereas objects belonging to different cluster are dissimilar.

Our analysis distinguishes between attacks and vulnerability scans on the Web-based systems. We are interested in seeing if attacks and vulnerability scans that ended up on the Web Server would be separated into different clusters. Furthermore, we are interested to see what characteristics of attacks and vulnerability scans on a particular Web System contribute to better separation between them.

5.1 Feature Normalization

In this section, we explain the feature normalization process. In our analysis, the ranges of the features differ significantly. Some features take continuous values whereas other take binary values such as zero or one. We apply Min Max Normalization, since we want each feature to lie within the new range $[0, 1]$ and the underlying distribution of the feature within the new range of values to remain the same [41]. We use the function "mmnorm" from the "dprep" R-package.

As it can be seen from formula (5.1), each value of a feature is subtracted by the minimum value of that feature, and then this value is divided by the difference of the range of that feature. These new values are multiplied by the new range of the feature and finally added to the new minimum value of the feature. Thus, these operations transform the feature value x_i to lie within the new range $(min_{target}, max_{target})$ which is typically $[0, 1]$.

$$x_i' = [\frac{x_i - min_{value}}{max_{value} - min_{value}}](max_{target} - min_{target}) + min_{target} \tag{5.1}$$

5.2 Feature Selection

Feature Selection (FS) [29] can be described as the search for a subset of original features. The motivation of using feature selection is that it reduces the number of features, removes the irrelevant and noisy features, speeds up the algorithm and hopefully improves the performance of the algorithm. The best feature subset is measured by some evaluation criterion. Depending on what evaluation criterion is used, feature selection algorithms are divided into three categories [29]: the *filter* model, the *wrapper* model, and the *hybrid* model. The filter model uses general characteristics of the data to evaluate the feature subset and does not use any learner (algorithm). The wrapper model uses some learner (mining algorithm) and its performance is used as the evaluation criterion. The hybrid model is a mixture of the filter and the wrapper model.

The several approaches that we use for feature selection are described as follows.

- One method is to rank the features from the most informative to the least informative; that is, to measure the information gain for each feature [20]. Then, we pick the top three features that have the highest information gain. This selection method belongs to the filter method since it uses characteristics of the data to evaluate the features and does not use any learning algorithm. Formally, for each feature A_i, the information gain is calculated by the formula (5.2):

$$IG_i = H(C) - H(C|A_i) \tag{5.2}$$

$$H(C) = -\sum_{c \in C} p(c) log_2 p(c) \tag{5.3}$$

$$H(C|A) = -\sum_{a \in A} p(a) \sum_{c \in C} p(c|a) log_2 p(c|a) \tag{5.4}$$

where formula (5.3) and (5.4) give the entropy of the class feature C before and after observing the feature A.

- Next, we use the feature selection method called Sequential Forward Selection (SFS). This method starts with an empty set, and in the first step adds the feature that performs the best. Then in a subsequent step adds another feature that together with the previously added feature(s) gives the best performance, and so forth. In [38], the classifier used for this algorithm is Support Vector Machine. This method is a wrapper since it uses Support Vector Machine classifier to evaluate the subset of features.

- We also use SFS with another classifier called J48. The output of the SFS algorithm is a subset of features. This method is a wrapper since it uses J48 classifier to evaluate the subset of features.

- We also select features from the nodes of decision trees generated for each data set in [38].

- Finally, we use feature selection based on the Sparse K-means clustering suggested in [48]. This is a new algorithm used for feature selection for clustering.

The selected features are then used in the cluster analysis of the sessions. We compare the performance of the K-means clustering when all session features are used and when only a subset of selected features are used. We also explore what feature selection approach will give the best results for each data set.

5.3 K-means algorithm

In this section we describe the K-means algorithm which is used in our cluster analysis. As it is stated in [12], we use K-means algorithm because it is one of the quickest and simplest algorithm that solves the clustering problem. Let $X = \{x_1, ..., x_n\}$ be a data set consisting of n Web sessions. Each Web session is a data point in d-dimensional Euclidian space, $x_i = (f_1, ..., f_d)$, where $f_1, ..., f_d$ are the d feature values of the i-th Web session. The goal is to split the Web sessions into K clusters so that the distance of the n data points from their respective cluster centroids is minimized [23]. Each cluster has a center μ_k which is known as centroid and it can be considered as a representative of the group.

Thus, the input of the K-means algorithm is the nxd data matrix, the number of clusters K, and the initial value of the centroids. The algorithm has the following steps [23]:

1. Initialize K points that will represent the initial group centroids.
2. For each data point estimate the Euclidian distance between that point and each centroid using equation (5.5). Assign that point (session) to the cluster that has the closest centroid.
3. After all points (sessions) have been assigned, recalculate the positions of the K centroids, that is, μ_k is now the average of all points assigned to that cluster.
4. Repeat step 2 and step 3 until the centroids no longer move (they converge)

$$dist(x, y) = (\sum_{i=1}^{n}(x_i - z_i)^2)^{\frac{1}{2}} \tag{5.5}$$

5.4 Selecting the number of clusters

K-means algorithm assumes the number of clusters K to be known a priori. In this section, we explain how we select the number of clusters before we apply the K-means algorithm. To do so, we use the methods proposed in [43]. First, we use the intra-cluster distance measure which is defined as the distance between a point and its cluster centroid and we take the

average of all of these distances, i.e.

$$intra = \frac{1}{N}\sum_{i=1}^{K}\sum_{x\varepsilon C_i}\|x - z_i\|^2 \tag{5.6}$$

where N is the number of sessions (points), K is the number of clusters, and z_i is the cluster centroid of cluster C_i. We also measure the inter-cluster distance, which represents the distance between clusters, and we want this measure to be as big as possible. The formula for inter cluster distance is:

$$inter = min(\|z_i - z_j\|^2),\ \text{i=1,2,...,K-1; j=i+1,...,K} \tag{5.7}$$

The minimum value for the validity measure given with equation (5.8) should help us choose the number of clusters K for the K-means algorithm.

$$validity = \frac{intra}{inter} \tag{5.8}$$

We minimize the intra-cluster distance and since this measure is in the numerator, we minimize the validity measure. We maximize the inter-cluster distance measure, and since this is in the denominator, we again want to minimize the validity measure.

5.5 Assessing performance of K-means

In this section we explain how we evaluate the performance of the K-means algorithm. The output of this algorithm is K clusters of Web sessions. Since we have previous knowledge (label) of each session, we investigate the type of sessions that ended up in each cluster, that is, we count how many attacks or vulnerability scans are in each cluster. The purpose is to assign a label for each cluster. Each cluster is labeled as either attack or vulnerability scan, based on the majority of the sessions in it. That is, if attack sessions are the majority in the cluster, that cluster is labeled as an attack cluster, whereas if vulnerability scan sessions are the majority, we assign a vulnerability scan label to that cluster.

After we have labeled each cluster, we construct the following confusion matrix:

	actual Vulnerability Scan	actual Attack
predict Vulnerability Scan	A	B
predict Attack	C	D

Table 5.1: Confusion Matrix

In our analysis, the values A, B, C, D from Table 5.1 denote true negatives, false negatives, false positives and true positives, respectively. The measures that we compute to assess the performance of the K-means algorithm are:

$$\text{probability of detection (pd)} = \text{recall} = \frac{D}{B+D} \tag{5.9}$$

$$\text{probability of false alarms (pf)} = \frac{C}{A+C} \tag{5.10}$$

$$\text{precision (prec)} = \frac{D}{D+C} \tag{5.11}$$

$$balance = 1 - \frac{\sqrt{(0-pf)^2 + (1-pd)^2}}{\sqrt{2}} \tag{5.12}$$

$$\text{overall accuracy} = \frac{A+D}{A+B+C+D} \tag{5.13}$$

In order to measure the qualities of clustering results of the K-means algorithm for each data set we use the measures defined as follows. The *overall accuracy*, defined by equation (5.13), is the percentage of true negatives and true positives, or in other words what the learner has detected correctly. It should be noted that using only overall accuracy measure can be misleading especially when the data set has uneven class distributions [33]. For the purposes of completeness, we compute the overall accuracy measure even though in our analyzes some data sets have a significantly smaller number of attacks compared to vulnerability scans. *Probability of detection*, which sometimes is referred to as *recall*, is defined by equation (5.9). Note that in our analysis the probability of detection is the probability of detecting an attack, which is the ratio of detected attacks (true positives) to

all attacks. *Probability of false alarms*, defined by equation (5.10), is the the ratio of detected attacks when no attack was present to all vulnerability scans. *Precision*, defined by equation (5.11), is the ratio of true positives to the to the number of true and false positives. The precision determines how many of the identified attacks were correct. *Balance*, defined by equation (5.12), denotes balance between the probability of detection and the probability of false alarm. Ideally, we want the probability of detection to be 1 and the probability of false alarm to be 0, but this is rarely achieved in practice. As it can be observed from the equation (5.12), the *balance* measures the Euclidian distance from this ideal spot of pf=0, pd=1 to a pair of (pf,pd). Then, the balance is normalized by the diagonal in the ROC square $\sqrt{2}$ and subtracted from 1. It follows that the higher balances are closer to the ideal spot of pf=0, pd=1 [33].

5.6 Results of Cluster Analysis

In this section, we first present the clustering results for each data set, then we give the summary tables and compare the results across all data sets. Finally, we give the conclusions drawn from our results.

5.6.1 Clustering results for the WebDBAdmin I data set

In this subsection, we present clustering results of malicious Web sessions from the WebDBAdmin I data set. Figure 5.1 shows the validity measure for a different number of clusters with sessions characterized by all features. As it can be seen from Figure 5.1, the minimum value for the validity measure is when the number of clusters is three, therefore we select K to be three. The minimum value of the validity measure corresponds to the number of clusters that have minimum inter-cluster distance and maximum intra-cluster distance.

As previously discussed, the role of the K-means is to split the malicious sessions into K clusters. Table 5.2 shows the distribution of vulnerability scan and attack sessions in the resulting clusters. Moreover, it shows and compares the clusters created when sessions are characterized by all features and when sessions are characterized only by a few features selected with various techniques. (The number of sessions that are the majority in each

cluster is highlighted in bold. The type of these sessions denotes the label of the cluster.)

First, we discuss the resulting clusters of Web sessions which are characterized by all 43 features.

As it can be observed from the Figure 5.2 and the first row of the Table 5.2, in the case when all features are used, the majority of sessions in Clusters 2 and 3 are vulnerability scans, whereas Cluster 1 contains only attack sessions. All five attack sessions that belong to Cluster 1 are attacks from the the e-mail harvesting category. Clearly, these five attacks are different from the other attacks since they are grouped in one cluster. Most of the vulnerability scans in Cluster 2 belong to categories such as: fingerprinting phpMyAdmin, dfind, other fingerprints, Static, and so on. Cluster 3 includes the password cracking attacks on phpMyAdmin user accounts, however these attacks are dominated by vulnerability scans labeled as Static and phpMyAdmin.

Next, we discuss the clusters of Web sessions which are characterized by three features (i.e.,GET, Number of requests, and Applications) with highest information gain. These clusters can be graphically represented and observed from Figure 5.3. The majority of sessions in Cluster 3 and 4 are attacks, whereas the majority of sessions in Cluster 1 and 2 are vulnerability scans. In particular, Cluster 3 contains all five attacks from e-mail harvesting category (note that these attacks are also separated in one cluster when all features are used), and Cluster 4 contains 13 sessions (72.22% out of all) labeled as password cracking phpMyAdmin user accounts. These attacks when characterized by all features are a minority in a cluster labeled as vulnerability scans.

Next, we discuss the performance of the K-means clustering for data set WebDBAdmin I. Tables 5.3, 5.4, 5.5, 5.6, 5.7, and 5.8 show the actual and predicted attack and vulnerability scan sessions for the WebDBAdmin I data set, in the cases when K-means used all the features of the sessions and in the cases when different feature selection methods are used.

As it can be seen from the third column in Table 5.29 and Figure 5.9, the probability of detection of attack when all features are used to characterize the sessions is 17.00%, and with feature selection methods, such as three features with the highest InfoGain, features from the nodes of the pruned decision trees, SFS-SVM and SFS-J48, the probability of detection improves significantly to 72.41%, 79.00%, 65.52%, and 58.62%, respectively. The probability

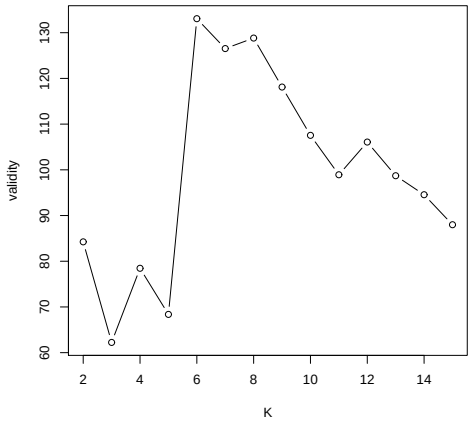

Figure 5.1: Validity for WebDBAdmin I

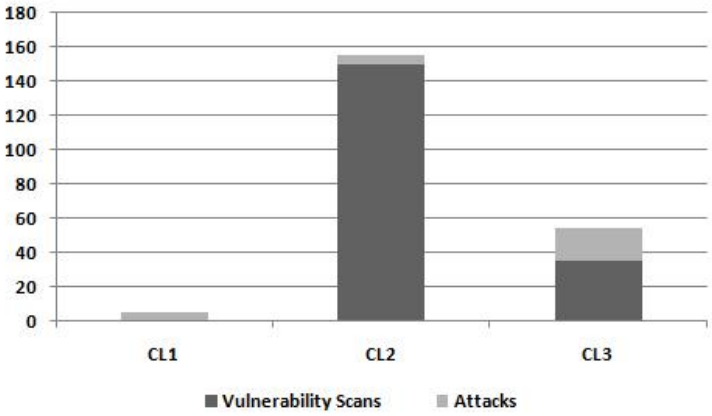

Figure 5.2: Number of Attacks and Vulnerability Scans for each Cluster for the WebDBAdmin I data set

Data set	Session Type	c1	c2	c3	c4	c5	Total
WebDBAdmin I	Vulnerability Scan	0	**150**	**35**			185
all features	Attack	**5**	5	19			29
	Total	5	155	54			214
WebDBAdmin I	Vulnerability Scan	**32**	**152**	0	1		185
feature selection	Attack	8	0	**5**	**16**		29
max InfoGain	Total	40	152	5	17		214
WebDBAdmin I	Vulnerability Scan	**35**	**150**	0	0	0	185
feature selection	Attack	1	5	**13**	**4**	**6**	29
from dec.trees	Total	36	155	13	4	6	214
WebDBAdmin I	Vulnerability Scan	**36**	**40**	0	**109**	0	185
feature selection	Attack	1	4	**11**	5	**8**	29
SFS-SVM	Total	37	44	11	114	8	214
WebDBAdmin I	Vulnerability Scan	**39**	**144**	0	0	2	185
feature selection	Attack	7	5	**9**	**5**	**3**	29
SFS-J48	Total	46	149	9	5	5	214
WebDBAdmin I	Vulnerability Scan	**8**	**60**	**27**	**90**	0	185
feature selection	Attack	5	1	14	4	**5**	29
sparse k-means	Total	13	61	41	94	5	214

Table 5.2: Summary of Clusters of Web Sessions for the WebDBAdmin I data set

of false alarm is very low, from 0% to 1%. The balance measure is the best when features from pruned trees are selected to characterize the sessions in data set WebDBAdmin I. (In Figure 5.9, for the WebDBAdmin I data set, the point that is the closest to the ideal point (0,1).)

Clearly, this shows that when we remove noisy session features, the K-means algorithm better separates attacks from vulnerability scan sessions in the WebDBAdmin I data set. Only the feature selection based on the sparse K-means clustering does not improve the the probability of detection. The overall accuracy of the K-means is fairly high, which is from 88% to 97%.

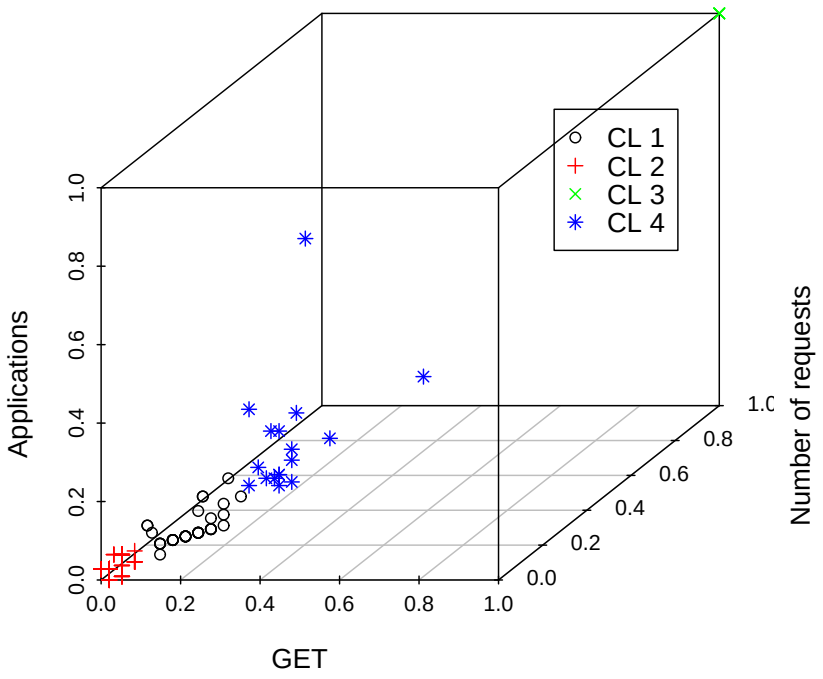

Figure 5.3: 3D scatter plot of clusters of Web sessions from the WebDBAdmin I data set

	actual VS	actual Attack
predict VS	185	24
predict Attack	0	5

Table 5.3: Confusion Matrix for the WebDBAdmin I data set with all features

	actual VS	actual Attack
predict VS	185	6
predict Attack	0	23

Table 5.4: Confusion Matrix for the WebDBAdmin I data set with FS from trees

	actual VS	actual Attack
predict VS	185	10
predict Attack	0	19

Table 5.5: Confusion Matrix for the Web-DBAdmin I data set with SFS-SVM feature selection

	actual VS	actual Attack
predict VS	183	12
predict Attack	2	17

Table 5.6: Confusion Matrix for the Web-DBAdmin I data set with SFS-J48 feature selection

	actual VS	actual Attack
predict VS	185	24
predict Attack	0	5

Table 5.7: Confusion Matrix for the Web-DBAdmin I data set with sparse K-means clustering

	actual VS	actual Attack
predict VS	184	8
predict Attack	1	21

Table 5.8: Confusion Matrix for the Web-DBAdmin I data set with features with highest InfoGain

5.6.2 Clustering results for the Web 2.0 I data set

In this subsection, we present the clustering results for data set Web 2.0 I. Figure 5.4 shows the validity measure for a different number of clusters of sessions characterized by all features. We select the number of the clusters to be 10.

After K-means splits the malicious sessions into K clusters, we investigate the types of sessions in the resulting clusters. Table 5.9 shows the distribution of vulnerability scan and attack sessions for each cluster. Furthermore, one can compare the clusters created when sessions are characterized by all features and when sessions are characterized by only a few features selected with various techniques.

First, we discuss the resulting clusters of Web sessions which are characterized by all 43 features.

As it can be observed from Figure 5.5 and the first row of Table 5.9, in the case when all features are used, the majority of Web sessions in Clusters 1, 6 and 10 are attacks, whereas the dominating sessions in the other clusters are vulnerability scans. Cluster 1 contains 117 (46.98%) of all the sessions in which attackers posted spam on Wiki. Cluster 6 has 61 (24.49%) of all the Spam on Wiki sessions along with 4 CVE attacks. Cluster 10 contains only the Denial of Service (DoS) on Microsoft IIS attacks, which means that the DoS attacks are significantly different from the other attacks since they are grouped in a single cluster.

Next, we discuss the clusters of Web sessions characterized by the three features (i.e., POST, Maximum length and Minimum length) with highest information gain. Figure 5.6

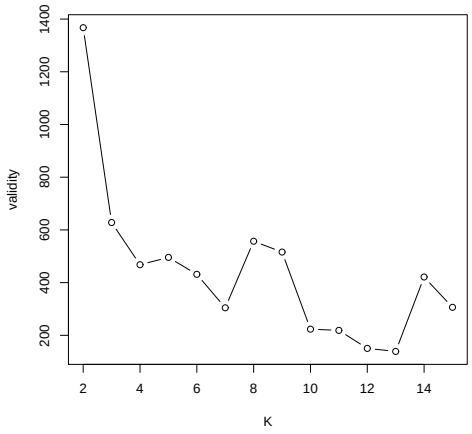

Figure 5.4: Validity for the Web 2.0 I data set

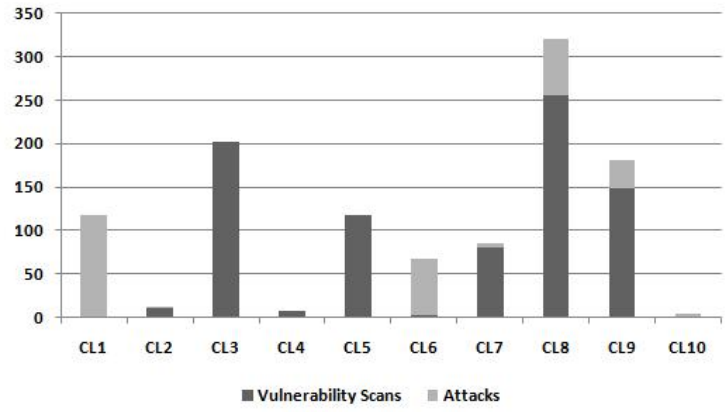

Figure 5.5: Number of Attacks and Vulnerability Scans for each Cluster for the Web 2.0 I data set

illustrates the clusters in 3D space.

The majority of sessions in Cluster 2 and 4 are attacks, whereas the majority of sessions in Cluster 1 and 3 are vulnerability scans. Specifically, the most frequent types of attacks in Cluster 2 are the 169 (67.87%) of all the Spam on Wiki sessions. Similarly, the dominant type of attacks in Cluster 4 are the 80 Spam on Wiki sessions. The majority of sessions in Cluster 1 are vulnerability scans such as the 143 sessions that belong to the Wiki category. Cluster 3 contains most of the vulnerability scan sessions from the Web 2.0 data set. It is interesting to note that the POST feature has nonzero values for most of the sessions in Cluster 2 and 4 (in which the majority are attacks), whereas most of the sessions in Cluster 1 and 3 (in which the majority are vulnerability scans) have zero values of the POST feature.

Another interesting observation from the the third row in Table 5.9, in the case when features are selected from nodes of pruned decision tree, is that Cluster 5 contains 99% of all the Spam on Wiki sessions. It follows that the spam on Wiki attacks are significantly different from the other sessions since almost all are grouped in a single cluster. Cluster 4 contains all the DoS attack Web sessions.

Next, we discuss the performance of the K-means algorithm for data set Web 2.0 I. The confusion matrices given in Tables 5.10, 5.11, 5.12, 5.13, 5.14, and 5.15 show the actual and predicted attacks and vulnerability scans for the Web 2.0 I data set, in the case when

Dataset	Session Type	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	Total
Web 2.0 I	Vulnerability Scan	0	**11**	**202**	**8**	**118**	2	**80**	**255**	**148**	0	824
all features	Attack	**117**	2	0	0	0	**65**	6	66	33	**4**	293
	Total	117	13	202	8	118	67	86	321	181	4	1117
Web 2.0 I	Vulnerability Scan	**178**	4	**642**	0							824
feature selection	Attack	5	**171**	31	**86**							293
max InfoGain	Total	183	175	673	86							1117
Web 2.0 I	Vulnerability Scan	**601**	1	**172**	0	29	**21**					824
feature selection	Attack	24	**2**	10	**4**	**251**	2					293
from dec.trees	Total	625	3	182	4	280	23					1117
Web 2.0 I	Vulnerability Scan	2	1	**821**								824
feature selection	Attack	**4**	**225**	64								293
SFS-SVM	Total	6	226	885								1117
Web 2.0 I	Vulnerability Scan	**783**	41									824
feature selection	Attack	36	**257**									293
SFS-J48	Total	819	298									1117
Web 2.0 I	Vulnerability Scan	**260**	**111**	**130**	**204**	**119**	0					824
feature selection	Attack	70	35	6	0	0	**182**					293
sparse k-means	Total	330	146	136	204	119	182					1117

Table 5.9: Summary of Clusters of Web Sessions for the Web 2.0 I data set

K-means used all session features and in the cases when different feature selection algorithms are used. Using these matrices, we calculate probability of detection, the probability of false alarm, precision, balance and overall accuracy.

	actual VS	actual Attack
predict VS	822	107
predict Attack	2	186

Table 5.10: Confusion Matrix for the Web 2.0 I data set with all features

	actual VS	actual Attack
predict VS	794	36
predict Attack	30	257

Table 5.11: Confusion Matrix for the Web 2.0 I data set with FS from trees

	actual VS	actual Attack
predict VS	821	64
predict Attack	3	229

Table 5.12: Confusion Matrix for the Web 2.0 I data set with SFS-SVM feature selection

	actual VS	actual Attack
predict VS	783	36
predict Attack	41	257

Table 5.13: Confusion Matrix for the Web 2.0 I data set with SFS-J48 feature selection

As it can be seen from the fourth column in Table 5.29 and Figure 5.9, the probability of detection of attack when all features are used is 63.00% and with the feature selection methods such as features with highest InfoGain, features from the nodes of the pruned

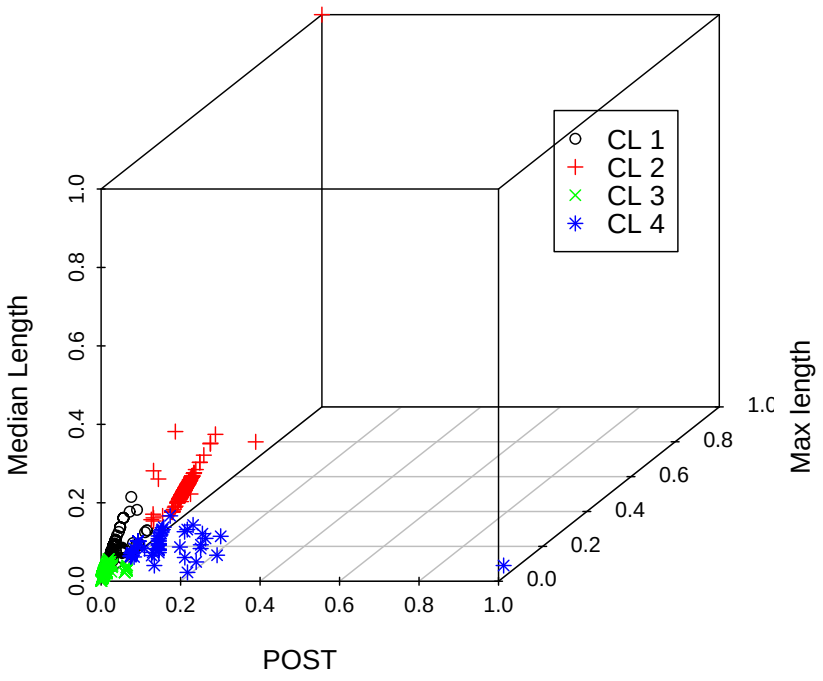

Figure 5.6: 3D scatter plot of clusters of Web sessions from the Web 2.0 I data set

	actual VS	actual Attack
predict VS	824	111
predict Attack	0	182

Table 5.14: Confusion Matrix for the Web 2.0 I data set with sparse K-means clustering

	actual VS	actual Attack
predict VS	820	36
predict Attack	4	257

Table 5.15: Confusion Matrix for the Web 2.0 I data set with with features with highest InfoGain

decision trees, SFS-SVM, SFS-J48, and sparse clustering, the probability of detection of attack is 87.71%, 87.00%, 78.16%, 87.71%, and 62.12% respectively. This means that when feature selection is used, the probability of detection of attack increases (except for the sparse K-means clustering method), which again shows that when the noisy features are expelled, the K-means algorithm can separate better the attacks from vulnerability scan sessions from the Web 2.0 I data set.

It should be mentioned that the probability of false alarm is very low, from 0% to 4.98%, which is very important for the area of attack detection. The balance value is the best when

features with the highest information gain are used. The overall accuracy of the K-means for the Web 2.0 I data set with reduced number of features is high, achieving 96.41%. It is important to note that the probability of attack detection increases from 63% to 87% with feature selection, and the overall accuracy increases too.

5.6.3 Clustering results for the WebDBAdmin II data set

In this subsection, we present the clustering results for the WebDBAdmin II data set. Table 5.16 shows the distribution of vulnerability scan and attack sessions for each cluster when sessions are characterized by all features and when sessions are characterized only by a few features selected with various techniques.

First, we discuss the resulting clusters of Web sessions which are characterized by all 43 features.

As it can be observed from the first row in Table 5.16, only Cluster 6 is labeled as an attack cluster, whereas in the other clusters vulnerability scan sessions are dominating. Cluster 6 contains 7 (20.58%) of all the attacks that are labeled as "other attacks".

Next, we discuss the clusters created when sessions are characterized by the three features (i.e., Server error, Max length, Median length) with the highest information gain. These clusters are illustrated in Figure 5.7. Vulnerability scan sessions dominate across all four clusters. This means that K-means algorithm cannot separate the attacks from vulnerability scans sessions from the WebDBAdmin II data set. The types of vulnerability scans that are in Cluster 3 are: 304 (out of 305) sessions that belong to the Static category, all the 19 Dfind vulnerability scan sessions, and some fingerprinting phpMyAdmin sessions. The majority of vulnerability scan sessions in Cluster 1 and 2 and 4 belong to the category fingerprinting phpMyAdmin.

Next, we discuss the performance of the K-means algorithm for the WebDBAdmin II data set. The confusion matrices given in Tables 5.17, 5.18, 5.19, 5.20, 5.21, and 5.22 show the actual and predicted attacks and vulnerability scans for the WebDBAdmin II data set, in the case when K-means used all session features and in the cases when feature selection methods are used. The fifth column in Table 5.29 shows the probability of detection, probability of

Dataset	Session Type	c1	c2	c3	c4	c5	c6	c7	c8	Total
WebDBAdmin II	Vulnerability Scan	**13**	**78**	**146**	**160**	**80**	2	**17**	**17**	513
all features	Attack	0	0	12	3	9	**7**	3	2	36
	Total	13	78	158	163	89	9	20	19	549
WebDBAdmin II	Vulnerability Scan	**17**	**46**	**448**	**2**					513
feature selection	Attack	3	4	29	0					36
max InfoGain	Total	20	50	477	2					549
WebDBAdmin II	Vulnerability Scan	**22**	5	**486**						513
feature selection	Attack	2	**14**	20						36
from dec.trees	Total	24	19	506						549
WebDBAdmin II	Vulnerability Scan	**513**	0	0						513
feature selection	Attack	20	**13**	**3**						36
SFS-SVM	Total	533	13	3						549
WebDBAdmin II	Vulnerability Scan	**18**	3	**448**	**44**					513
feature selection	Attack	3	**3**	29	1					36
SFS-J48	Total	21	6	477	45					549
WebDBAdmin II	Vulnerability Scan	**176**	**166**	**171**						513
feature selection	Attack	21	3	12						36
sparse k-means	Total	197	169	183						549

Table 5.16: Summary of Clusters of Web Sessions from the WebDBAdmin II data set

false alarm, precision, balance and overall accuracy when using all features and when using different feature selection methods.

As it can be seen from the Figure 5.9, for the WebDBAdmin II data set, the probability of detection of attack when all features are used is 19.00% and with the feature selection methods such as: features with highest InfoGain, features from the nodes of the decision trees, SFS-SVM, SFS-J48, and sparse clustering, the probability of detection of attack is 0.00%, 38.00%, 44.44%, 8.33%, and 0.00%, respectively. This means that some feature subsets do not lead to better probability of detection of attack, such as for example the three features with the highest InfoGain.

The probability of false alarm for WebDBAdmin II is low, around 0%. The overall accuracy of the K-means for the WebDBAdmin II data set is from 93% to 96%, which is high because of the dominance of vulnerability scans over attacks. Data set WebDBAdmin II is an example of achieving high overall accuracy and low probability of detection of attack. In such cases, using only overall accuracy can be misleading since we have high accuracy up

to 96% and low probability of detection of attack from 0% to 44%.

One possible explanation why attacks are not typically grouped in separate cluster(s) (i.e., why the probability of detection is low) in the case of the WebDBAdmin II data set is the fact that it contains only 36 attacks, out of which 35 are labeled as "other attacks". These "other attacks" are significantly different from each other, and some of them are even unknown.

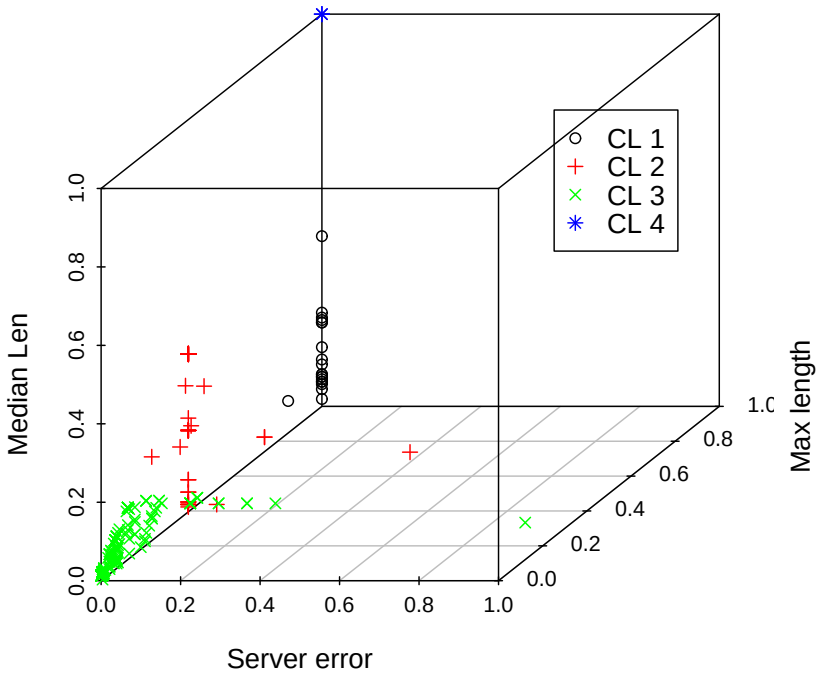

Figure 5.7: 3D scatter plot of clusters of Web sessions from the WebDBAdmin II data set

	actual VS	actual Attack
predict VS	511	29
predict Attack	2	7

Table 5.17: Confusion Matrix for the WebDBAdmin II data set with all features

	actual VS	actual Attack
predict VS	508	22
predict Attack	5	14

Table 5.18: Confusion Matrix for the WebDBAdmin II data set with FS from trees

	actual VS	actual Attack
predict VS	513	20
predict Attack	0	16

Table 5.19: Confusion Matrix for the WebDBAdmin II data set with SFS-SVM feature selection

	actual VS	actual Attack
predict VS	510	33
predict Attack	3	3

Table 5.20: Confusion Matrix for the WebDBAdmin II data set with SFS-J48 feature selection

	actual VS	actual Attack
predict VS	513	36
predict Attack	0	0

Table 5.21: Confusion Matrix for the WebDBAdmin II data set with sparse K-means clustering

	actual VS	actual Attack
predict VS	513	36
predict Attack	0	0

Table 5.22: Confusion Matrix for the WebDBAdmin II data set with features with highest InfoGain

5.6.4 Clustering results for the Web 2.0 II data set

In this subsection, we present the clustering results for data set Web 2.0 II. Table 5.23 shows the distribution of vulnerability scan and attack sessions for each cluster created when sessions are characterized by all features and when sessions are characterized by a few features selected with various feature selection techniques.

First, we discuss the resulting clusters of Web sessions characterized by all 43 features.

As it can be observed from the first row of Table 5.23, in the case when all features are used, the majority of session in Clusters 2, 5, 6, and 8 are attacks, whereas in the other clusters, the dominant type of sessions are vulnerability scans. Cluster 2 contains 471 (33.38%) of all Spam on Blog sessions along with some vulnerability scan sessions such as fingerprinting Blog (214). In Cluster 5, there are 358 (33.93%) of all Spam on Wiki session attacks and 40 sessions that are labeled as password cracking Wiki user accounts. Cluster 6 contains 939 (66.54% out of all) Spam on Blog sessions along with some fingerprinting Blog sessions (345). Cluster 8 contains 560 (53.08%) of all Spam on Wiki along with 18 password cracking on Wiki accounts sessions. All in all, in the clusters which are labeled as attacks, the majority of sessions are ether Spam on Blog or Spam on Wiki along with some password cracking Wiki user accounts.

Next, we examine the clusters when the three features (i.e., POST, Max length, Mean length) with the highest informational gain are used to characterize the sessions in data set Web 2.0 II. These clusters are illustrated in the Figure 5.8. The majority of sessions

in Clusters 1 and 2 are attacks, whereas the majority of sessions in Clusters 3 and 4 are vulnerability scans. In particular, Cluster 1 contains 1408 (99.78%) of all Spam on Blog sessions, which are almost all grouped in one cluster. However, Cluster 1 also contains vulnerability scan sessions that belong to the Blog and/or Wiki category (hence the reason for false alarms). Next, Cluster 2 contains 90.99% of all Spam on Wiki sessions but also contains minority of vulnerability scans labeled as Blog and/or Wiki. Almost all the sessions in Cluster 3 are vulnerability scans from several categories such as Static, Blog, Wiki and so on. Cluster 4 has majority of vulnerability scans from Wiki and Wiki(home page) category. It should be noted that most of the sessions in Clusters 3 and 4 (in which the majority are vulnerability scans) have zero values for the POST feature, whereas for many sessions in Cluster 1 and 2 (in which attacks are the majority) the POST feature has nonzero values. Similarly, this phenomenon for the POST feature is also observed for the Web 2.0 I data set.

Next, we discuss the performance of the K-means algorithm by investigating the confusion matrices given in Tables 5.24, 5.25, 5.26, 5.27, and 5.28, which show the actual and predicted attacks and vulnerability scans for the Web 2.0 II data set, in the case when K-means is based on all session features and in the cases when feature selection methods are used. From the values of these matrices we calculate the probability of detection, probability of false alarm, precision, balance and overall accuracy.

Dataset	Session Type	c1	c2	c3	c4	c5	c6	c7	c8	Total
Web 2.0 II	Vulnerability Scan	**78**	407	**139**	**161**	87	656	**355**	177	2060
all features	Attack	60	**506**	129	14	**407**	**976**	23	**610**	2725
	Total	138	913	268	175	494	1632	378	787	4785
Web 2.0 II	Vulnerability Scan	498	315	**1033**	**214**					2060
feature selection	Attack	**1448**	**1086**	1	190					2725
max InfoGain	Total	1946	1401	1034	404					4785
Web 2.0 II	Vulnerability Scan	**42**	1	66	**587**	1091	273			2060
feature selection	Attack	33	**37**	**889**	230	**1219**	**317**			2725
SFS-SVM	Total	75	38	955	817	2310	590			4785
Web 2.0 II	Vulnerability Scan	**353**	**80**	**1026**	298	2	301			2060
feature selection	Attack	349	56	1008	**429**	**153**	**730**			2725
SFS-J48	Total	702	136	2034	727	155	1031			4785
Web 2.0 II	Vulnerability Scan	494	666	174	**210**	**516**				513
feature selection	Attack	**916**	**1004**	**595**	173	37				36
sparse k-means	Total	1410	1670	769	383	553				549

Table 5.23: Summary of Clusters of Web Sessions from the Web 2.0 II data set

As it can be seen from the last column from Table 5.29 and the Figure 5.9, for the Web 2.0 II data set, the probability of detection of attack when all features are used is 91.00% and when feature selection methods are used, such as the three features with the highest InfoGain, SFS-SVM, SFS-J48, and sparse clustering, the probability of detection of attack is 92.99%, 90.35%, 48.15%, and 92.29%, respectively. It follows that the probability of detection is somewhat higher or approximately the same for all feature selection methods except when SVM-J48 is used. The best results for the Web 2.0 II data set in terms of having low false alarm and high probability of detection are achieved with the three features that have the highest information gain.

Another observation which can be seen in Figure5.9 is that the Web 2.0 II data set when compared to the other data sets, has the highest probability of false alarms, in the range of 29.17% to 69.47%. This is because for the Web 2.0 II data set, K-means places vulnerability scan sessions in clusters in which attacks are the majority. For example, when all session features are used, in Cluster 2 besides Spam on Blog sessions there are also sessions that are labeled as Blog, or in clusters in which the dominating sessions are Spam on Wiki there are also sessions labeled as Wiki. It is obvious that K-means groups spam on Wiki sessions with Wiki sessions and/or Blog sessions. This is why the Web 2.0 II data set has a high rate of false alarms.

The Web 2.0 II data set, when compared to the other data sets, has the lowest overall accuracy from around 57% to 79%.

	actual VS	actual Attack
predict VS	733	226
predict Attack	1327	2499

Table 5.24: Confusion Matrix for the Web 2.0 II data set with all features

	actual VS	actual Attack
predict VS	629	263
predict Attack	1431	2462

Table 5.25: Confusion Matrix for the Web 2.0 II data set with SFS-SVM FS

	actual VS	actual Attack
predict VS	1459	1413
predict Attack	601	1312

Table 5.26: Confusion Matrix for the Web 2.0 II data set with SFS-J48 feature selection

	actual VS	actual Attack
predict VS	726	210
predict Attack	1334	2515

Table 5.27: Confusion Matrix for the Web 2.0 II data set with with sparse K-means clustering

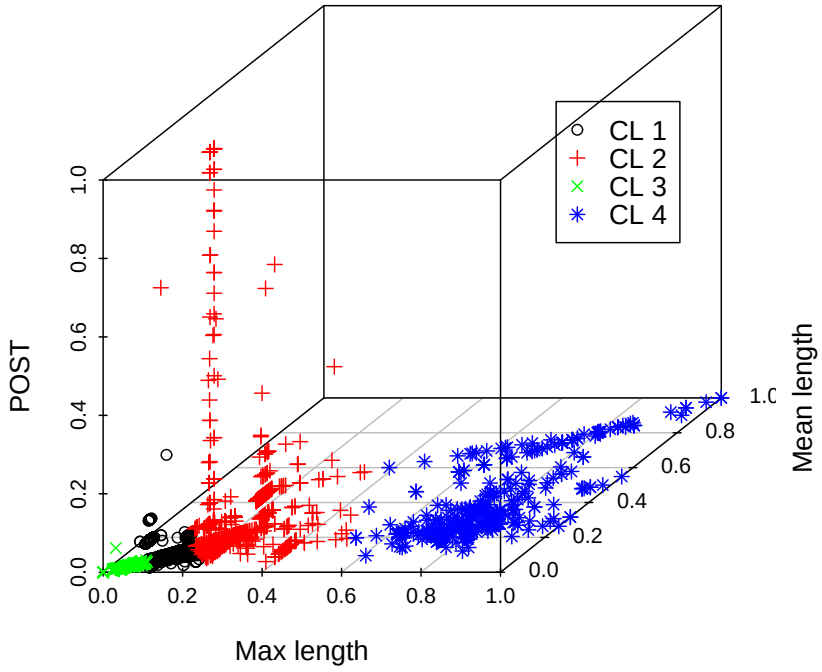

Figure 5.8: 3D scatter plot of clusters of Web sessions from the Web 2.0 II data set

	actual VS	actual Attack
predict VS	1247	191
predict Attack	813	2534

Table 5.28: Confusion Matrix for the Web 2.0 II data set with features with highest InfoGain

5.7 Summary of Clustering Results

In this section, we give the summary highlights based on the discussions of the clustering results. Table 5.29 gives the summary of the clustering results for each data set, in the case when sessions are characterized by all features, and in the cases when sessions are characterized by a few features selected with various feature selection techniques. (Note

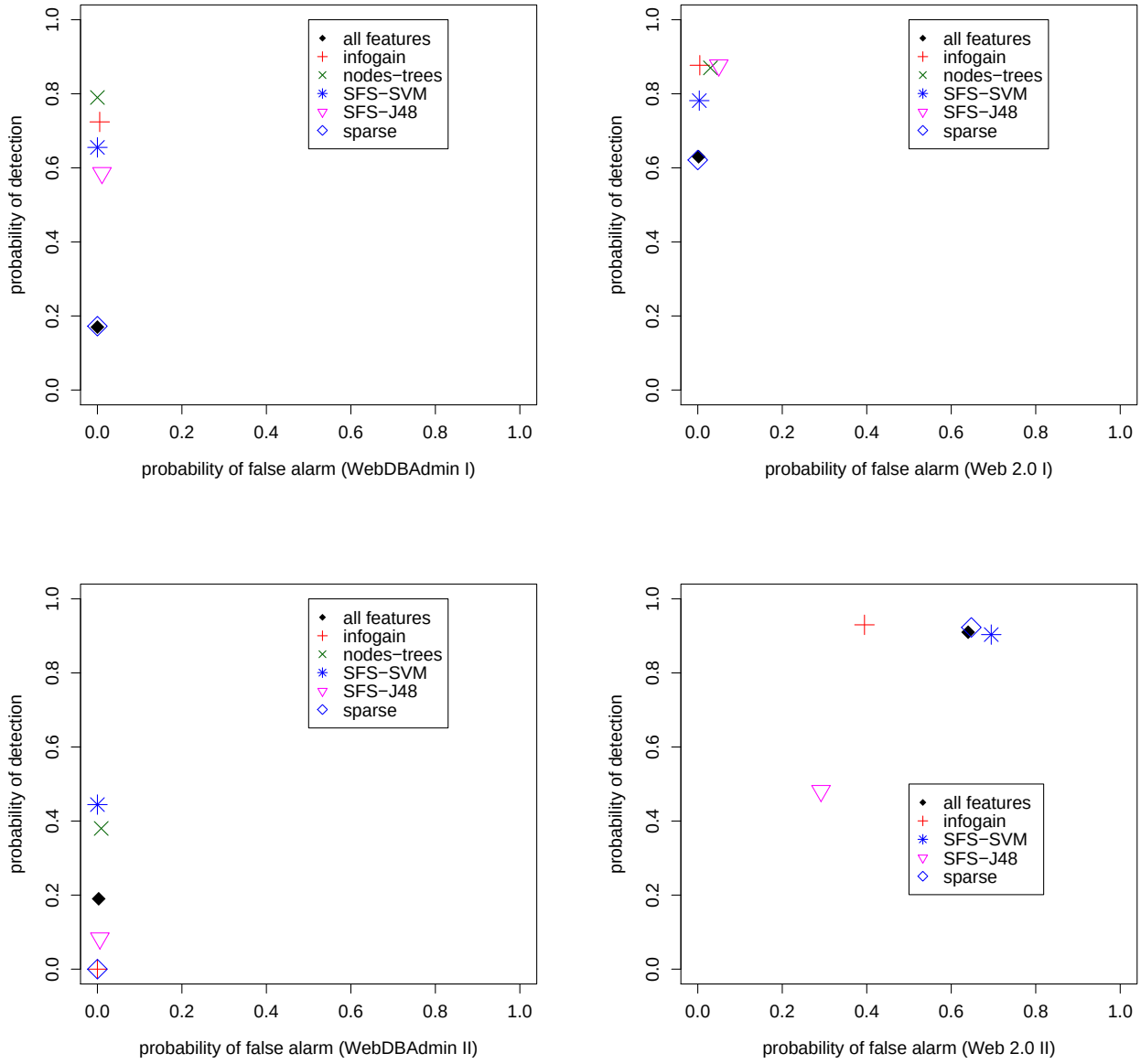

Figure 5.9: ROC curves for WebDBAdmin I, Web 2.0 I, WebDBAdmin II, Web 2.0 II

that the method that gives the best balance value is highlighted in bold, and the method with worst balance value is highlighted in italic.) Figure 5.9 illustrates the probability of false alarm vs probability of detection for each data set when all session features are used and when feature selection methods are used. The following observations can be made:

- As it can be observed from Figure 5.9, and the first row of Table 5.29, in the case when sessions are characterized by all features, the probability of detection of attack session is the best for the Web 2.0 II data set (91.00%). One possible explanation is that the Web 2.0 II data set has much more attacks compared to the other data sets. However, the probability of false alarm is also the biggest for Web 2.0 II whereas for the others is close to 0%. If we compare all data sets in the case when the sessions are characterized by all features, the best clustering results are achieved by the Web 2.0 I data set, in terms of having low false alarms and high probability of detection. (In the figure 5.9, the point that is the closest to the ideal point (0,1).)

- From Figure 5.9, and the second row of Table 5.29, one can see that when a feature selection method based on the three features with the highest information gain is used, the probability of detection, balance, and overall accuracy increase significantly (compared to the case when all features are used) for all data sets, except for WebDBAdmin II. This is because, as mentioned earlier, the attacks labeled as "other attacks" could not be grouped in one cluster due to significantly different characteristics.

- As it can be observed from the the third row of Table 5.29, the number of features selected from the nodes of the pruned decision trees ranges from three to six. Also, from Figure 5.9, one can see that when features are selected with this feature selection method, the probability of detection is increased for all data sets compared to when all features are used. Also, the overall accuracy is improved, as well as the balance value. The pruned tree for the Web 2.0 II data set included almost all the features, which is why we did not use it as a feature selection method for this data set.

- The number of features selected using the Sequential Forward Selection method with SVM, ranges from three to seven. From Figure 5.9, and the fourth row of Table 5.29,

one can see that probability of detection is increased for all data sets (except for the Web 2.0 II data set) compared to when all features are used. It appears that for the Web 2.0 II data set, some Spam on Wiki attack sessions are misclassified. Also, the overall accuracy is improved except for the Web 2.0 II data set.

- The number of features selected using the Sequential Forward Selection method with J48, ranges from two to six. With this feature selection method, as shown in Figure 5.9, and the fifth row of Table 5.29, the probability of detection and the overall accuracy are increased for WebDBAdmin I and the Web 2.0 I data sets compared to when all features are used.

- When Sparse K-means clustering method proposed in [48] is used, the probability of detection is only improved for the Web 2.0 II data set. The overall accuracy of the sparse K-means clustering is the same as when all session features are used.

- It appears that data set Web 2.0 I has the highest balance value compared to the other data sets (in the case when all features are used, and when features are selected with any method). This can also be observed in Figure 5.9, since all the points for the Web 2.0 I data set are close to the ideal point of (0,1). This means that this data set has low probability of false alarm and high probability of detection of attacks. Apparently, attacks and vulnerability scans sessions in the Web 2.0 I data set can be well separated.

- The WebDBAdmin II data set has the lowest balance value compared to the other data sets (especially when features are selected with any method). The WebDBAdmin II data set has very low probability of detection of attacks (see Figure 5.9) when compared to the other data sets. Clearly, the attack and vulnerability scan sessions in the WebDBAdmin II data set cannot be well separated due to the significantly different nature of attacks that belong to this data set.

One general conclusion that can be drawn from the clustering results is that using feature selection methods typically helps to better separate attack Web sessions from vulnerability scan Web sessions. One intuitive explanation is that when redundant and noisy features are

removed from the data sets, the K-means not only works faster, but it gives better results in terms of separating attacks from vulnerability scan sessions.

Another conclusion is that using only the overall accuracy to measure the performance of clustering may be misleading, especially if the data set has uneven class distributions. In that case, using other measures such as probability of detection, probability of false alarm, precision, and balance can be used so that one can understand the results better. For example, the WebDBAdmin II data set has high overall accuracy, but low probability of detection.

Typically, cluster analysis can effectively separate attacks from vulnerability scans of malicious Web traffic, especially if important features are used to characterize the Web sessions. However, the extent of the improvement when only a few important features are used to characterize the sessions depends on the data set and its sessions.

		WebDBAdmin I	**Web 2.0 I**	**WebDBAdmin II**	**Web 2.0 II**
all features	k	3	10	8	8
	pd	17.00%	63.00%	19.00%	91.00%
	pf	0.00%	0.20%	0.30%	64.00%
	prec	100.00%	98.93%	77.77%	65.31%
	bal	*41.00%*	*73.00%*	42.00%	54.00%
	acc	89.00%	90.00%	94.00%	67.00%
	# of features	43	43	43	43
fs-max InfoGain	k	4	4	4	4
	pd	72.41%	87.71%	0.00 %	92.99 %
	pf	0.54 %	0.48 %	0.00 %	39.46 %
	prec	95.45 %	98.46 %	0.00 %	75.70 %
	bal	80.48 %	**91.30 %**	*29.28 %*	**71.65 %**
	acc	95.79 %	96.41 %	93.44 %	79.01 %
	# of features	3	3	3	3
	selected features	9, 1, 18	10, 28, 26	24, 28, 26	28, 25, 10
fs -nodes trees	k	5	6	3	NA
	pd	79.00%	87.00%	38.00%	NA
	pf	0.00%	3.00%	0.90%	NA
	prec	100.00%	89.54%	73.68%	NA
	bal	**85.00%**	90.00%	56.00%	NA
	acc	97.00%	94.00%	95.00%	NA
	# of features	3	7	7	NA
	selected features	10, 18, 34	10, 33, 1, 11, 31, 21, 23	24, 21, 23,32, 7, 26, 18	NA
SFS-SVM	k	5	3	3	6
	pd	65.52%	78.16%	44.44%	90.35%
	pf	0.00%	0.36%	0.00%	69.47%
	prec	100.00%	98.70%	100.00%	76.78%
	bal	75.62%	84.55%	**60.72%**	*50.41%*
	acc	95.33%	94.00%	96.36%	64.60%
	# of features	5	5	3	7
	selected features	34,10,18,23 30	14, 10, 18, 38 40	10, 37, 39	2, 9, 10, 15, 6, 27 33
SFS-J48	k	5	2	4	6
	pd	58.62%	87.71%	8.33%	48.15%
	pf	1.08%	4.98%	0.58%	29.17%
	prec	89.47%	86.24%	50.00%	68.58%
	bal	70.73%	90.63%	35.18%	57.93%
	acc	93.46%	93.11%	93.44%	57.91%
	# of features	2	4	6	10
	selected features	2, 19	10, 28, 31, 33	2, 9, 24, 27 28, 38	8, 2, 9, 36, 21 27, 28, 31, 30, 38
sparse clust	k	5	6	3	5
	pd	17.24%	62.12%	0.00%	92.29%
	pf	0.00%	0.00%	0.00%	64.76%
	prec	100.00%	100.00%	0.00%	65.34%
	bal	41.48%	73.21%	29.29%	53.89%
	acc	88.79%	90.06%	93.44%	67.73%

Table 5.29: Summary of Clustering Results for all data sets

Chapter 6

Conclusion

In this thesis we presented an empirical analysis of attacker activities on Web-based systems with a typical three-tier architecture. The analysis was based on data collected by high-interaction honeypots, during a period of almost four months. The Web applications running on our honeypots included phpMyAdmin, and Web 2.0 applications such as WordPress and Mediawiki. The data collected through the application traffic logs from our honeypots was grouped into four data sets: WebDBAdmin I, Web 2.0 I, WebDBAdmin II, and Web 2.0 II. The data is in a form of *Web Sessions*, each defined as a sequence of requests issued from the same user during a single visit to the Web System. We characterize each session with 43 different features. Our analysis distinguished between *attack* sessions and *vulnerability scans* sessions on the Web-based systems.

The work in this thesis aims to quantify attacker activities which is not a common practice in the cyber security area. The contributions of our work include (1) modeling some features of the malicious Web traffic and (2) cluster analysis of malicious Web traffic.

In the feature modeling part of this thesis we analyzed three features of malicious Web sessions: Number of Requests per session, Bytes transferred per session, and Duration. We also analyzed the number of requests and number of sessions per unique attacker. Only a few attempts were made in the past to statistically model some aspects of malicious traffic; however, neither of the related work included modeling the features of malicious Web sessions, especially from Web 2.0 applications.

Potentially, there is a significant benefit from statistical modeling of the malicious traffic.

For example, these models can be used for simulation of realistic malicious traffic for the purpose of verification and validation of systems' security or to help the intrusion detection process.

The most important observations that we made based on the feature modeling part of this thesis are as follows:

- Features of malicious sessions such as Number of requests, Bytes transferred, and Duration follow skewed distributions, including heavy-tailed. This means that most of the observed malicious Web sessions were short, with a small number of requests and bytes transferred. However, there were sessions that had at least one of these three features with large values. These large values formed the tail of the corresponding feature. This practically means that malicious sessions with extremely large number of request and/or bytes transferred and/or duration can happen with non-negligible probability. It should be mentioned that we did not observe sessions that had significantly large values for all three features.

- Number of requests per unique attacker, as well as the number of sessions per unique attacker follow skewed distributions, including heavy-tailed, with a small number of attackers submitting most of the malicious traffic. Practically, this means that a large number of requests to the honeypot originated from a few attackers. These kind of attackers appeared rarely, but they generated the majority of the malicious Web traffic.

- The Number of request per session, Bytes transferred per session, and Session duration had positive pairwise correlations. The positive correlation was mainly due to the large number of short sessions with small number of requests and bytes transferred.

- We observed differences between the features of malicious Web sessions across all data sets, and these variations were statistically significant and non-random. This behavior may be expected due to the fact that honeypots had different configurations and the data was collected in different periods.

The second part of this thesis included cluster analysis of malicious Web traffic. Not many clustering analysis studies have been done for malicious traffic. Most of the related

work studies did cluster analysis on non-malicious network traffic, or cluster analysis on data sets that had both malicious and non-malicious traffic for the purpose of intrusion detection. We used cluster analysis to explore whether attack and vulnerability scan Web sessions could be separated into different clusters. Some potential benefits from this cluster analysis include identifying attacks within malicious Web traffic consisting of many vulnerability scans and attacks, which is important because attacks are more critical events. Also, cluster analysis helps the process of knowledge discovery on the characteristics of the malicious Web traffic, and thus contributes towards better understanding of the types of malicious behaviors.

The most significant observations from the cluster analysis of malicious Web traffic are as follows:

- Feature selection methods typically helped to better separate attacks from vulnerability scan Web sessions. One intuitive explanation is that when feature selection expelled features that were redundant and noisy, better results were archived by the cluster analysis. However, the extent of the improvement depended on the data set and its sessions.

- Typically, the performance of the K-means algorithm measured in overall accuracy was very high for all data sets. However, one insight from our analysis was that using only overall accuracy can be misleading, especially if the data set has uneven class distributions. In our case, most of the data sets had much more vulnerability scans than attacks. For one data set specifically, the overall accuracy was high but the probability of attack detection was low. Therefore, using additional measures of performance for the K-means such as probability of detection of attack, probability of false alarm, balance and precision can be useful for better comprehension of the clustering results.

Our future work includes further investigation of the performance of several other clustering algorithms such as EM (Expectation-Maximization) clustering, and hierarchial clustering. We also plan to do more formal comparison of the performance of different combinations of clustering algorithms and feature selection methods.

References

[1] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, M. Herrb, "Lessons Learned From the Deployment of a High-Interaction Honeypot," *Proceddings of the Sixth European Dependable Computering Conference*, 2006.

[2] T. W. Anderson and D.A. Darling, "A test of goodness of fit," *J. Amer. Stat. Assn.*, 49:765–769, 1954.

[3] Auckland Data Sets, "http://www.wand.net.nz/wand/wits/auck/."

[4] J. D. Banfield and A. E. Raftery, "Model-based Gaussian and non-Gaussian clustering," Biometrics, 49(3) 803–821, September 1993.

[5] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *SIGCOMM Comput. Commun. Rev.*, 2006.

[6] R. Bloomfield, I. Gashi, A. Povyakalo, and V. Stankovic. "Comparison of emirical data from two honeynets and a distributed honeypot network," *19th Int'l Symp. Software Reliability Engineering*, pp. 219-228, 2008.

[7] P. Cheeseman and J. Strutz. "Bayesian Classification (AutoClass): Theory and Results". *In Advances in Knowledge Discovery and Data Mining, AAI/MIT Press*, USA, 1996.

[8] Collaborative Malware Collection and Sensing. "https://alliance.mwcollect.org"

[9] M. Cukier, R. Berthier, S. Panjwani, and S. Tan, "A statistical analysis of attack data to separate attacks," *In Proceedings of the International Conference on Dependable Systems and Networks* pp. 383–392, IEEE Computer Society, Washington, DC, USA, 2006.

[10] DARPA Intrusion Detection Evaluation Project: "http://www.ll.mit.edu/IST/ideval/"

[11] http://en.wikipedia.org/wiki/Spam_in_blogs

[12] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," *In MineNet 06: Proceedings of the 2006 SIGCOMM workshop on Mining network data* pp. 281-286, New York, NY, USA: ACM Press, 2006.

[13] J. Erman, A. Mahanti, and M. Arlitt,"Internet traffic identification using machine learning techniques," *in Proc. of 49th IEEE GlobalTelecommunications Conference (GLOBECOM 2006)*, San Francisco, USA, December 2006.

[14] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson, "Semi-supervised network traffic classification", *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) Performance Evaluation Review, vol. 35, no. 1*, pp. 369-370, 2007.

[15] M. Ester, H. Kriegel, J. Sander, and X. Xu. "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases withNoise," *In 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD 96)*, Portland, USA, 1996.

[16] K. Goseva-Popstojanova, B. Miller, R. Pantev, and A. Dimitrijevikj, "Empirical Analysis of Attackers' Activity on Multi-Tier Web Systems," 24th *IEEE International Conference on Advanced Information Networking and Applications (AINA-10)*, Pert, Australia, April 2010.

[17] K. Goseva-Popstojanova, S. Mazimdar, and A. Singh, "Empirical Study of Session-based Workload and Reliability for Web Servers," *15th IEEE International Symposium on Software Reliability Engineering (ISSRE 2004)*, Saint-Malo, France, November 2004, pp. 403–414.

[18] K. Goseva-Popstojanova, R. Pantev, A. Dimitrijevikj and B. Miller, "Quantification of Attackers Activities on Servers running Web 2.0 Applications," *9th IEEE International Symposium on Network Computing and Applications (NCA 2010)*, Cambridge, MA, July 2010.

[19] K. Goseva-Popstojanova, F. Li, X. Wang, and A. Sangle, "A Contribution towards solving the Web workload puzzle," *36th IEEE/IFIP Intl Conf. Dependable Systems & Networks*, pp. 505–514, 2006.

[20] M.A. Hall, G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge & Data Engineering*, 2003.

[21] http://hihat.sourceforge.net/linking1.htm

[22] N. Hohna, D. Veitch, and T. Ye, "Splitting and merging of packet traffic: Measurement and modelling," Performance Evaluation, 62:164-177, 2005.

[23] http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html

[24] M. Kaaniche, E. Alata, V. Nicomette, Y. Deswarte, and M. Dacier, "Empirical analysis and statistical modelling of attack processes based on honeypots," *Workshop on Empirical Evaluation of Dependability and Security*, 2006.

[25] Teuvo Kohonen. Self-Organizing Map. Springer-Verlag, New York, 1997.

[26] William H. Kruskal, W. Allen Wallis, "Use of Ranks in One-Criterion Variance Analysis"

[27] K. Labib and V. R. Vemuri, "An Application of Principal Component Analysis to the Detection and Visualization of Computer Network Attacks," *Annals of Telecommunications* France, pp. 218–234. Nov/Dec Issue 2005.

[28] A. Lakhina, M. Crovella and C. Diot,"Mining Anomalies Using Traffic Feature Distributions," *in Proc. of ACM SIGCOMM,* August 2005.

[29] H. Liu, L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Trans. Knowl. Data Eng.* 17(4) pp.491-502, 2005.

[30] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "Neural-Gasnetwork for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Networks*, 4(4) 558–569, July 1993.

[31] A. McGregor, M. Hall, P. Lorier, J. Brunskill, "Flow Clustering Using Machine Learning Techniques," *In Proceedings of the Fifth Passive and Active Measurement Workshop (PAM 2004)*, France, April 2004.

[32] http://www.mediawiki.org/

[33] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Trans. Software Eng.*,vol. 33, no. 1, pp. 2–13, Jan. 2007.

[34] Steven P. Millard, Nagaraj K. Neerchal, "Environmental Statistics with S-Plus"

[35] B. S. Miller, Analysis of Attacks on Web Based Applications, Master's Thesis, WVU, Morgantown, WV, 2009

[36] G. Munz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," *in Leistungs-, Zuverlssigkeitsund Verlsslichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 4. GI/ITG-Wks. MMBne.* Hamburg, Germany, 2007.

[37] http://www.net-security.org/secworld.php?id 6056

[38] Risto Pantev, "Analysis and Classification of Current Trends in Malicious HTTP Traffic", Master's Thesis, WVU, Morgantown, WV, 2011

[39] R. Perdisci,W. Lee, and N. Feamster, "Behavioral clustering of http-based malware and signature generation using malicious network traces," *In USENIX Symposium on Networked Systems Design and Implementation*, NSDI 2010.

[40] http://www.phpmyadmin.net/home_page/index.php

[41] K. L. Priddy, P. E. Keller "Artificial neural networks: an introduction"

[42] Project Malfease. "http://malfease.oarci.net"

[43] S. Ray, R.H. Turi, "Determination of Number of Clusters in K-Means Clustering and Application in Colour Image Segmentation," *ProceedingsConference on Advances in Pattern Recognition and Digital Techniques (ICAPRDT'99)*, Calcutta, India, 137-143, 1999.

[44] S. I. Resnick, "Heavy Tail Modeling of Teletraffic Data," *The Annals of Statistics*, Vol.25, No.5,1997, pp. 1805–1849.

[45] http://www.sans.org/top-cyber-security-risks/summary.php

[46] http://www.sans.org/top-cyber-security-risks/trends.php

[47] S.Siegel, and N.J. Castellan,"Nonparametric Statistics for the Behavioural Sciences," Second Edition, McGraw-Hill, 1988.

[48] D. Witten and R. Tibshirani, "A framework for feature selection in clustering,". *To Appear: Journal of the AmericanStatistical Association*, 2010.

[49] http://wordpress.org/

[50] L. Yingqiu, L. Wei, L. Yun-Chun, "Network traffic classification using K-means clustering,"*In: Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS)*, pp. 360-365. IEEE Computer Society, Washington, 2007.

[51] S. Zander, T. Nguyen, and G. Armitage, "Automated Traffic Classification and Application Identification using Machine Learning," *In Proceedings of IEEE LCN*, Sydney, Australia, Nov, 2005.

[52] S. Zhong, T. M. Khoshgoftaar, and N. Seliya,"Clustering-based Network Intrusion Detection," *International Journal of Reliability, Quality, and Safety Engineering*, 2005.