Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Characterization and classification of malicious Web traffic

Katerina Goseva-Popstojanova[a,*], Goce Anastasovski[a], Ana Dimitrijevikj[a,1], Risto Pantev[b,1], Brandon Miller[c,1]

[a] Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506, USA
[b] Microsoft, Redmond, WA, USA
[c] KeyLogic Systems, Morgantown, WV, USA

ARTICLE INFO

Article history:
Received 25 August 2013
Received in revised form
14 December 2013
Accepted 25 January 2014

Keywords:
Web security
Empirical study
Malicious web sessions
Vulnerability scans
Attacks
Statistical inference
Classification

ABSTRACT

Web systems commonly face unique set of vulnerabilities and security threats due to their high exposure, access by browsers, and integration with databases. This study is focused on characterization and classification of malicious cyber activities aimed at Web systems. The empirical analysis is based on three datasets, each in duration of four to five months, collected by high-interaction honeypots which ran fully functional three-tier Web systems. We first explore the types and prevalence of malicious scans and attacks to Web systems, and the extent to which these malicious activities differ in different periods of time or on Web servers running different services. In addition to descriptive statistical analysis, we include an inferential statistical analysis of the malicious session attributes, such as duration, number of requests and bytes transferred in a session. Then, we use supervised machine learning methods to classify attacker activities to two classes: vulnerability scans and attacks. Our main observations include the following: (1) Some characteristics of the malicious Web traffic were invariant across different servers and time periods, such as for example the dominant use of the search-based strategy for attacking the servers and the heavy-tailed behavior of session attributes. (2) On the other side, servers running different services experienced almost complementary profiles of vulnerability scan and attack types. (3) Supervised learning methods efficiently distinguished attack sessions from vulnerability scan sessions, with high probability of detection and very low probability of false alarms. (4) Decision tree based methods J48 and PART performed better than SVM across all datasets. (5) Attacks differed from vulnerability scans only in a small number of session attributes; depending on the dataset, classification of malicious activities can be performed using from four to six features without significantly affecting learners' performance compared to when all 43 features were used.

* *Corresponding author*. Tel.: +1 3042939691.
E-mail address: Katerina.Goseva@mail.wvu.edu (K. Goseva-Popstojanova).
[1] This work was done while Ana Dimitrijevikj, Risto Pantev, and Brandon Miller were affiliated with West Virginia University.

0167-4048/$ – see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cose.2014.01.006

1. Introduction

Many business and everyday activities heavily rely on Web applications. These applications have many vulnerabilities and typically are targeted by a large number of cyber attacks due to their high exposure, access by browsers, and integration with databases. The 2012 Cost of Cyber Crime Study conducted by the Ponemon Institute reported that the average annualized cost of cyber crime for the 56 organizations included in the study was 8.9 million dollars per year (CyberCrime, 2012). The most costly cyber crimes were caused by denial of service, malicious insiders and Web-based attacks. SANS Institute Annual update of the top 20 security risks (SANS, 2007) stated that almost half of the vulnerabilities discovered in 2007 were Web application vulnerabilities. Another study recently conducted by the WhiteHat Security (WhiteHat, 2012), which was based on assessment of around 7000 Web sites, reported that the average number of serious vulnerabilities found per Web site in 2011 was 79. When it comes to cyber attacks, the Computer Security Institute reported that 92% of respondents to a survey experienced more than ten Web site incidents (Gordon et al., 2005). The cyber attacks have short-term impacts on day-to-day activities of end users, businesses, and governments (e.g., losses due to fraudulent activities, unavailability of computer resources) and long-term impacts (e.g., loss of intellectual property, national security breaches) (Choo, 2011).

The constant introduction of new technologies makes the problem of securing Web systems even more challenging. For example, Web 2.0 technologies enhance information sharing, collaboration, and functionality of the Web, but due to users ability to create content they also provide attackers with a broad range of new vulnerabilities to exploit. These trends clearly illustrate the need for better understanding of malicious cyber activities based on both qualitative and quantitative analysis, which will allow better protection, detection, and service recovery.

To be of practical value, analysis of malicious activities have to account for emerging technologies that typically introduce new types of vulnerabilities. However, there is an evident lack of publicly available, good quality, recent data on cybersecurity threats and malicious attacker activities. Therefore, significant amount of intrusion detection research work in the past was based on publicly available, but outdated datasets, such as the KDD Cup 1999 dataset (KDD, 1999) derived from the DARPA Intrusion Detection Evaluation Project (DARPA, 1999). Even more, most of research work on intrusion detection was focused on development of data mining techniques aimed at constructing a "black-box" that classifies the network traffic on malicious and non-malicious, rather than on discovery of the nature of malicious activities (Julisch, 2002).

Facing the lack of publicly available, recent data on malicious attacker activities, we decided to develop and deploy high-interaction honeypots as a means to collect such data. These honeypots were legitimate servers, which were used to collect information on attacker activities. They were configured in a three-tier architecture (consisting of a front-end Web server, application server, and a back-end database) and had meaningful functionality. Furthermore, they ran standard off-the shelf operating system and applications which followed typical security guidelines and did not include user accounts with nil or weak passwords.

Our experimental setup was based on a sound design, which limits the threats to validity. We developed and deployed honeypots running two different sets of services: one running Web 2.0 applications (i.e., blog and wiki) and another running widely used Web-based database administration software (i.e., phpMyAdmin). The work presented in this paper is based on three datasets collected by these honeypots, which allowed us to compare malicious activities aimed at systems with same configuration during different periods in time (i.e., Web 2.0 I and Web 2.0 II), as well as malicious activities aimed at different system configurations during same period of time (i.e., Web 2.0 II and WebDBAdmin). Each dataset is in duration of four to five months and consists of two sets of data collected by a pair of identical honeypots, one advertised and the other unadvertised. This way we were able to distinguish between malicious activities that used search-based strategy (based on search engines and crawlers) and those that use IP-based strategy (when an attacker scans or attacks an IP addresses without previous involvement of search engines and crawlers).

Using these three datasets we conducted in-depth empirical analysis of attacker activities classified as different types of vulnerability scans and attacks. It should be noted that our datasets represent dynamic information on attacker activities, unlike data extracted from vulnerability databases (e.g., (NVD, 2013)) that are focused on static information related to description of known vulnerabilities and the ways they may be exploited. Correspondingly, we study and model dynamic attacker behaviors aimed at scanning and attacking Web systems, which is different than modeling the discovery process of vulnerabilities present in software applications (see for example the work by Woo et al. (2011)).

In the context of this paper, a Web session is considered as an *attack* session if the attacker attempts to exploit a vulnerability in at least one request in that session. If all requests in the session were used to check for vulnerabilities then the session is considered as *vulnerability scan*. Specifically, we addressed the following research questions related to the characterization of the malicious cyber activities:

RQ1: What types of vulnerability scans and attacks are launched on Web systems? (Sections 4.1 and 4.2)

RQ2: What are the statistical characteristics of malicious Web sessions? (Section 4.3)

RQ3: Are the types and distributions of the vulnerability scans and attacks invariant (1) over time (i.e., for the same system configuration during different periods of time) and (2) across systems running different services? (Sections 4.2 and 4.3)

This paper also addresses the problem of automatic classification of malicious Web sessions to two classes: vulnerability scans and attacks. Both attacks and vulnerability scans are malicious activities. Being able to automatically classify them is important because actual attacks are much more critical events than vulnerability scans. It should be noted that our goal was not to identify whether attacks were preceded by vulnerability scans. Rather, our goal was to distinguish between them, regardless of their temporal order and origin. The fact that our datasets consisted only of malicious Web sessions allowed us to avoid the "needle in the haystack" problem (that is, the need to separate the malicious traffic from a large amount of normal traffic) before we can study the qualitative and quantitative characteristics of malicious cyber activities. Furthermore, although we used machine learning for classification, our goals were very different from standard classification to malicious and non-malicious traffic employed in intrusion detection systems. Instead we automatically classify the collected malicious traffic to vulnerability scans and attacks, which can support intrusion prevention and detection that evolves dynamically with ever changing attacker behaviors. With the increase in the number and diversity of malicious behaviors on Internet, automatic classification of malicious traffic holds enormous potential for improving the protection and resiliency of services and systems. It can be used, for example, to support the generation of attack signatures or to develop attack patterns for testing system resilience to attacks.

We characterized malicious Web sessions with 43 features (i.e., session attributes) which reflect different session characteristics such as the number of requests in a session, number of requests with a specific method type (e.g., GET, POST, OPTIONS), number of requests to dynamic application files, length of requests substrings within a session, and so on. (It should be noted that the terms 'session attributes' and 'features' are used interchangeably in this paper.) In general, our work is based on the hypotheses that different malicious activities exhibit different behavioral patterns, which provides bases for using machine learning methods for their classification. In particular, we explore the following research questions related to classification of malicious traffic:

RQ4: Can supervised machine learning methods be used to automatically distinguish between Web attacks and vulnerability scans? (Section 5.3.1)

RQ5: Do attacks and vulnerability scans differ in a small number of features? Do some learners perform consistently better than other across multiple datasets? (Section 5.3.2)

RQ6: Are the features with best predictive power consistent across different data sets? (Section 5.3.3)

This paper integrates and significantly extends our previous work (Goseva-Popstojanova et al., 2010a, b; 2012). Its main contributions include:

- *Development of a sound experimental approach for collecting malicious cyber activities aimed at Web systems.* Although based on honeypots, our data collection approach is complementary to other existing approaches based on honeypots, such as for example honeypots deployed for the purpose of being compromised in order to analyze the behavior of the adversaries following compromises (Alata et al., 2006; Ramsbrock et al., 2007; Salles-Loustau et al., 2011). Another complementary approach to ours is to use passive monitoring of the unused address space or honeyfarm of active responders (Barford et al., 2010; Vrable et al., 2005), which can be used to collect information on malicious activities such as worm outbreaks and botnet sweeps, but are unlikely to observe attacks that spread along application-specific topologies (i.e., Web).
- *Using three datasets*, which is important for generalizability of observations (i.e., the external validity of the results). Specifically, in addition to the dataset used in our previous work (Goseva-Popstojanova et al., 2010b) (labeled Web 2.0 I in this paper), we used two additional datasets (labeled as Web 2.0 II and WebDBAdmin).
- *Empirical characterization of the malicious Web traffic.* At the transport layer we studied the patterns related to countries of origin, attack sources, vertical and horizontal visits, and distribution of the TCP traffic across different ports. For the malicious HTTP sessions we identified invariant characteristics of vulnerability scans and attacks across our three datasets, as well as explored the differences in malicious behaviors during different time periods and across servers running different services. In addition to descriptive statistics, we used inferential statistics, including hypotheses testing.
- *Automatic classification of malicious Web activities, with a goal to distinguish between attack and vulnerability scan sessions.* Related papers which dealt with using machine learning methods to study malicious behaviors had different goals (i.e., to distinguish among three types of attacks on port 445 (Cukier et al., 2006) or to group malware programs with similar behaviors (Bailey et al., 2007; Bayer et al., 2009; Perdisci et al., 2010)). Unlike related work, we used multiple learners and compared their performance on three datasets. Finally, we identified a small subset of features which were most useful for classification of malicious activities, and thus built the simplest, most efficient model for each data set. None of the related works used feature selection methods.

The rest of the paper is organized as follows. The related work is presented in Section 2. The experimental setup used for data collection is described in Section 3. In Section 4 we present the analysis of malicious traffic at the transport layer, the descriptive statistical analysis of the malicious HTTP traffic broken down to different vulnerability scan classes and attack classes, and inferential statistical analysis of characteristics of malicious Web behaviors. The results of using machine learning methods to automatically classify malicious Web sessions are presented in Section 5. The main observations are summarized in Section 6 and the concluding remarks are given in Section 7.

2. Related work

During the last decade several approaches have been developed and deployed with an intent to monitor and collect real world data about malicious activities on the Internet. These approaches differ in the degree to which they behave like real end-host systems. On one side of the spectrum of malicious data collection approaches are network telescopes, which passively monitor the nonproductive traffic directed at unused IP address space (Moore et al., 2006). The principal strength of network telescopes approach is scalability. However, network telescopes are entirely passive and thus neither can elicit exploits of most attacks nor allow for analysis of malicious activities. To address this deficiency researchers have developed so called active responders which reply to inbound packets (Bailey et al., 2005; Pang et al., 2004). Active responders have been used to identify large-scale events such as worm outbreaks, botnet sweeps, and misconfigurations (Yegneswaran et al., 2005).

Next in the spectrum of different interactivity levels are the low-interaction honeypots that emulate particular operating systems and services, such as Leurre.com (Leurrecom, 2003). Analysis of frequently targeted ports, port sequences, and attack origins, based on data collected by multiple low-interaction honeypots was presented in several works (Chen et al., 2005; Pouget et al., 2005). The analysis conducted by Kaaniche et al. (2006) was based on data collected from 14 low interaction honeypots and included using linear regression to model the number of attacks per unit of time and fitting a distribution to the time between two consecutive attacks. The low-interaction honeypots, however, only allow for performing limited activities and can be easily fingerprinted by the attackers.

In order to provide more realistic experience to the attackers and gather more information about attacks, high-interaction honeypots supported by the Honeynet Project utilize actual operating systems and applications (Honeynet, 1999). The work by McGrew and Vaughn (2006) was based on one high-interaction honeypot and two low-interaction honeypots. The analysis consisted of distribution of attacks across different ports, attacks origins, and description of two instances of successful attacks. Similar analysis based on three high interaction honeypots, each running different operating system, was presented by Dacier et al. (2004). Panjwani et al. (2005) explored whether port scans are precursors to attacks based on network traffic data collected from two high-interaction honeypots. The classification of the network traffic data on port scans, vulnerability scans, and attacks was based on the number of packets per connection and specific types of vulnerability scans and attacks were not identified. The behavior of the attackers who succeeded in breaking into a high-interaction honeypot which had weak passwords for multiple SSH user accounts was studied by Alata et al. (2006). Bloomfield et al. (2008) compared the data collected by Leurre.com and two high-interaction honeypots which ran several unrelated applications. The analysis included most often scanned ports, number of attacking hosts, persistence of attackers, and the distribution of the time between the first packet exchanges from reappearing IPs. Another paper (Berthier et al., 2008) compared the events that targeted similar ports on the same day across data collected by two high-interaction honeypots and data from two global repositories.

A prototype of honeyfarm system called Potemkin attempted to balance between large scale monitoring allowed by active responders/low-interaction honeypots and the high fidelity of data collection enabled by high-interaction honeypots (Vrable et al., 2005). In the heart of Potemkin is binding external requests to physical resources dynamically, only for short periods of time necessary to emulate the execution behavior of dedicated hosts. Honeyfarms like Potemkin are most useful for capturing randomly targeted large-scale attacks (e.g., worms, viruses or botnets) (Vrable et al., 2005). Many non-random attacks (e.g., Web, peer-to-peer, instant messenger, e-mail) that spread along application-specific topologies and carefully select their victims likely will never touch a large-scale honeyfarm (Vrable et al., 2005).

It is also worth mentioning a study based on analysis of firewall logs collected over four months from over 1600 different networks world wide (Yegneswaran et al., 2003). That work included analysis of dominant ports visited by attackers, identification of the worst offenders, and analysis of the worm related traffic.

Significant amount of work in the past was focused on using different data mining methods for intrusion detection, that is, for classification of network traffic to malicious and non-malicious (see for example (Julisch, 2002; Noel et al., 2002; Liao et al., 2013) and references therein). Xu et al. (2008) used data mining and entropy-based techniques to build behavior profiles of Internet backbone traffic. Results showed that a large majority of the clusters fell into three profiles: typical server/service behavior (mostly providing well-known services), typical heavy-hitter host behavior (predominantly associated with well-known services), and typical scan/exploit behavior (frequently manifested by hosts infected with known worms).

Using data mining techniques for classification of some aspects of malicious traffic is an emerging recent trend. Cukier et al. (2006) analyzed the malicious attacks to port 445 based on data collected by two high-interaction honeypots. That work was focused on distinguishing among three types of attacks using the K-means clustering algorithm. Several recent papers were focused on clustering system events collected during execution of sample malware programs (Bailey et al., 2007; Bayer et al., 2009; Perdisci et al., 2010). These papers used anti-virus scanners to label the collected samples and applied single-linkage hierarchical clustering to group the malicious samples in classes with similar behaviors.

The work presented in this paper extends our previous work (Goseva-Popstojanova et al., 2010a, b) in several directions. First, in addition to the dataset used in previous paper (Goseva-Popstojanova et al., 2010b) (labeled Web 2.0 I), in this paper we considered two additional datasets (Web 2.0 II and WebDBAdmin). Using more than one dataset is important for generalizability of observations, i.e., the external validity of the results.

Second, we analyzed the malicious traffic at the transport layer, including countries of origin, attack sources, vertical and horizontal visits, and distribution of the TCP traffic to different ports, which was not studied in our previous work (Goseva-Popstojanova et al., 2010a, b).

Third, the focus of this paper is on comparative analysis of vulnerability scans and attacks which appear in these three datasets, with a goal to identify invariant characteristics associated with malicious activities aimed at Web systems, as well as to explore whether there are differences in malicious behaviors during different time periods and across servers running different services. In addition to descriptive statistics, we used inferential statistics, including hypotheses testing for session attributes across the three datasets.

Finally, we integrated the relevant results of using machine learning algorithms to automatically classify malicious Web activities into two classes (i.e., vulnerability scans and attacks) (Goseva-Popstojanova et al., 2012). This is an important aspect of our work because automatic classification of malicious traffic collected by advertised honeypots with realistic configurations can be used to dynamically characterize ever changing cyber behaviors in support of adaptive intrusion prevention and detection.

3. Experimental setup for data collection

For collecting the data used in this paper, we developed and deployed high-interaction honeypots with two different configurations. For each configuration, the honeypot systems followed the design shown in Fig. 1. The honeywall, which acts as a bridging firewall between the honeypot and the Internet, is an integral part of each honeypot system. Any traffic going to or from the honeypots passed through the honeywall, which logged all packets using TCPDump and then silently forwarded the traffic without modifying the hop count of the packets. The honeywall also limited the outbound connections an attacker can initiate from a honeypot, which reduced the risk of malicious activities originating from a compromised honeypot. The captured network traffic was stored in a central data repository which ran on a separate physical host. We also collected information related to the system activity and various applications running on our honeypots.

The honeypots could be accessed from outside the network on which they resided, as well as by insiders (i.e., users using computers residing in the same network as honeypots). Our main goal was to collect attacker activities that spread along application specific typologies and operating systems, through the Internet, including from the network where the honeypots resided. It should be noted that the only legitimate human users of the honeypots were the members of our research group. Therefore, as any work based on data collected by honeypots, insider attacks by legitimate users who misuse their legitimate access to sensitive information and resources are out of the scope of this work.

Since our main goal was to study the patterns and characteristics of attacker activities, instead of a collection of independent services typical for the related work, each of our honeypots ran a Web based system with a three-tier architecture (i.e., Web server, application server(s), and database). In this paper we use three datasets, collected from two honeypot systems running two different sets of Web services. To allow for sound comparison, both honeypot systems (i.e., Web 2.0 and WebDBAdmin) ran the same operating system Windows XP Service Pack 2, with Microsoft Internet Information Services (IIS) Web server (version 5.1), PHP Server (version 5.0.2), and MySQL database (version 4.1). The static Web content was also identical on both honeypot systems. In addition, on each honeypot we installed the SSHWindows (version 3.8.1p1) server, which is an OpenSSH server for Windows.

The first honeypot system ran two open source Web 2.0 applications: the most widely used wiki software MediaWiki (version 1.9.0), which is used as an application base for Wikipedia, and the most downloaded open source blogging software Wordpress (version 2.1.1). From this configuration, we collected two datasets: *Web 2.0 I* in duration of close to four months (i.e., 119 days) and *Web 2.0 II* in duration of five months (i.e., 154 days).

The second honeypot system instead of Web 2.0 applications ran phpMyAdmin (version 2.9.1.1) as a Web service, which is a popular open source application widely used to handle database administration over the Web. *WebDBAdmin*is the dataset consisting of malicious traffic collected from this honeypot system in duration of five months (i.e., 154 days), during the same time period as the Web 2.0 II dataset. (To minimize the threats to validity we did not include in this paper one of the datasets used in our previous work (Goseva-Popstojanova et al., 2010a), which was collected by a honeypot that also included phpMyAdmin as a Web service, but ran different operating system (i.e., Linux), different Web server (i.e., Apache) and hosted different static content.)

We created multiple user accounts for the operating system, services, and the applications running on each honeypot system. The accounts were for different roles and with varying degrees of usage permissions. In order to prevent simple password cracking attempts from succeeding, all user accounts were assigned strong passwords. Furthermore the root or administrator accounts were restricted so that they can only be accessed locally or from the data collection server.

The 'Home page' of the front-end Web server was a static HTML page which contained links to the Web applications, as well as links to other Web pages that contained static HTML content and included pictures and videos. A random text generator was used to generate random content (consisting of existing English words) for each of the Web applications, so it would appear to attackers that the applications were being actively used. It should be noted that the blog and wiki created this way appeared as legitimate servers because, as our data showed, attackers posted blog and wiki messages using automatic scripts. Web 2.0 applications were configured to accept anonymous submissions, that is, submissions from users which are not logged in. In Wordpress, anonymous users could post comments to blog entries. In MediaWiki, anonymous users had the same permission level as logged in users and could post and edit entries. On each honeypot a MySQL server was also installed and configured similarly to what would be found on a typical Web server. The primary function of the MySQL server was to serve as a back-end for the Web applications. Thus, the MySQL database on the WebDBAdmin honeypot was populated with data and the MySQL server allowed for a user login via phpMyAdmin interface. The MySQL server on the Web 2.0 honeypot contained one database for each of the two Web 2.0 applications, as well as a system database.

A unique characteristic of our experimental set-up is the fact that for each configuration we built two identical honeypots. One of the honeypots was *advertised* using a technique called 'transparent linking', which involved placing hyperlinks pointing to our honeypot on regular, public Web pages, so that the advertised honeypot was indexed by search engines and Web crawlers, but could not be accessed directly by humans. This way we allowed for attacks based on search engines and crawlers (using the so called *search-based strategy*)

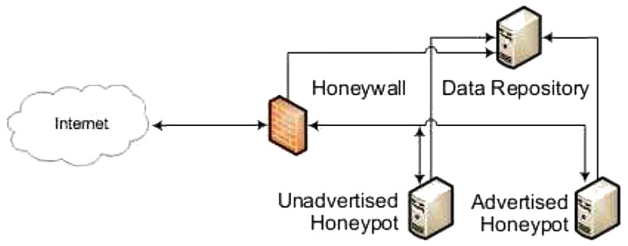

Fig. 1 – Design of our honeypot systems.

Table 1 – Unique IP sources, TCP connections and TCP packets, by dataset for both advertised and unadvertised honeypots.

	Web 2.0 I		Web 2.0 II		WebDBAdmin	
	Advertised	Unadvertised	Advertised	Unadvertised	Advertised	Unadvertised
Unique IP sources	1281	776	2676	790	1029	643
TCP connections	27,670	23,311	38,872	8116	12,361	2867
TCP packets	286,809	176,971	462,781	74,334	151,963	19,713

and thus have a more realistic setup. The second honeypot was not advertised anywhere on the Web. This *unadvertised* honeypot could only be reached by *IP-based strategy* when an attacker scans or attacks an IP address without (previous) involvement of search engines and crawlers. In our setup the unadvertised honeypot served as a control and allowed us to determine the relative contribution of search-based strategies (which only work on the advertised honeypot) to IP-based strategies (which work on both honeypts).

Both the advertised and unadvertised honeypots have their own IP addresses and hostnames, and ran on VMWare virtual machines. Since the honeypots could not be accessed directly by human users because of the ‘transparent linking’ approach used for advertising, the only non-malicious sessions in the logs consisted of system management traffic generated by our team and legitimate Web crawlers such as Google and MSNbot. Removing the system management traffic was a trivial task. The crawlers were removed based on the IP addresses listed in iplists.com and other similar sites and based on manual inspection of the remaining traffic.

To summarize, although our experimental set-up is based on using high-interaction honeypots that ran real operating systems and applications, it has three distinct characteristics:

- Instead of collection of unrelated services, our honeypots followed a three-tier architecture (consisting of a front-end Web server, application server, and a back-end database) and had a meaningful functionality.
- Our honeypots ran standard off-the-shelf operating system and applications which followed typical security guidelines and had only the vulnerabilities existing in the specific version, present in any other installation of that version running on Internet. Furthermore, the operating system and applications did not include user accounts with nil or weak passwords.
- Data collection was based on a sound experimental design, which for each configuration used a pair of two identical honeypots – one advertised and another unadvertised to serve as control. This set-up allowed us to easily distinguish between attackers activities that targeted specific services and those that reached the servers at random.

4. Characterization of malicious Web traffic

4.1. *Analysis of malicious traffic at the transport protocol layer*

In this subsection we present the analysis of the basic patterns in attackers behaviors that can be extracted from the headers of the IP packets. Table 1 shows the number of unique IP sources, and the TCP connections[2] and TCP packets for both advertised and unadvertised honeypots, for all three datasets. We first explored from which countries the attacks originated and whether attackers tended to revisit their targets. Then, we studied the malicious traffic patterns in terms of visited ports. In particular, we explored whether attackers tended to visit systematically multiple ports on a single machine or alternatively visited the same port on multiple machines. Because the TCP component dominated the malicious traffic, we also studied in details the distribution of the TCP traffic to different ports.

First, we focus on the country of origin of the source IP addresses that produced malicious traffic to our honeypots. Table 2 shows a summary of the percentage of unique IP sources and TCP connections per country, on each honeypot, in each dataset. The malicious UDP traffic, which was insignificant compared to the TCP traffic, is not included in Tables 1 and 2.

We note several interesting trends across different datasets. **The majority of unique attackers, across all datasets, to both advertised and unadvertised honeypots, came from the United States and China**. This observation is consistent with the observations made by some of the previous works which included the USA and China among the top countries from which malicious attacker behaviors originated (Pouget et al., 2005; Dacier et al., 2004; Bloomfield et al., 2008). The most significant portion of unique IP sources, however, did not always result in most significant contribution to the number of connections. For example, in Web 2.0 I dataset the United States had a significant portion of unique IP sources, but a very small portion of the total connections. This phenomenon was due to the fact that a single IP source address in Romania generated a significant number of connections, which skewed the distribution. Specifically, this attacker used an IP-based strategy to launch a massive password cracking attacks on the MySQL server on both the advertised and unadvertised Web 2.0 I honeypots.

The existence of heavy-hitters (i.e., single attackers who generated significant amount of malicious traffic) is another common pattern. For example, in addition to the previously mentioned attacker from Romania, in the case of Web 2.0 II dataset an attacker from Russia launched many connections to the HTTP server running on the advertised honeypot and another attacker from Mexico tried to break into the SSH server on both advertised and unadvertised honeypots.

[2] Since TCP is a connection oriented protocol, we define a *connection* as a unique tuple {source IP address, source port, destination IP address, destination port} with a maximum inter-arrival time between packets of 64 s following the definition used in network traffic analysis (Hohn et al., 2005).

Table 2 – Unique IP Sources (S) and TCP Connections (C) per country (as percentages), by dataset for both advertised and unadvertised honeypots.

Country	Web 2.0 I				Web 2.0 II				WebDBAdmin			
	Advertised		Unadvertised		Advertised		Unadvertised		Advertised		Unadvertised	
	S	C	S	C	S	C	S	C	S	C	S	C
USA	28.32	0.76	18.53	0.45	26.47	30.81	20.76	15.13	43.93	61.07	22.55	20.44
China	21.05	9.94	29.37	16.27	18.43	15.77	23.04	18.16	14.58	12.65	20.22	21.45
Romania	0.75	57.15	0.35	43.94	0.56	0.14	1.27	0.64	0.68	0.53	0.78	1.15
Taiwan	12.28	10.36	17.13	7.63	1.76	0.42	4.43	3.04	3.21	0.94	5.13	2.16
N. Korea	1.75	6.62	2.10	0.48	4.97	3.61	4.30	7.57	5.54	7.04	5.13	11.68
Mexico	0.25	1.33	0.35	1.87	0.49	8.11	1.39	34.27	0.58	0.11	0.78	0.31
Russia	1.75	0.02	2.10	0.02	4.07	20.62	4.05	1.07	1.94	4.42	3.11	2.30
Japan	0.39	0.08	2.10	18.74	2.13	0.48	0.89	0.22	0.78	0.39	2.02	0.84
Other	33.47	13.75	27.97	10.6	41.12	20.03	39.86	19.90	28.76	12.85	40.28	39.66
# Countries	47		40		94		77		66		66	

It is interesting to study whether attackers revisited systems or tended to visit once and did not come back. To quantify this information, we adopted the *attack source* metric, which is defined as a single source IP address connected to a honeypot on a single day (Berthier et al., 2008). Our analysis shows that **an overwhelming majority of unique source IP addresses visited each honeypot on only one day** (90%, 78%, and 80% for Web 2.0 I, Web 2.0 II, and WebDBAdmin dataset, respectively). These attackers were either unsuccessful in attacking the system or were searching for a specific vulnerability or service, did not find it, and never returned. The fact that attacker sources do not seem to revisit their targets was also observed by Dacier et al. (2004).

We also explored if attackers systematically visited multiple ports at a single honeypot. To characterize this type of attacker behavior, we use the term *vertical visit*, introduced by Yegneswaran et al. (2003), which is defined as a sequence of connections to multiple ports on the same IP address, performed by a single source IP address within a 1 h period of time. We observed vertical visits in all datasets, on both advertised and unadvertised honeypots. However, the amount of traffic attributed to vertical visits constituted less than 1% of the overall number of connections to the honeypots. Overall, across all datasets, we observed a total of 288 vertical visits, which utilized 72 unique port sequences. Although the total number of vertical visits was similar across all datasets, the targeted port sequences differed significantly. For example, *Windows services* were attacked more often in the Web 2.0 I dataset than in the others. This group includes sequences of UDP ports 1026 and 1027 (related to attacks on the Windows messenger, which was not running on our honeypots), as well as the sequence of UDP ports 137 and 1027. (UDP port 137 is used by the Windows NetBIOS Name Service.) Unlike the Windows related port sequences, the *Microsoft SQL Server (MSSQL)* was targeted many times in the Web 2.0 II and WebDBAdmin datasets, but only once in the Web 2.0 I dataset. All honeypots experienced vertical scans on various *HTTP related ports*, such as 80, 443, 8000, 8080, and 8888. All honeypots also experienced activity on *Worm related ports*, i.e., TCP ports 5554, 1023, 9898, which are known to be used as backdoors left open by well-known worms (Chen et al., 2005). In addition, we observed vertical scans to the high numbered *Ephemeral ports*, which likely were looking for peer-to-peer applications, and sequences of combinations of MSSQL and MYSQL ports, or SSH and HTTP ports.

In general, based on the similarity of the targeted services, port sequences, and frequencies in the Web 2.0 II and WebDBAdmin datasets it appears that **vertical visits were time specific and likely dependent upon current vulnerabilities.** Furthermore, since the number of attacks to the advertised and unadvertised honeypots were similar for most targeted services within the same dataset, we concluded that **vertical visits were conducted using an IP-based search strategy.**

Further, we explored another type of systematic attacker behaviors, so called *horizontal visit*, which is defined as a visit to the same port on several machines in a subnet, from the same source IP address, within one hour time period (Yegneswaran et al., 2003). Similarly as the vertical visits, horizontal visits constituted a very small portion of the traffic to each honeypot (generally less than 2%). Majority of the horizontal visits across all three datasets came to four ports: SSH (22), HTTP (80), MSSQL (1433) and SMTP (25). In addition to scanning/attacking other well known services (e.g., FTP, MYSQL), we also observed horizontal visits to the TCP port 10000, which is typically used by the Network Data Management Protocol (NDMP) and have been linked to malware known as OpwinTRojan, or the TCP port 12174 which has been connected with remote exploitation of design error vulnerability in Symantec System Center (CVE-2009-1431). Furthermore, we noticed that more than three-quarters of all horizontal scans were executed only once. The fact that horizontal visits reached the advertised and unadvertised honypots indicates that the attackers used an IP based strategy.

Both vertical and horizontal visits were introduced and explored by Yegneswaran et al. (2003). Although we used the same definitions, comparing the empirical values is not meaningful because they used very different source of data (i.e., network firewall logs).

As expected the traffic on all honeypots was dominated by the TCP component. Therefore, we next compare and contrast the distribution of the malicious TCP traffic to different destination ports across the three datasets. To allow for a fair comparison, because the duration of the data collection was different (i.e., 17 weeks for the Web 2.0 I dataset versus 22 weeks for the Web 2.0 II and WebDBAdmin datasets), we

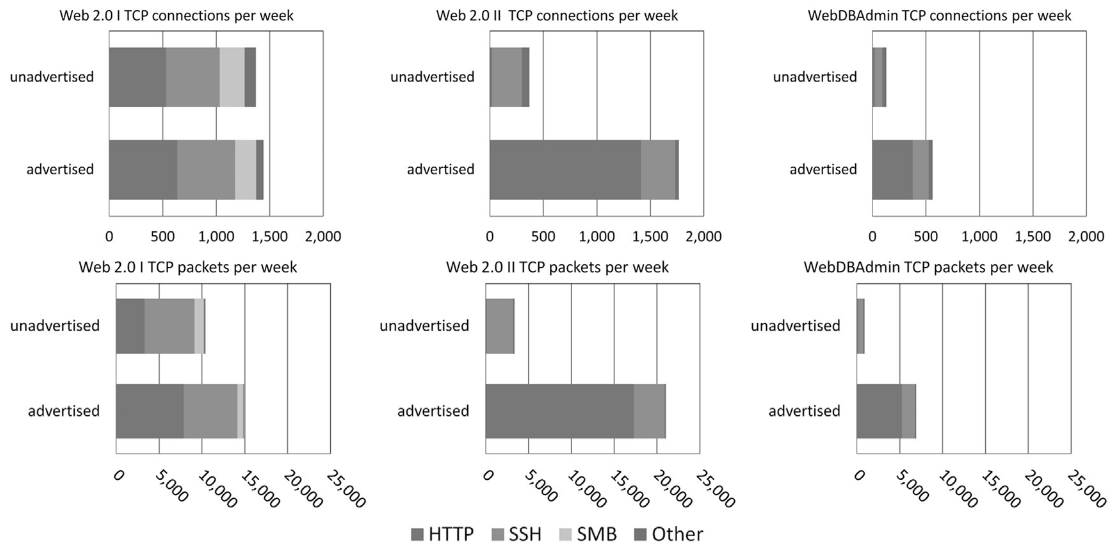

Fig. 2 – Distribution of the malicious TCP connections and packets per week, for all three datasets.

computed the number of connections/packets per week for each dataset. Fig. 2 shows, for each dataset, the breakdown of the TCP connections/packets per week across the three dominant components of TCP traffic (i.e., HTTP (port 80), SSH (port 22) and SMB (port 445)) and the traffic to the other TCP ports grouped in "Other". Based on the plots shown in Fig. 2 we made the following observations:

- Web 2.0 I dataset had close number of connections to port 80 to advertised and unadvertised honeypots. On the other side, advertised honeypots of Web 2.0 II and WebDBAdmin systems had significantly more HTTP traffic (both TCP connections and packets) than their unadvertised counterparts. The different behavior observed in the Web 2.0 I dataset was due to a single DoS attack, which resulted in a significant number of requests to both advertised and unadvertised honeypots. (Further details on this attack are given in Section 4.2.) When this heavy hitter was excluded, the Web 2.0 I dataset exhibited the same pattern as the other two datasets.
- **Both Web 2.0 I and Web 2.0 II had more malicious TCP traffic to port 80 than WebDBAdmin, which suggests that Web 2.0 applications, such as blog and wiki, are more attractive targets for attackers.**
- **In all datasets SSH (port 22) was the second most popular port on all servers, with significant amount of traffic (in both connections or packets) on the advertised and unadvertised honeypots**. Our further analysis showed that over 90% of the SSH packets on each honeypot were part of dictionary password cracking attacks, which indicates that using weak passwords may still be among the weakest links in systems security.
- Only Web 2.0 I dataset contained significant amount of TCP traffic to port 445, which is used by SMB (Server Message Block) protocol for file sharing in Windows operating systems and in the past has often been used for spreading worms and viruses.

Several related works based on data collected by honeypots reported that ports 80, 22 and 445, among others, were targeted by attackers (Dacier et al., 2004; Berthier et al., 2008; Bloomfield et al., 2008). However, comparing the actual percentages would not make sense because the honeypots in related work ran different set of applications, often unrelated to each other, and in most cases were not advertised.

The rest of the paper is focused on analysis of the malicious HTTP traffic, which showed the richest set of attackers activities, including those aimed at Web 2.0 applications.

4.2. *Descriptive statistical analysis of the malicious HTTP traffic*

In this subsection we analyze the vulnerability scan and attack classes observed across the HTTP traffic in the three datasets. This analysis was carried out based on the data extracted from the HTTP application level logs of the corresponding honeypots. As described in Section 3, after filtering out the legitimate crawlers and system management traffic our datasets contained only malicious sessions.

Labeling malicious traffic is not trivial and with the current state-of-the-art cannot be done automatically. Therefore, we used a semiautomated process based on identification of patterns in the HTTP application level logs, which is briefly described next. Since the textual format of the log data is not suited for flexible, customized analysis, we used our custom developed tool to parse the IIS logs and include the log entries in a relational database. In each dataset we first identified the unique HTTP requests. Classification of these unique requests was based on the specific patterns of attacker activities. For this we looked at different fields of the HTTP requests, such as the method used, values passed to the parameters, agent field, bytes transferred, error code, etc. For example, any request that posted spam to the wiki had to use `POST` method and to pass `submit` value to the `action` parameter (i.e., `action = submit`) in a request to the `/wiki/index.php` page.

Once the pattern that represents a specific activity was identified, we queried the database and labeled the corresponding requests. This process was repeated for each recognizable attacker activity. We also searched the publicly available vulnerability databases such as http://web.nvd.nist.gov or http://www.securityfocus.com/for specific signatures seen in the requests on our honeypots. More detailed description of the labeling process can be found in (Pantev, 2011).

Our initial classification of vulnerability scans and attacks was based on a large number of fine-grained classes. Consolidating the initial fine-grained classes into smaller number of coarse-grained classes shown in Tables 3–5 allows us to present the main patterns of attacker behaviors in a limited space. It should be noted that the process of identification of different vulnerability scan and attack classes was comprehensive, which is evident by the small number of unidentified attacks, which are only a part of the already small percentage of 'Other attacks' listed in Tables 3–5 (Some of the 'Other attacks' were identified, but included in this class due to the fact that they did not belong to any other class.)

We start the discussion with different types of vulnerability scans. *DFind* and *Other fingerprinting* attempts (e.g., HTTP request methods OPTIONS and CONNECT, Toata and Morfeus scanners, set of remote procedure calls XMLRPC.PHP) occurred in each dataset on both advertised and unadvertised honeypots, with approximately the same number of sessions, which indicates that these vulnerability scanning techniques used IP-based strategy to reach the honeypots.

All datasets have vulnerability scanning in *Static* + category, which in addition to visiting static pages, may include scans for applications that were not installed on our honeypots. For all three datasets advertised honeypots had significantly more sessions (as well as requests) in this class than unadvertised honeypots. This suggests that attackers predominately used search-based strategy for scanning the Static + category.

Vulnerability scans to *Blog* and *Wiki* (and their combination and combinations with *Static+*) appeared only at the advertised Web 2.0 I and Web 2.0 II honeypots. None of these vulnerability scans ended on the unadvertised Web 2.0 honeypots nor on the advertised and unadvertised WebDBAdmin honeypots. Obviously these vulnerability scans were exclusively done using a search-based strategy.

Conversely, the advertised honeypot from the WebDBAdmin dataset experienced the most significant number of the vulnerability scans to *phpMyAdmin* class (individually and in combination with *Static+*). Only several sessions in these classes appeared on the unadvertised WebDBAdmin honeypot, and on both Web 2.0 II honeypots. These results again indicate the dominant use of the search-based strategy.

Next, we present the observations related to the attack classes. It is obvious that there were much more attack sessions (and requests) on advertised honeypots than on unadvertised honeypots. Also, there was a difference in the amount of malicious traffic across the three datasets. The advertised Web 2.0 II dataset had almost 10 times more attack sessions than Web 2.0 I dataset. These two data sets were collected in consecutive time periods on the same honeypot. The significantly higher number of attack sessions may be due two reasons: increased activity on the Internet during the later time period and more likely wider propagation of the indexing

Table 3 – Distribution of the vulnerability scans and attacks on the Web 2.0 I honeypots. The classes with no traffic are left blank.

	Advertised Web 2.0 I honeypot				Unadvertised Web 2.0 I honeypot			
	Sessions		Requests		Sessions		Requests	
Vulnerability scans: Total	**824**	**73.77%**	**4349**	**44.07%**	**67**	**87.01%**	**1361**	**15.35%**
DFind	24	2.15%	25	0.25%	23	29.87%	24	0.27%
Other fingerprinting								
Static+	181	16.20%	1522	15.42%	41	53.25%	1243	14.02%
Blog	107	9.58%	253	2.56%				
Wiki	385	34.47%	923	9.35%				
Blog & Wiki	73	6.54%	406	4.11%				
Static+ & Blog	10	0.90%	72	0.73%				
Static+ & Wiki	19	1.70%	319	3.23%	2	2.60%	65	0.73%
Static+ & Blog & Wiki	25	2.24%	829	8.40%	1	1.30%	29	0.33%
phpMyAdmin								
Static+ & phpMyAdmin								
Attacks: Total	**293**	**26.23%**	**5519**	**55.93%**	**10**	**12.99%**	**7504**	**84.65%**
DoS	4	0.36%	3724	37.74%	9	11.69%	7490	84.49%
Password cracking phpMyAdmin								
Password cracking Blog	9	0.81%	127	1.29%				
Password cracking Wiki								
Spam on Blog	23	2.06%	82	0.83%				
Spam on Wiki	249	22.29%	1217	12.33%				
RFI	4	0.36%	13	0.13%				
SQL injection	2	0.18%	34	0.34%	1	1.30%	14	0.16%
XSS	2	0.18%	322	3.26%				
Other Attacks								
Total	**1117**	**100.00%**	**9868**	**100.00%**	**77**	**100.00%**	**8865**	**100.00%**

Table 4 – Distribution of the vulnerability scans and attacks on the Web 2.0 II honeypots. The classes with no traffic are left blank.

	Advertised Web 2.0 II honeypot				Unadvertised Web 2.0 II honeypot			
	Sessions		Requests		Sessions		Requests	
Vulnerability scans: Total	**2059**	**43.03%**	**4713**	**27.20%**	**38**	**73.08%**	**155**	**52.19%**
DFind	20	0.42%	20	0.12%	20	38.46%	20	6.73%
Other fingerprinting	2	0.04%	32	0.18%	2	3.85%	21	7.07%
Static+	327	6.83%	562	3.24%	7	13.46%	9	3.03%
Blog	690	14.42%	1024	5.91%	1	1.92%	13	4.38%
Wiki	922	19.27%	2224	12.83%				
Blog & Wiki	77	1.61%	594	3.43%				
Static+ & Blog	1	0.02%	4	0.02%				
Static+ & Wiki	3	0.06%	11	0.06%				
Static+ & Blog & Wiki	3	0.06%	80	0.46%				
phpMyAdmin	11	0.23%	150	0.87%	8	15.38%	92	30/98%
Static+ & phpMyAdmin	3	0.06%	12	0.07%				
Attacks: Total	**2726**	**56.97%**	**12,615**	**72.80%**	**14**	**26.92%**	**142**	**47.81%**
DoS								
Password cracking phpMyAdmin								
Password cracking Blog	1	0.02%	12	0.07%				
Password cracking Wiki	71	1.48%	181	1.04%				
Spam on Blog	1411	29.49%	3396	19.60%				
Spam on Wiki	1055	22.05%	5996	34.60%				
RFI	5	0.10%	7	0.04%	1	1.92%	2	0.67%
SQL injection								
XSS	11	0.23%	149	0.86%				
Other Attacks	172	3.59%	2874	16.59%	13	25.00%	140	47.14%
Total	**4785**	**100.00%**	**17,328**	**100.00%**	**52**	**100.00%**	**297**	**100.00%**

Table 5 – Distribution of the vulnerability scans and attacks on the WebDBAdmin honeypots. The classes with no traffic are left blank.

	Advertised WebDBAdmin honeypot				Unadvertised WebDBAdmin honeypot			
	Sessions		Requests		Sessions		Requests	
Vulnerability scans: Total	**513**	**93.44%**	**1249**	**76.44%**	**34**	**56.67%**	**113**	**41.54%**
DFind	19	3.46%	19	1.16%	20	33.33%	20	7.35%
Other fingerprinting	3	0.55%	36	2.20%	2	3.33%	37	13.60%
Static+	306	55.74%	503	30.78%	6	10.00%	7	2.54%
Blog								
Wiki								
Blog & Wiki								
Static+ & Blog								
Static+ & Wiki								
Static+ & Blog & Wiki								
phpMyAdmin	155	28.23%	372	22.77%	4	6.67%	30	11.03%
Static+ & phpMyAdmin	30	5.46%	319	19.52%	2	3.33%	19	6.99%
Attacks: Total	**36**	**6.56%**	**385**	**23.56%**	**26**	**43.33%**	**159**	**58.46%**
DoS								
Password cracking phpMyAdmin	1	0.18%	50	3.06%				
Password cracking Blog								
Password cracking Wiki								
Spam on Blog								
Spam on Wiki								
RFI	1	0.18%	1	0.06%				
SQL injection								
XSS								
Other Attacks	34	6.19%	334	20.44%	26	43.33%	159	58.46%
Total	**549**	**100.00%**	**1634**	**100.00%**	**60**	**100.00%**	**272**	**100.00%**

that led to increased number of search-based attacks. The Web 2.0 II and WebDBAdmin datasets were collected during the same time period on two honeypots with different configurations. Based on the significantly higher volume of attack traffic in Web 2.0 II dataset it appears that attackers had more interest in Web 2.0 applications than in database administration server. It should be noted that our intent was not to establish general trends of attack traffic, but rather to explore if the amount of malicious traffic and its nature (i.e., distribution across different attacks classes) differ for servers with different configurations and during different time periods. Details of the distribution of attack traffic across different classes are as follows.

The only *Denial of Service (DoS) attack* was observed on Web 2.0 I honeypots. It was based on the Microsoft IIS WebDAV PROPFIND and SEARCH Method Denial of Service Vulnerability. The attack was unsuccessful because this vulnerability was fixed in the version of the operating system which ran on our honeypots, thus resulting in requests with 404 errors. The attacker used IP-based strategy to reach both the advertised and unadvertised honeypots. Typical for DoS attacks, although the number of sessions was small, they contributed to a significant number of malicious requests on both servers. In particular there were four DoS sessions on the advertised and nine on the unadvertised Web 2.0 I honeypot. These sessions contributed to 34.74% of the requests on advertised Web 2.0 I honeypot and 84.49% of the requests on the unadvertised Web 2.0 I honypot. At this point we revisit the results presented in Fig. 2 and the observation that the amount of TCP traffic to port 80 in the Web 2.0 I dataset was close on both advertised and unadvertised honeypots (unlike the other two datasets – Web 2.0 II and WebDBAdmin – which experienced significantly more malicious TCP traffic to port 80 on advertised honeypot than on the unadvertised counterpart). The main reason behind this observation was the DoS attack, which used IP-based strategy and thus reached both the advertised and unadvertised Web 2.0 I honeypots. None of the other datasets had DoS or other type of IP-based attack with significant amount of traffic to port 80.

Majority of attack sessions on Web 2.0 honeypots were related to posting *Spam on Wiki* and *Spam on Blog*. Spam obviously is becoming a major problem for many servers that host Web 2.0 applications due to their interactive nature. *Spam on Wiki* was in a form of posting a new topic or editing existing topics with random text or advertisements, each including link(s) to other Web site(s). *Spam on Blog* was in a form of comments to already posted discussion topics. The comments contained random text or advertisements and often link(s) to other Web site(s). The fact that the spam on the Blog was predominately posted as comments instead as new blog entries shows that spammers used the quickest and easiest way to spread spam, even if that would mean less visibility on the site. (Unlike comments, the more visible blog entries would require creating an account and logging in.) We tested the links included in the Wiki and Blog spam with the Google's Safe Browsing diagnostic page and by manual inspection. It appeared that none of the links contained malicious content. Instead, it seems the attackers' goals were either to sell products (e.g., pharmaceutics) or to generate traffic to their Web sites in order to increase the advertising profits (e.g., spamdexing). We also explored the origin of the spam attacks. On the Web 2.0 I honeypot over 63% of unique IPs that posted spam were from China, followed by close to 9% of unique IPs located in the USA. The distribution of unique IPs was more even in case of Web 2.0 II honeypot – close to 19% of the spam attacks originated from the USA and around 14% from China. The fact that no attempts to post spam ended on the unadvertised honeypots clearly indicates the use of search-based strategies.

The advertised honeypots of all three datasets experienced some small percentage of *Password cracking attacks* to user accounts existing on different applications. However, the numbers of password cracking attempts to the user accounts of Wiki, Blog, and phpMyAdmin applications were insignificant when compared to the password cracking attempts to the SSH user accounts. This indicates that attackers who try to penetrate a system by breaking weak passwords go for the easiest, most obvious target – the SSH user accounts.

The small number of observed *Remote File Inclusion (RFI)*, *SQL injection*, and *Cross-site Scripting (XSS)* attacks mostly ended up on the advertised honeypots. Some of these attacks were based on known vulnerabilities posted at the National Vulnerability Database (NVD, 2013). Examples include RFI attacks based on CVE-2007-4009, CVE-2006-5402, CVE-2008-2836, CVE-2007-6488, and CVE-2008-3183, SQL injection attacks based on CVE-2008-6923 and CVE-2007-2821, and XSS based on CVE-2007-0308. Some of the attacks classified as *Other attacks* were also based on known vulnerabilities (e.g., CVE-2006-6374 and CVE-2008-3906).

Closer look at Tables 3–5 indicates that attackers predominately used the search-based strategy to reach the Web servers because the advertised honeypots received significantly more traffic than the unadvertised honeypots. Furthermore, Web 2.0 I and Web 2.0 II datasets, collected by honeypots with a same configuration, had similar types of vulnerability scans and attacks. The amounts of traffic in these categories, however, were different. The types of vulnerably scans and attacks observed on the WebDBAdmin honeypot, on the other side, were very different from those observed in the two Web 2.0 datasets. The fact that attackers tailored their attacks to the specific software applications running on the target honeypots once again illustrates the use of search-based strategy.

4.3. *Inferential statistical analysis of the malicious HTTP traffic*

The analysis of the non-malicious Internet traffic has a long tradition and led to models that formally describe different aspects of the traffic (see (Leland et al., 1994; Paxson and Floyd, 1995; Hohn et al., 2005; Goseva-Popstojanova et al., 2006b, a) and references therein). Motivated by the long tradition of non-malicious traffic modeling and analysis, in this section we present inferential statistical analysis of the characteristics of malicious cyber activities. These results are based on the analysis of the datasets collected by the advertised honeypots because they experienced significantly more malicious HTTP traffic than their unadvertised counterparts. In particular, we explore questions such as: "What models can be used to describe different attributes of malicious sessions?", "Are

Table 6 – Distributions fitted to the tail of Session duration.

	Web 2.0 I	Web 2.0 II	WebDBAdmin
Min/median/max	0/2/4330 s (1.20 h)	0/4/53,383 s (14.83 h)	0/0/6985 s (1.94 h)
Distribution	Pareto	Pareto	Pareto
Parameters	α=0.6	α=1.1	α=0.5
	b=14	b=1376	b=40
# of points in the tail	300	200	100

Table 7 – Distributions fitted to the tail of Number of requests per session.

	Web 2.0 I	Web 2.0 II	WebDBAdmin
Min/median/max	1/2/1021 requests	1/2/285 requests	1/2/50 requests
Distribution	Pareto	Pareto	Lognormal
Parameters	α=0.8	α=1.6	σ=0.512
	b=6	b=29	μ=2.556
# of points in the tail	100	50	50

the differences of malicious sessions characteristics across different datasets statistically significant?", and "What distributions can be used to describe the number of malicious sessions/requests generated by unique attackers?"

4.3.1. *Analysis of malicious web session attributes*

In this section we study three attributes of malicious sessions: session duration (in time units), number of requests per session, and bytes transferred per session. Motivated by the results presented so far we explored whether, similarly to the non-malicious traffic, heavy-tailed distributions can be used to describe the attributes of malicious Web sessions. It should be noted that the fact that the Web traffic has been shown to exhibit heavy-tailed behavior (Goseva-Popstojanova et al., 2006a) does not necessary mean that the malicious traffic would follow the same behavior. This aspect of the malicious Web sessions, to the best of our knowledge, has not been explored in the related work.

In practical terms, a random variable[3] that follows a heavy-tailed distribution can give rise to extremely large values with non-negligible probability, even though most of the values may be small. For example, most of the sessions on our honeypots had small number of requests, however there were a few session with many requests.

The simplest heavy-tailed distribution is the Pareto distribution with a shape parameter (i.e., tail index) $\alpha>0$ and scale parameter $b>0$. If $1<\alpha\leq2$, the distribution has a finite mean and infinite variance; if $0<\alpha\leq1$, the distribution has infinite mean and variance. To estimate the tail index α of a Pareto distribution we used the log–log complementary distribution (LLCD) plots and Hills estimator (Goseva-Popstojanova et al., 2006a). For each session attribute we first fitted the Pareto distribution and then tested the goodness-of-fit using the Anderson–Darling test (Anderson and Darling, 1954). In the cases when Pareto distribution was not a good fit, we fitted and tested the goodness-of-fit for the lognormal distribution. (More details on these distributions are given in Appendix A.)

As shown in Table 6 **the tails of the *Session duration* for all three datasets were modeled well with Pareto distribution** (at significance level 0.05). In other words, for each dataset there were a few sessions that were very long compared to the most of the other sessions. The longest sessions in Web 2.0 I, Web 2.0 II, and WebDBAdmin datasets were 1.20 h, 14.83 h and 1.94 h, respectively. The attackers activities in these long sessions were labeled as *Static+ & Wiki*, *Spam on Wiki*, and *Static+*, respectively.

[3] In our case each session attribute (e.g., session duration, number of requests, and bytes transferred in a session) is a random variable.

Table 7 summarizes the models fitted into the tail of the *Number of requests* per *session* for all three datasets. **The tails of the *Number of requests per session* for both Web 2.0 I and Web 2.0 II datasets followed Pareto distributions.** Web 2.0 I dataset had heavier tail (i.e., more requests per session) than the Web 2.0 II dataset, which is obvious from the tail index values (α=0.8 compared to α=1.6). So, what did the attackers do in sessions with the largest number of requests? The session with 1021 requests in the Web 2.0 I dataset was due to *DoS* on Microsoft IIS, while the session with 285 requests in the Web 2.0 II dataset was labeled as *Spam on Wiki*. **In the case of WebDBAdmin dataset, the hypothesis that the tail of the *Number of requests per session* follows a Pareto distribution was rejected. Instead, a lognormal distribution with parameters given in** Table 7 **was a good fit.** The longest session in WebDBAdmin dataset had only 50 requests and it was labeled *Password cracking on phpMyAdmin*.

As shown in Table 8, ***Bytes transferred per session* for all three datasets followed Pareto distribution.** It is interesting to see what were the attackers activities in the sessions that had maximum bytes transferred for each dataset. The session with 35 MB transferred in Web 2.0 I dataset was due to *Static +* content, while the session with 55 MB transferred in Web 2.0 II dataset was labeled as *Blog and Wiki*. The session with 55 MB transferred in WebDBAdmin dataset was labeled as *Static+ and PhpMyAdmin*.

Next, we explored if each malicious session attribute – Session duration, Number of requests per session, and Bytes transferred per session – experienced statistically significant differences across the three datasets. Due to the fact that session attributes did not follow normal distribution, we could not use the parametrical ANOVA F test to test the central tendency across all three datasets. Instead, we used the Kruskal–Wallis test (Siegel and Castellan, 1988), which is a nonparametric test. Here we only present the outcomes of the statistical tests. The formal null hypotheses and detailed results of the statistical tests are given in Appendix B.

Table 8 – Distributions fitted to the tail of Bytes transferred per session.

	Web 2.0 I	Web 2.0 II	WebDBAdmin
Min/median/max	0B/9KB/35 MB	0B/20KB/55 MB	0B/2.3KB/55 MB
Distribution	Pareto	Pareto	Pareto
Parameters	α=0.6	α=1.2	α=0.5
	b=36,035	b=162,272	b=18,747
# of points in the tail	100	80	250

For each of the session attributes – Session duration, Number of requests, and Bytes transferred – the Kruskal–Wallis test produced p-values which were very close to zero, that is, less then the significance level 0.05. This means that we can reject each of the null hypotheses that there is no difference in session duration/number of requests/bytes transferred across datasets in favor of the alternative hypotheses that, for each attribute, at least one of the datasets yielded larger observations than at least one of the other datasets. To find where were the differences we used the method for multiple comparisons (Siegel and Castellan, 1988) implemented in R (R Development Core Team, 2008).

For each session attribute (i.e., Session duration/Number of requests/Bytes transferred) and each combination of dataset pairs, the null hypothesis that "there is no difference between values" of that specific attribute was rejected. These results show that malicious Web sessions had different duration/number of requests/bytes transferred on the same Web systems during different periods of time (i.e., Web 2.0 I and Web 2.0 II datasets), as well as on different Web systems during the same period of time (i.e., Web 2.0 II and WebDBAdmin datasets), with statistical significance. In general, it appears that the attribute values of malicious Web sessions depended both on the server configurations and the time period of data collection.

4.3.2. *Malicious traffic generated by unique IP sources*

Another interesting question to ask is whether the malicious sessions (and requests) observed on our honeypots were generated uniformly by different attackers or there were heavy-hitters that produce much more sessions (or request) than others. The discussion presented in Section 4.1 and the inspection of the histograms of the number of malicious sessions/requests generated by unique source IP addresses indicated that the latter is true for all datasets. To explore this question formally, we first fitted the Pareto distribution in the data for the sessions and requests generated by unique IP source addresses, and then tested the goodness-of-fit using the Anderson–Darling test (Anderson and Darling, 1954). In the cases when Pareto was not a good fit, we fitted the lognormal distribution and again tested the goodness-of-fit.

Table 9 summarizes the distributions of the sessions and requests per unique source IP for all datasets. In all cases either Pareto or lognormal distribution were a good fit, which is a formal confirmation of the heavy hitters phenomenon discussed in Section 4.1. Typical for a heavy-tailed and skewed distributions, a small number of attackers tended to generate many requests. For example, 38% of the requests in the Web 2.0 I dataset were from one IP address (i.e., one attacker) who launched a DoS attack on the Microsoft IIS Server. This result is consistent with the result presented in the only related work that studied the statistical characteristics of the "worst offender" IPs (Yegneswaran et al., 2003), which showed that very few sources were responsible for generating a significant fraction of all non-worm scans that were observed in the collected firewall logs.

5. Classification of malicious sessions

Being able to distinguish automatically between vulnerability scans and attacks is important because attacks are much more critical events than vulnerability scans. The work presented in this section is based on the hypotheses that different malicious activities exhibit different behavioral patterns, which provides bases for using machine learning methods for their classification. With the increase in the number and diversity of malicious behaviors on Internet, automatic classification holds enormous potential for improving the protection and resiliency of services and systems. It can be used, for example, to support the generation of attack signatures or to develop attack patterns for testing system resilience to attacks.

In this section we first describe the features (i.e., session attributes) extracted for each malicious Web sessions and our data mining approach, and then present the results of using machine learning techniques to automatically classify malicious Web sessions to two classes: vulnerability scan class and attack class. As in Section 4.3, the results presented in this section are based on the datasets collected by the advertised honeypots.

5.1. *Description of the extracted features*

Each malicious Web session consists of one or more HTTP requests. We characterize each session with 43 different features (i.e., session attributes), which are listed and briefly described in Table 10. These 43 features extend the feature sets used by Cukier et al. (2006) and by Perdisci et al. (2010) (consisting of four and seven features, respectively) by considering features similar to those used in articles on network and Web server intrusion detection, and Web crawlers identification. These articles are referenced in Table 10.

As it can be noticed from Table 10 some of the features are scalars that describe a specific session characteristic (e.g., Number of requests in a session, Session duration in seconds, binary indication of an ASCII control character). For attributes that describe individual requests within a session (e.g., Number of parameters passed in a request substring) we created a vector and then computed the corresponding aggregate metrics: Mean, Median, Minimum, Maximum, and

Table 9 – Distributions of the number of requests and sessions generated from unique source IPs.

	Web 2.0 I		Web 2.0 II		WebDBAdmin	
	Sessions	Requests	Sessions	Requests	Sessions	Requests
Min/median/max	1/1/58	1/3/3724	1/1/64	1/2/151	1/1/312	1/2/2541
Distribution	Lognormal	Pareto	Lognormal	Lognormal	Pareto	Pareto
Parameters	σ=0.70538	$\alpha = 1.3$	σ=0.718	σ=0.594	$\alpha = 0.8$	$\alpha = 1.2$
	μ=2.0034	b=26	μ=1.7372	μ=2.864	b=7	b=22

Table 10 – The list of 43 features (i.e., session attributes) which were extracted from the Web Server's log files.

Feature id	Feature(s) name(s)	Feature(s) description
Metrics used in our previous work (Goseva-Popstojanova et al., 2010b)		
1	Number of requests	Count of the total number of requests within a single HTTP session.
2	Bytes transferred	The amount of traffic, measured in bytes, transferred in a single HTTP session.
3	Duration	The duration of an HTTP session, measured in seconds.
Metrics regarding the time between requests in a session (Lourenco and Belo, 2006)		
4–8	Mean, Median, Minimum, Maximum and Standard Deviation of the Time between requests	Time, measured in seconds, between successive requests in a single HTTP session. Since a single HTTP session can have multiple requests, these are aggregate metrics which represent the vector of times between successive requests in a session.
Metrics regarding the number of requests with a particular method type (Lourenco and Belo, 2006)		
9–14	GET, POST, OPTIONS, HEAD, PROPFIND, and OTHER	Number of requests in a single HTTP session with a particular method type (i.e., GET, POST, OPTIONS, HEAD, PROPFIND, and OTHER). OTHER is the number of requests that used one of the other HTTP method types: PUT, DELETE, TRACE, or CONNECT.
Metrics regarding the number of requests towards different types of files (Lourenco and Belo, 2006)		
15–19	Picture, Video, Static app. files, Dynamic app. files, Text	Number of requests within a session that were towards Picture (.jpeg, .jpg, .gif), Video (.avi, .mpg, .wmv), Static HTML (.html, .htm), Dynamic Application (.asp, .php), or Text files (.txt, .ini, .css).
Metrics regarding response status codes (introduced in this paper)		
20–24	Informational (1xx), Success (2xx), Redirect (3xx), Client error (4xx), Server Error (5xx)	Number of requests within a session that belong to each group of status codes (1xx–5xx).
Metrics regarding the length of request substrings within a session (Kruegel et al., 2005; Almgren et al., 2000)		
25–29	Mean, Median, Minimum, Maximum, and Standard Deviation of the Length of the request substring	Length, in number of characters, of the request substring which tells what was requested. The HTTP method and HTTP protocol version are not considered as parts of the request substring. These features are aggregate values of the length of all requests substrings within a session.
Metrics regarding the number of parameters passed in request substrings within a session (Almgren et al., 2000)		
30–34	Mean, Median, Minimum, Maximum, and Standard Deviation of the Number of parameters	Number of the HTTP request parameters passed to a form with each request. These features aggregate values of all parameters passed in a session.
Binary metrics (Estevez-Tapiador et al., 2005; Kruegel et al., 2005; Garcia et al., 2006; Lourenco and Belo, 2006)		
35	robots.txt	Indicates if robots.txt file was accessed in at least one of the requests in a session.
36	Night	Indicates if the session occurred between 12 am and 8 am (local time).
37	Remote sites injected	Indicates if there is a remote site injection in at least one of the requests in a session.
38	Semicolon used	Indicates if a ";" was used to divide multiple attributes passed to an application in at least one of the requests in a session.
Binary metrics (Almgren et al., 2000)		
39	Unsafe characters	Indicates if a character was encoded with suspicious encoding.
40	Reserved characters	Indicates if reserved characters like $, +, @, etc. were used.
41	ASCII control char	Indicates if ISO-8859-1 characters in ranges 00–1F and 7F were used.
42	Non-ASCII control char	Indicated if ISO-8859-1 characters in range 80-FF were used.
43	Invalid characters	Indicates if an invalid encoding was used.

Standard Deviation. Since the ranges of the 43 features differed significantly we first applied Min Max Normalization, resulting in a new range [0,1] for each feature.

5.2. *Description of our data mining approach*

To classify the observed malicious traffic to attacks and vulnerability scans we used three different supervised machine learning methods: Support Vector Machines (SVM) (Boser et al., 1992), decision tree J48 (which is a Java implementation of the C4.5 decision tree algorithm (Quinlan, 1993)), and a rule induction method based on partial decision trees (PART) (Frank and Witten, 1998). For the SVM we used the Radial Basis Function (RBF) as a kernel function. For the tree based methods, J48 and PART, we used the Reduced Error Pruning (REP) (Quinlan, 1987) to build the pruned trees, which have reduced size or number of nodes and thus help avoiding unnecessary complexity and over-fitting of the data.

To evaluate learners' performance, we first computed the confusion matrix

	Actual: Vulnerability scan	Actual: Attack
Predicted: Vulnerability Scan	*TN*	*FN*
Predicted: Attack	*FP*	*TP*

where *TN*, *FN*, *FP*, and *TP* denote true negatives, false negatives, false positives, and true positives, respectively. Then, we computed the following performance metrics which assess different aspects of the classification:

$$\text{accuracy (acc)} = (TN + TP)/(TN + FN + FP + TP) \quad (1)$$

$$\text{probability of detection}(pd) = TP/(FN + TP) \quad (2)$$

$$\text{probability of false alarms}(pf) = FP/(TN + FP) \quad (3)$$

$$\text{precision } (prec) = TP/(TP + FP) \quad (4)$$

$$\text{balance } (bal) = 1 - \sqrt{(0 - pf)^2 + (1 - pd)^2}/\sqrt{2} \quad (5)$$

The *accuracy*, given with (1), provides the percentage of sessions that are detected correctly. Since attacks are more critical events than vulnerability scans *probability of detection* defined by (2), which sometimes is also called *recall*, accounts for the probability of detecting an attack (i.e., the ratio of detected attacks to all attacks). *Probability of false alarm*, defined by (3), is the ratio of vulnerability scans misclassified as attacks to all vulnerability scans, that is, the probability of falsely reporting a vulnerability scan as an attack. (The reader is reminded that our datasets contain only malicious traffic.) *Precision*, defined by (4), determines the fraction of sessions correctly classified as attacks out of all sessions classified as attacks. Ideally, we want probability of detection to be 1 and probability of false alarm to be 0. It appears that a rather useful metric of performance is so called *balance*, which is defined as the Euclidian distance from this ideal point of $pf=0,pd=1$ to a pair of (pf,pd). For convenience, the balance is normalized by the maximum possible distance across the ROC square $\sqrt{2}$ and then subtracted from 1 (see (5)). It follows that higher balance is better since (pf,pd) point falls closer to the ideal point (0,1).

Besides applying the learners to all 43 features we also employed a feature selection method. The motivation for using feature selection was to explore whether a small subset of session characteristics can be used to efficiently separate attack sessions from vulnerability scan sessions. In addition to learning about characteristics of malicious activities, reducing the number of features by removing the irrelevant and noisy features speeds up the machine learning algorithms and hopefully improves their performance (Liu and Yu, 2005). In this paper we used information gain feature selection method which ranks the features from the most informative to least informative using the information gain as a measure (Liu and Yu, 2005). This is a filter selection method because it uses the characteristics of the data to evaluate the features (i.e., does not use any learning algorithm).

Since our goal was to identify the smallest number of features sufficient to accurately distinguish attacks from vulnerability scans, we used the following procedure for each individual dataset. For each learner, we started with the highest ranked feature and included one feature at a time until reaching less than or equal to 1% difference of the probability of detection (pd) compared to the case when all 43 features were used. (Obviously, different learners required different number of features to reach the 1% threshold.) Then, we chose the smallest set of features from the best learner among all learners for the specific dataset and used it to test the performance of all learners for that dataset.

The machine learning for all learners was done using 10-fold cross-validation, which involves partitioning the dataset into ten complementary subsets using stratified random sampling, and then using nine subsets as training data and validating the results on the remaining subset (called validation data or testing data). This process was repeated ten times, with each of the ten subsets used exactly once as validation data. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation. The results presented in Section 5.3 are the averages of the results from the ten folds.

5.3. *Results of automatic classification of malicious traffic*

5.3.1. *Classification results using all 43 features*

In this section we explore RQ4, i.e., if supervised machine learning methods can be used to automatically distinguish between Web attacks and vulnerability scans using all 43 features. Table 11 presents the performance metrics of SVM, J48 and PART and their pruned versions, for all three datasets. The method that has the best (worst) probability of detection is highlighted in *italic* font shown in a box (**bold**). As it can be seen from Table 11, in case of Web 2.0 I and Web 2.0 II datasets the learners had very high probability of detection (in the ranges of 96.92%–98.63% and 92.52%–97.36%, respectively). WebDBAdmin dataset had significantly lower probability of detection (in the range of 41.67%–75.00%) when compared to the other two datasets. However, the lower probability of detection in the case of WebDBAdmin dataset was not unexpected, having in mind that this dataset had the smallest percentage of attacks (only 6.56%). Even more, 34 out of total 36 observed attacks were classified as 'Other attacks' (see Table 5) because they differed among themselves. In addition, the remaining two attacks also belonged to two different classes (i.e., Password cracking phpMyAdmin and RFI). Therefore, whenever any of these attack sessions appeared in the validation (i.e., testing) data it did not have corresponding instances in the training data. In this context, we can say that the best performing learner J48 was able to detect 75% of the new, previously unseen attacks.[4]

[4] It should be noted that the probability of detection for the fourth dataset used in our previous work (Goseva-Popstojanova et al., 2012) was in the range of 82.76%–96.55% even though it had only 29 attacks (i.e., 13.55% of the total number of malicious sessions). The better results were due to the fact that only 5 out of the 29 attacks were classified as 'Other', that is, there were attack patterns. As mentioned earlier, that dataset was not included in this paper because it was collected by a honeypot which ran different operating system and Web server, and hosted different static content.

Table 11 – Summary of classification results using all 43 features, for all three data sets. The method that gives the best (worst) probability of detection is highlighted in $\boxed{\textit{italic}}$ (bold).

Dataset	Features used	Learner	Accuracy	Prob. Of detection	Prob. Of false alarm	Precision	Balance
Web 2.0 I	All 43 features	$\boxed{\text{SVM}}$	99.37%	$\boxed{\textit{98.63\%}}$	0.36%	98.97%	98.79%
		$\boxed{\text{J48}}$	99.55%	$\boxed{\textit{98.63\%}}$	0.12%	99.66%	99.00%
		J48 pruned	99.19%	97.61%	0.24%	99.31%	98.24%
		PART	98.93%	96.93%	0.36%	98.95%	97.71%
		PART pruned	98.93%	**96.92%**	0.36%	98.95%	97.81%
Web 2.0 II	All 43 features	SVM	94.19%	**92.52%**	3.59%	97.15%	94.34%
		J48	94.19%	**92.52%**	3.59%	97.15%	94.34%
		J48 pruned	96.80%	96.88%	3.30%	97.49%	97.17%
		$\boxed{\text{PART}}$	96.91%	$\boxed{\textit{97.36\%}}$	3.69%	97.22%	97.29%
		PART pruned	96.90%	97.35%	3.69%	97.22%	96.79%
WebDBAdmin	All 43 features	SVM	95.81%	**41.67%**	0.39%	88.24%	57.92%
		$\boxed{\text{J48}}$	96.72%	$\boxed{\textit{75.00\%}}$	1.75%	75.00%	75.00%
		J48 pruned	97.09%	72.22%	1.17%	81.25%	76.30%
		PART	96.54%	63.89%	1.17%	79.31%	70.57%
		PART pruned	96.53%	63.88%	1.16%	79.31%	74.45%

Interestingly, the probability of false alarm was low in all three datasets, in many cases less than 1% and never above 3.69%. Another important observation made based on the data presented in Table 11 is that accuracy on its own can be a misleading measure of learner's performance. Specifically, when all 43 features were used the accuracy was very high (i.e., above 94%) for all datasets and all learners, even in cases when the probability of detection was low to moderate (i.e., at most 75%).

Overall, as indicated by moderately to very high probability of detection and low probability of false alarm, we can conclude that supervised machine learning methods can be used to successfully distinguish between attack and vulnerability scan sessions.

5.3.2. *Feature selection and comparison of learners' performance*

In this section we discuss the results of the feature selection process and compare the learners, that is, address the two parts of RQ5: Do attacks and vulnerability scans differ in a small number of features? Do some learners perform consistently better than other across multiple datasets?

When distinguishing between attacks and vulnerability scans it is very important to choose the simplest possible model because it leads to better efficiency and performance of the machine learning algorithms. The results of using information gain as a feature selection method and the procedure described in Section 5.2 showed that attacks differed from

Table 12 – Summary of classification results with selected features, for all three data sets. The method that gives the best (worst) probability of detection is highlighted in $\boxed{\textit{italic}}$ (bold).

Dataset	Features used	Learner	Accuracy	Prob. of detection	Prob. of false alarm	Precision	Balance
Web 2.0 I	10, 28, 26, 25	SVM	98.75%	**95.90%**	0.24%	99.29%	97.10%
		$\boxed{\text{J48}}$	99.19%	$\boxed{\textit{97.61\%}}$	0.24%	99.30%	98.30%
		J48 pruned	98.83%	96.24%	0.24%	99.30%	97.34%
		$\boxed{\text{PART}}$	99.19%	$\boxed{\textit{97.61\%}}$	0.24%	99.30%	98.30%
		PART pruned	98.83%	96.24%	0.24%	99.30%	97.34%
Web 2.0 II	28, 25, 10, 26, 29, 2	SVM	91.58%	**85.66%**	5.80%	99.49%	89.85%
		J48	95.71%	95.26%	3.69%	97.16%	95.76%
		$\boxed{\text{J48 pruned}}$	95.63%	$\boxed{\textit{95.56\%}}$	4.27%	96.73%	95.64%
		PART	94.96%	94.79%	4.81%	96.31%	94.99%
		PART pruned	94.65%	94.83%	5.59%	95.74%	94.62%
WebDBAdmin	24, 28, 26, 17, 25, 2	SVM	94.54%	**25.00%**	0.58%	75.00%	46.97%
		$\boxed{\text{J48}}$	97.09%	$\boxed{\textit{75.00\%}}$	1.36%	79.41%	82.30%
		$\boxed{\text{J48 pruned}}$	97.27%	$\boxed{\textit{75.00\%}}$	1.17%	81.82%	82.23%
		PART	96.90%	69.44%	1.17%	80.65%	78.38%
		PART pruned	95.81%	69.44%	2.34%	67.57%	78.33%

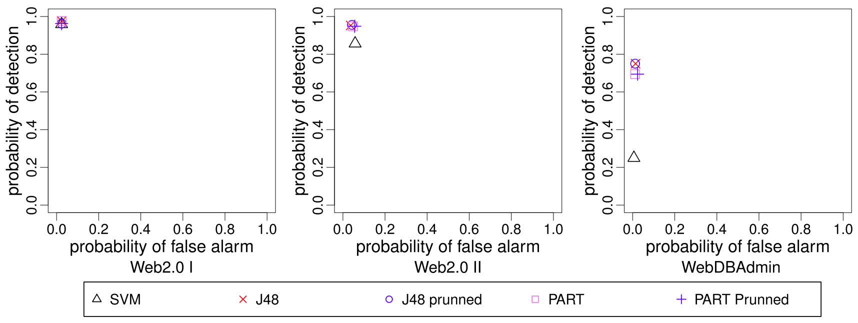

Fig. 3 – ROC squares for learners applied on the selected features, for each data set.

vulnerability scans only in a small number of features (i.e., session characteristics). Specifically, **depending on the dataset, from four to six features can be used to classify malicious activities without significantly worsening learners' performance compared to when all 43 features were used.** (In Table 12 the selected features are ordered from the most to the least informative.)

Next, we compare the performance of the machine learning methods used in this paper: SVM, J48, and PART. Fig. 3 shows the ROC squares for all three dataset. In each of these ROC plots, the *x*-axis shows the probability of false alarm and the *y*-axis shows the probability of detection. The data point (0,1) in the upper left corner corresponds to the optimal performance, i.e., high probability of detection with a low probability of false alarm.

The results presented in Tables 11 and 12, and Fig. 3 show that the **tree based learners J48 and PART, both when used with all features and with selected features, outperformed the SVM**. Only in one out of six cases (for Web 2.0 I dataset with all 43 features) SVM had the best probability of detection value which was equal to J48. However, even in that case J48 had slightly lower probability of false alarm, and thus slightly better balance, than SVM. SVM did significantly worse than J48 and PART in case of WebDBAdmin dataset, which had small number of attacks, almost all different from each other. Furthermore, SVM required more features than the tree-based methods to achieve the desired less than 1% difference when selected features were used compared to using all 43 features. Specifically, instead of four, six, and six features for datasets Web 2.0 I, Web 2.0 II, and WebDBAdmin shown in Table 12 which were all selected based on one of the tree-based methods, SVM required twelve, sixteen, and ten features, respectively. Combining these observations with the fact that SVM had the longest execution time of the three learners, clearly indicates that tree-based learners J48 and PART are preferred methods for classification of malicious traffic.

When it comes to J48 and PART there was no clear winner (see Fig. 3). For both J48 and PART pruned trees resulted in as good as, or better probability of detection than the unpruned trees. With respect to the tree size, the number of leaves in J48 trees was comparable to the number of rules in PART, for all datasets. Pruning the trees reduced these numbers approximately in half. For example, for the WebDBAdmin dataset, unpruned J48 tree had eight leaves, while pruned J48 tree had five leaves. Similarly, unpruned PART had eight rules and pruned PART had four rules. Interestingly, the size of the trees and the number of rules for the Web 2.0 II dataset was significantly bigger than those generated for the other datasets. For example, unpruned J48 tree for Web 2.0 II had 59 leaves and 117 nodes, while pruned J48 tree had 33 leaves and 65 nodes. This is due to the fact that Web2.0 II dataset contained more similar instances of attacks and vulnerability scans, and therefore more complex rules were required to divide them into two classes.

5.3.3. *Analysis of the best predictive features*

In this section we analyze the features with best predictive power, which is important because they help understanding how attacks and vulnerability scans differ. Furthermore, we explore whether the features with best predictive power were consistent across different datasets, that is, answer RQ6.

When comparing the small subsets of best predicting features given in Table 12 we observe a significant consistency across datasets (i.e., features (25), (26), and (28) appeared in all three datsets, and feature (10) appeared in the two Web 2.0 datasets). To further explore this issue, we considered the top ten features for each data set ordered from the most to least informative based on the information gain, which are shown in Table 13.

Seven features – (2) Bytes transferred, (10) Number of requests with POST method, and features (25) through (29) which are related to the length of the request substrings within a session – appeared among top ten features in all three datasets. The bytes transferred (i.e., feature (2)) was among features with good predictive power because there were groups of vulnerability scans sessions, as well as groups of attack sessions that had the same number of bytes transferred. However, there was no clear difference between the values of bytes transferred in vulnerability scan sessions and attack sessions. As expected, the number of requests with POST method (i.e., feature (10)) played important role in distinguishing attacks from vulnerability scans. Thus, while all vulnerability scans had zero requests with POST method, attacks had zero or more requests with POST method. For example, posting spam on wiki or blog in Web 2.0 I and Web 2.0 II datasets included at least one request with POST method. Other features that played a prominent role in classifying malicious sessions were the features (25) through (29), which are related to the length of the request substrings within a session. We looked carefully into the sessions and noticed that attacks tended to have longer request substrings than vulnerability scans.

Table 13 – Top ten features for each dataset, ordered from the most to least informative based on the information gain.

Dataset	(ID) Feature name
Web 2.0 I	(10) Number of requests with POST method
	(28) Max length of request substrings
	(26) Median length of request substrings
	(25) Mean length of request substrings
	(27) Min length of request substrings
	(29) Standard deviation of length of request substrings
	(30) Mean number of parameters in requests
	(33) Max number of parameters in requests
	(2) Bytes transferred
	(34) Standard deviation of number of parameters in requests
Web 2.0 II	(28) Max length of request substring
	(25) Mean length of request substring
	(10) Number of requests with POST method
	(26) Median length of request substring
	(29) Standard deviation of length of request substring
	(2) Bytes transferred
	(30) Mean number of parameters in requests
	(27) Min length of requests substring
	(17) Number of requests to static files
	(34) Standard deviation of number of parameters in requests
WebDB-Admin	(24) Number of requests with 'Server' errors
	(28) Max length of request substrings
	(26) Median length request substrings
	(17) Number of requests to static files
	(25) Mean length of request substrings
	(2) Bytes transferred
	(10) Number of requests with POST method
	(27) Min length of request substrings
	(21) Number of requests with 'Success' status code
	(29) Standard deviation of length of request substrings

Features which deal with the number of parameters in requests within a session (i.e., (30) and (34)) were selected in the top ten features only for the Web 2.0 I and Web 2.0 II datasets. This is because Web 2.0 honeypots contained interactive content (i.e., ran MediaWiki and Wordpress). By studying these three features we noticed that attacks tended to pass more parameters in a session than vulnerability scans.

On the other side, WebDBAdmin dataset had two features related to the request status code among the top ten features which did not appear in either of the two Web 2.0 datasets: (24) Number of requests with 'Server' errors and (21) Number of requests with 'Success' status code. Looking closely into the raw data we noticed that attacks tended to cause the server to reply with 'Server' error (i.e., status code 500 in our case) more often than vulnerability scans. To be more precise, 26 out of the 36 attack sessions had at least one request with status code 500, as opposed to only 6 vulnerability scan sessions out of total 513 vulnerability scan sessions. Among 26 attacks with non-zero number of server errors we identified known attacks such as CVE-2010-1151, which aimed to inject arbitrary PHP code into a configuration file. In order for the CVE-2009-1151 to be successful the administrator must not have deleted the '/config/' directory within the '/phpMyAdmin/' directory, which was not the case with our honeypot and therefore the attack was not successful. Instead the system returned server error. Most of the requests with the success status code (i.e., feature (21)) in the attack sessions were accessing 'setup.php' in the '/phpMyAdmin/scripts/' directory.

The features with best predictive power may be different for other datasets collected by systems that run different configurations and therefore are likely to be exposed to different types of malicious activities, with different behaviors. Hence, instead of advocating a particular subset of features as the best predictors for all systems, we recommend that classification of malicious activities should always include feature selection method that would help identifying the best subset for a particular system.

5.3.4. *Using PART rules to characterize malicious web sessions*

Tree-based learners allow for high level of interpretability of the results. Table 14 presents several examples of PART rules, which provide information on vulnerability scan and attack patterns observed on our honeypots. These rules can be used to improve the intrusion protection and detection systems, such as firewalls and intrusion detection tools, by dynamically updating the patterns of observed attacks.

As it can be seen in Table 14, different rules covered significantly different numbers of instances. For example, the first attack rule for the Web 2.0 II dataset classified correctly 2178 out of the 2726 attack sessions. The second attack rule, which is more complex, classified only 98 of the 2726 attack sessions.

6. Summary of the main findings

Next we summarize the main findings, specifically addressing the research questions and providing a reference to the section with detailed analysis. With respect to research question RQ1, which is related to the types of vulnerability scans and attacks launched on Web systems, the main findings are:

- *Search-based strategy dominated the way attackers reached the Web systems* (Section 4.2.) A clear indication for this observation is the fact that for each dataset the advertised honeypot had significantly more malicious sessions than its unadvertised counterpart. Moreover, no vulnerability scan sessions and attack sessions aimed at Web 2.0 applications (i.e., Wiki and Blog) appeared on WebDBAdmin honeypots (see Table 5). Similarly, none or very little (less than 0.3%) malicious sessions related to phpMyAdmin vulnerability scans appeared on Web 2.0 I and Web 2.0 II honeypots (see Tables 3and 4).

Table 14 – Example PART rules for vulnerability scans and attacks.

PART rule	Dataset	Malicious activity	Classified/All instances
Number of requests with POST method = 0 AND Median length of request substrings <= 62 AND Median length of request substrings > 5	Web 2.0 I	Vulnerability scan	61/824
Mean number of parameters in requests <= 1 AND Number of requests to dynamic applications files <= 13 AND Min length of request substrings <= 15	Web 2.0 I	Vulnerability scan	563/824
Number of requests with POST method > 0 AND Number of requests with Success status code > 1	Web 2.0 I	Attack	268/293
Bytes transferred <= 29,762 AND Median length of request substrings > 29 AND Median length of request substrings <= 42 AND Max length of request substrings <= 107	Web 2.0 II	Vulnerability scan	11/2059
Max number of parameters in requests<= 1 AND Server Error <= 0 AND Client error <= 0 AND Remote sites injected <= 0 AND Pictures <= 1 AND Standard deviation of number of parameters <= 0.55	Web 2.0 II	Vulnerability scan	1240/2059
Number of requests with POST method > 0 AND Number of requests with Success status code > 0	Web 2.0 II	Attack	2178/2726
Bytes transferred > 41,609 AND Max length of request substrings > 59 AND Max length of request substrings <= 210	Web 2.0 II	Attack	98/2726
Number of requests with Server error <= 0 AND Number of requests with Success status code <= 5 AND Number of requests with Client error <= 0 AND Max length of request substring <= 8	WebDBAdmin	Vulnerability scan	402/513
Number of requests with POST method > 0	WebDBAdmin	Attack	6/36

- *IP-based strategy, although not often, was also used to target port 80* (Section 4.2). Thus, *DFind* and *Other fingerprinting* scans, as well as some of the RFI and SQL injection attacks appeared in all datasets, for both the advertised and unadvertised honeypots. The most prominent IP-based strategy attack was the DoS attack to both advertised and unadvertised Web 2.0 I honeypots.
- Systematic scanning or attacking of multiple ports on a same server or a single port on multiple servers (i.e., vertical and horizontal visits) existed, but contributed only 1–2% of the traffic (Section 4.1). In most cases these systematic attacker activities used IP-based strategy to reach the server.
- *Heavy-hitter sessions skewed the profiles of malicious traffic* (Section 4.1 and Section 4.3.1). An example of this type of behavior is the DoS attacks in Web 2.0 I dataset; only several DoS sessions had thousands of requests. This is a typical behavior for phenomena that follow heavy-tailed distributions, in which very large values (in this example the number of requests per session) can occur with non-negligible probability and although rare can have the mass of the probability distribution function.
- *Heavy-hitters behavior was also evident with respect to requests/ sessions generated by a single attacker* (Section 4.1 and Section 4.3.2). Thus, the analysis of countries of origin pointed out several examples of individual attackers who generated significant amount of connections. The fact that the number of malicious HTTP sessions/requests generated by unique source IP addresses followed skewed or heavy-tailed distributions statistically proves the same point – a huge percentage of the malicious traffic was generated by a few attackers.
- *Breaking user passwords is among most frequent types of attacks, which indicates that using weak passwords is still among the weakest links in systems security* (Section 4.1). This is evident by the facts that SSH (22) port was the second most popular port on all honypots, with over 90% of the SSH packets on each honeypot being part of dictionary password cracking attacks. The advertised and unadvertised honeypots received approximately the same amount of SSH traffic, which indicates the use of the IP-based strategy to reach the servers.

With respect to the RQ2, which is related to the statistical characteristics of malicious Web sessions, we found that the tails of Session duration, Number of request per session, and Bytes transferred per session were all described well with heavy tailed or skewed distributions, which means that with non-negligible probability these session attributes can take extremely large values (see Section 4.3.1). Specifically,

- Session duration for all three datasets was modeled well with Pareto distributions.

- Number of request per session of Web 2.0 I and Web 2.0 II datasets were modeled with heavy-tailed Pareto distribution. Lognormal distribution modeled well the number of request per session in WebDBAdmin datasets.
- Bytes transferred per session attribute followed Paerto distribution for all three datasets.

For RQ3, which is focused on exploring whether malicious behaviors are invariant (1) over time (i.e., for the same system configuration during different periods of time) and (2) across systems running different services, we concluded the following (Section 4.1 and Section 4.2):

- *The two advertised Web 2.0 datasets had similar types of vulnerability scans and attacks. The amount of malicious sessions and their distribution across different classes, however, were different*. Web 2.0 II dataset had significantly more sessions and somewhat more requests than Web 2.0 I dataset. Moreover, Web 2.0 II dataset had significantly higher percentage of *Spam on Blog* sessions than Web 2.0 I dataset (see Tables 3and 4). The values of each session attribute showed statistically significant difference for Web 2.0 I and Web 2.0 II datasets, even though they were modeled well with the same distribution.
- *Web servers running different services experienced almost complementary profiles of malicious activities*, i.e., there were no vulnerability scan sessions and attack sessions aimed at Web 2.0 applications in the WebDBAdmin dataset (see Table 5), nor there were many malicious sessions related to phpMyAdmin in Web 2.0 I and Web 2.0 II datasets (see Tables 3and 4). This is a consequence of using search-based strategy to identify and attack Web servers. As expected, the values of each session attribute showed statistically significant difference for Web 2.0 II and WebDBAdmin datasets.
- *Web 2.0 applications seem to be more attractive targets for attackers than database administration application*. Web 2.0 I and Web 2.0 II datasets had more TCP, as well as HTTP traffic than WebDBAdmin dataset. Posting spam on wiki and blog dominated the attack sessions on Web 2.0 advertised honeypot, which confirms the seriousness of the spamming to these applications.

With respect to the research questions RQ4 - RQ6, which are related to the use of supervised machine learning methods to automatically distinguish between attack sessions and vulnerability scan sessions, the main findings include

- *Supervised machine learning methods distinguished successfully between Web attacks and vulnerability scans* (Section 5.3.1). For two of the datsets all learners had probability of detection (i.e., recall) over 92% and the best learner had probability of detection over 97%. The results were satisfactory even for the third dataset, which was very imbalanced (i.e., had less than 7% attacks sessions), with 75% recall values produced by the best learner.
- *For realistic assessment of learners' performance metrics such as probability of detection and probability of false alarm have to be used because accuracy values were very high even in cases when the attack detection was moderate* (Section 5.3.1).
- *Attacks differed from vulnerability scans only in a small number of session characteristics, from four to six depending on the dataset* (Section 5.3.2). More importantly, predictions based on these best predictors had performance very close to predictions based on all 43 features.
- *Tree based methods J48 and PART performed better than SVM in terms of probability of detection. Additionally, J48 and PART required less features (i.e., session attributes), as well as they executed much faster* (Section 5.3.2). The pruned versions of J48 and PART were approximately half the size of the unpruned versions and yet produced close, and in some cases even better performance than the unpruned tree versions.
- *Significant number of features appeared among best predictors consistently across multiple datasets. However, the order of features based on the information gain was slightly different across datasets and there were some different features, especially for systems running different applications* (Sections 5.3.2 and 5.3.3). Hence, instead of advocating a particular subset of features as the best predictors for all systems, we recommend using feature selection method to identify the best subset for a particular system.

7. Conclusion

This paper presents a comparative analysis of attacker activities on typical multi-tier Web servers based on data collected by high-interaction honeypot systems. The presented work is based on a sound experimental design, which allowed us to make fair comparison between vulnerability scans and attacks observed across servers running different services and during different time periods.

The first set of concluding remarks is related to the experimental design. To allow for realistic studies of attacker activities it is of utmost importance to deploy honeypots that run typical configurations and fully functional systems. This approach allowed us to observe phenomena that would not have surfaced in a collection of independently running applications typically deployed on honeypots in related work. In addition, Web-based honeypots have to be advertised to enable for the use of search-based strategy, which appears to dominate the way attackers reach Web servers. This point is obvious when comparing the amounts of malicious traffic on the advertised and unadvertised honeypots. Surprisingly, most of the honeypots from related work that ran Web applications were not advertised.

The second set of concluding remarks is with respect to the comparison of malicious cyber activities across different datasets. It appears that when it comes to attacking 'any server' rather than a major governmental or e-commerce server, the easiest ways to attack the server seem to dominate. In our case, these included password cracking attacks on SSH user accounts and vulnerability scans and attacks (in a form of spam) on Web 2.0 applications which due to their interactive nature allow access by default. Significantly less malicious activities were aimed at specific known vulnerabilities. Comparing the results across multiple datasets we observed some invariant characteristics, but also some differences. The invariant characteristics included the dominance of the search-based strategy used to identify and attack the servers, heavy-tailed characteristics of malicious session attributes, and heavy-tailed or skewed distributions describing the HTTP traffic generated by unique attackers. On the other side, servers running different services had almost complementary profiles of vulnerability scan and attack types. Moreover, the attributes of malicious session were statistically different across the same system configuration during different periods of time, as well as across different system configurations running at the same period of time.

The last set of concluding remarks are related to using machine learning techniques for classification of malicious Web sessions. Our results showed that supervised machine learning algorithms can be used to successfully distinguish between vulnerability scan sessions and attack sessions. Even in a highly imbalanced dataset, with many different attacks each with small number of instances, the best algorithm was able to detect 3/4 of the attack sessions successfully. The results also showed that vulnerability scan sessions and attack sessions differ only in a small number of session characteristics and that many features with good predictive power were consistent across the three datasets.

Potentially, there is a significant benefit from qualitative and quantitative analysis of malicious traffic. For example, accumulated knowledge can be used for generating realistic malicious traffic for verification and validation of systems' security or to help the intrusion detection process. Intrusion prevention and detection have to evolve with ever changing attacker behaviors. Using advertised honeypots with realistic configurations to collect malicious traffic and applying the methods and techniques presented in this paper allow for dynamic characterization and classification of malicious cyber attacks in support of adaptive intrusion prevention and detection.

Acknowledgments

This work was funded in part by the National Science Foundation under the grants CNS-0447715 and CCF-0916284. The authors thank David Krovich, Jonathan Lynch and J. Alex Baker for their support with the experimental setup and data extraction process. We also thank the anonymous reviewers for their comments and suggestions.

Appendix A. Background on distribution fitting

A random variable X with cumulative distribution function $F(x)$ is said to be heavy-tailed if

$$1 - F(x) = x^{-\alpha}L(x) \tag{A1}$$

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 scale parameter $b>0$. The cumulative distribution function (CDF) for Pareto distribution is:

$$F(x) = P[X \leq x] = 1 - \left(\frac{b}{x}\right)^{\alpha}, \quad b \leq x < +\infty \tag{A2}$$

There is an important qualitative property of the moments of heavy-tailed distributions. If X is heavy-tailed with parameter α then its first $m < \alpha$ moments $E[X^m]$ are finite and its all higher moments are infinite. Thus, 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. As α decreases an arbitrary large portion of the probability mass may be present in the tail of the distribution. 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 (Goseva-Popstojanova et al., 2006a). 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 an evidence of a heavy-tailed distribution. If this is the case, we select 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. It estimates α as a function of the k largest elements in the dataset. 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 dataset. We pick $k<n$ and compute the Hill estimator

$$H_{k,n} = \frac{1}{k}\sum_{i=1}^{k} \log X_{(i)} - \log X_{(k+1)}. \tag{A3}$$

For each value of k we estimate the tail index parameter as $\alpha_{k,n} = 1/H_{k,n}$. 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 cases when the Pareto distribution was not a good fit, we fitted the lognormal distribution which is a skewed distribution with two parameters μ and σ. Lognormal distribution, unlike the Pareto distribution, always has a finite variance. The cumulative distribution function (CDF) for two parameter Lognormal distribution is:

$$F(x) = \Phi\left(\frac{\ln x - \mu}{\sigma}\right), \quad 0 < x < +\infty \tag{A4}$$

where Φ is the Laplace Integral.

For each of the distributions presented in this paper we used the Anderson–Darling goodness-of-fit test (Anderson and Darling, 1954). This test is more powerful than better known Kolmogorov–Smirnov and χ^2 tests, particularly for detecting deviations in the tail of a distribution. Thus, Anderson–Darling test gives more weight to the tails than the Kolmogorov–Smirnov test.

Appendix B. Null hypotheses and results of the statistical tests for the central tendency of session attributes

In this appendix we present the formal hypotheses for the cental tendency of each of the malicious session attributes for the statistical tests presented in Section 4.3.

Table 15 – Results of the Kruskal–Wallis for each session attribute.

Attribute	p value
Session duration	$2.2 \cdot 10^{-16}$
Number of requests per session	$2.2 \cdot 10^{-16}$
Bytes transferred per session	$2.2 \cdot 10^{-16}$

The null hypotheses related to the comparison of each session attribute across the three datasets are as follows:

$H1_0$: Distribution functions of *Session duration* are identical across all three datasets.

$H2_0$: Distribution functions of *Number of requests* per *session* are identical across all three datasets.

$H3_0$: Distribution functions of *Bytes transferred* per *session* are identical across all three datasets.

The alternative hypothesis for each session attribute is that at least one of the datasets yields larger observations than at least one of the other datasets.

Based on the results of the Kruskal–Wallis test shown in Table 15 all three null hypotheses were rejected in favor of the corresponding alternative hypotheses. In other words, for each attribute, at least one of the median values across the three datasets is different, that is, the differences between median values are non-random and statistically significant.

Once the null hypothesis is rejected, one can use a multiple comparisons method to determine which pair(s) tend to differ (Siegel and Castellan, 1988). Those pairs which have observed differences higher than the critical value are considered statistically different at the given probability (p level).

The null hypotheses related to the comparison of each session attribute across all possible combinations of dataset pairs are as follows:

For *Session duration*:

$H4_0$: There is no significant difference between the median *Session duration* in Web 2.0 I and Web 2.0 II datasets.

$H5_0$: There is no significant difference between the median *Session duration* in Web 2.0 II and WebDBAdmin datasets.

$H6_0$: There is no significant difference between the median *Session duration* in Web 2.0 I and WebDBAdmin datasets.

For *Number of requests per session*:

$H7_0$: There is no significant difference between the median *Number of requests* per *session* in Web 2.0 I and Web 2.0 II datasets.

$H8_0$: There is no significant difference between the median *Number of requests* per *session* in Web 2.0 II and WebDBAdmin datasets.

Table 16 – Results of multiple comparisons test for session duration.

Datasets	Observed difference	Critical difference
Web 2.0 I and Web 2.0 II	213.3686	148.1571
Web 2.0 II and WebDBAdmin	779.2389	200.9046
Web 2.0 I and WebDBAdmin	565.8703	232.3889

Table 17 – Results of multiple comparisons test for Number of requests per session.

Datasets	Observed difference	Critical difference
Web 2.0 I and Web 2.0 II	523.9119	148.1571
Web 2.0 II and WebDBAdmin	474.6976	200.9046
Web 2.0 I and WebDBAdmin	998.6095	232.3889

Table 18 – Results of multiple comparisons test for Bytes transferred per session.

Datasets	Observed difference	Critical difference
Web 2.0 I and Web 2.0 II	974.9832	148.1571
Web 2.0 II and WebDBAdmin	1737.0119	200.9046
Web 2.0 I and WebDBAdmin	762.0286	232.3889

$H9_0$: There is no significant difference between the median *Number of requests* per *session* in Web 2.0 I and WebDBAdmin datasets.

For *Bytes transferred per session*:

$H10_0$: There is no significant difference between the median *Bytes transferred* per *session* in Web 2.0 I and Web 2.0 II datasets.

$H11_0$: There is no significant difference between the median *Bytes transferred* per *session* in Web 2.0 II and WebDBAdmin datasets.

$H12_0$: There is no significant difference between the median *Bytes transferred* per *session* in Web 2.0 I and WebDBAdmin datasets.

The results of the multiple comparisons tests for Session duration, Number of requests per session, and Bytes transferred per session are shown in Tables 16–18, respectively. Since the observed differences between mean ranks of each dataset pair, for each session attribute were greater than the corresponding critical differences, the null hypotheses $H4_0$ – $H12_0$ are all rejected in favor of the alternative hypotheses that there is a difference between all pairs of datasets, for each session attribute.

REFERENCES

Alata E, Nicomette V, Kaaniche M, Dacier M, Herrb M. Lessons learned from the deployment of a high-interaction honeypot. In: 6th European dependable computing conference (EDCC'06); 2006. pp. 39–46.

Almgren M, Debar H, Dacier M. A lightweight tool for detecting Web server attacks. In: ISOC Symposium on network and distributed systems security; 2000. pp. 157–70.

Anderson T, Darling DA. A test of goodness of fit. J Am Statistical Assoc 1954;49(268):765–9.

Bailey M, Cooke E, Jahanian F, Nazario J, Watson D. The Internet motion sensor: a distributed blackhole monitoring system. In:

12th ISOC Symposium on network and distributed systems security (SNDSS); 2005. pp. 167–79.

Bailey M, Oberheide J, Andersen J, Mao ZM, Jahanian F, Nazario J. Automated classification and analysis of Internet malware. In: Recent advances in intrusion detection. Springer; 2007. pp. 178–97.

Barford P, Chen Y, Goyal A, Li Z, Paxson V, Yegneswaran V. Employing honeynets for network situational awarenessIn Cyber situational awareness; 2010. pp. 71–102.

Bayer U, Comparetti PM, Hlauschek C, Kruegel C, Kirda E. Scalable, behavior-based malware clustering. In: Network and distributed system security symposium (NDSS); 2009.

Berthier R, Korman D, Cukier M, Hiltunen M, Vesonder G, Sheleheda D. On the comparison of network attack datasets: an empirical analysis. In: 11th IEEE high assurance systems engineering symposium (HASE 2008); 2008. pp. 39–48.

Bloomfield R, Gashi I, Povyakalo A, Stankovic V. Comparison of empirical data from two honeynets and a distributed honeypot network. In: 19th international symposium on software reliability engineering (ISSRE 2008); 2008. pp. 219–28.

Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. In: 5th Annual workshop on computational learning theory; 1992. pp. 144–52.

Chen PT, Laih CS, Pouget F, Dacier M. Comparative survey of local honeypot sensors to assist network forensics. In: 1st international workshop on systematic approaches to digital forensic engineering; 2005. pp. 120–32.

Choo K-KR. The cyber threat landscape: challenges and future research directions. Comput Secur 2011;30:719–31.

Cukier M, Berthier R, Panjwani S, Tan S. A statistical analysis of attack data to separate attacks. In: International conference on dependable systems and networks (DSN 2006); 2006. pp. 383–92.

CyberCrime. 2012 Cost of cyber crime study: United States; 2012 [Ponemon Institute research report].

Dacier M, Pouget F, Debar H. Honeypots: practical means to validate malicious fault assumptions. In: 10th IEEE Pacific rim international symposium on dependable computing (PRDC 2004); 2004. pp. 383–8.

DARPA. DARPA intrusion detection evaluation project http://www.ll.mit.edu/mission/communications/ist/CST/index.html; 1999.

Estevez-Tapiador JM, Garcia-Teodoro P, Diaz-Verdejo JE. Detection of Web-based attacks through Markovian protocol parsing. In: 10th IEEE Symposium on computers and communications (ISCC); 2005. pp. 457–62.

Frank E, Witten IH. Generating accurate rule sets without global optimization. In: 15th International conference on machine learning; 1998.

Garcia V, Monroy R, Quintana M. Web attack detection using ID3In Professional practice in artificial intelligence; 2006. pp. 323–32.

Gordon LA, Loeb MP, Lucyshyn W, Richardson R. CSI/FBI computer crime and security survey. Computer Security Institute; 2005.

Goseva-Popstojanova K, Anastasovski G, Pantev R. Classification of malicious Web sessions. In: 21st International conference on computer communications and networks (ICCCN); 2012. pp. 1–9.

Goseva-Popstojanova K, Li F, Wang X, Sangle A. A contribution towards solving the Web workload puzzle. In: International conference on dependable systems and networks (DSN 2006); 2006. pp. 505–16.

Goseva-Popstojanova K, Miller B, Pantev R, Dimitrijevikj A. Empirical analysis of attackers activity on multi-tier Web systems. In: 24th IEEE international conference on advanced information networking and applications (AINA); 2010. pp. 781–8.

Goseva-Popstojanova K, Pantev R, Dimitrijevikj A, Miller B. Quantification of attackers activities on servers running Web 2.0 applications. In: 9th IEEE international Symposium on network computing and applications (NCA); 2010. pp. 108–16.

Goseva-Popstojanova K, Singh AD, Mazimdar S, Li F. Empirical characterization of session–based workload and reliability for Web servers. Empir Softw Eng 2006b;11(1):71–117.

Hohn N, Veitch D, Ye T. Splitting and merging of packet traffic: measurement and modelling. Perform Eval 2005;62(1):164–77.

Honeynet. The honeynet project http://www.honeynet.org/; 1999.

Julisch K. Data mining for intrusion detectionIn Applications of data mining in computer security; 2002. pp. 33–58.

Kaaniche M, Alata E, Nicomette V, Deswarte Y, Dacier M. Empirical analysis and statistical modelling of attack processes based on honeypots. In: Workshop on empirical evaluation of dependability and security; 2006.

KDD. KDD cup 1999 data; 1999.

Kruegel C, Vigna G, Robertson W. A multi-model approach to the detection of Web-based attacks. Comput Net 2005;48(5):717–38.

Leland WE, Taqqu MS, Willinger W, Wilson DV. On the self-similar nature of Ethernet traffic. IEEE/ACM Trans Netw 1994;2(1):1–15.

Leurrecom. Leurrecom.org honeypot project http://www.scoop.it/t/ict-security-tools/p/860164618/welcome-to-leurrecom-org-honeypot-project; 2003.

Liao H-J, Lin C-HR, Lin Y-C, Tung K-Y. Intrusion detection system: a comprehensive review. J Netw Comput Appl 2013;36(1):16–24.

Liu H, Yu L. Toward integrating feature selection algorithms for classification and clustering. IEEE Trans Knowl Data Eng 2005;17(4):491–502.

Lourenco AG, Belo OO. Catching Web crawlers in the act. In: 6th International conference on web Engineering; 2006. pp. 265–72.

McGrew R, Vaughn RB. Experiences with honeypot systems: development, deployment, and analysis. In: 39th Annual Hawaii international conference on system sciences (HICSS'06), vol. 9; 2006. 220a–220a.

Moore D, Shannon C, Brown DJ, Voelker GM, Savage S. Inferring Internet denial-of-service activity. ACM Trans Comput Syst 2006;24(2):115–39.

Noel S, Wijesekera D, Youman C. Modern intrusion detection, data mining, and degrees of attack guilt. In: Applications of data mining in computer security. Springer; 2002. pp. 1–31.

NVD. National vulnerability database http://nvd.nist.gov/; 2013.

Pang R, Yegneswaran V, Barford P, Paxson V, Peterson L. Characteristics of Internet background radiation. In: 4th ACM SIGCOMM conference on Internet measurement; 2004. pp. 27–40.

Panjwani S, Tan S, Jarrin KM, Cukier M. An experimental evaluation to determine if port scans are precursors to an attack. In: International conference on dependable systems and networks (DSN 2005); 2005. pp. 602–11.

Pantev R. Analysis and classification of current trends in malicious http traffic. Matser Thesis. West Virginia University; 2011.

Paxson V, Floyd S. Wide area traffic: the failure of Poisson modeling. IEEE/ACM Trans Netw 1995;3(3):226–44.

Perdisci R, Lee W, Feamster N. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In: 7th USENIX conference on networked systems design and implementation; 2010. 26–26.

Pouget F, Dacier M, Pham VH. Leurre. com: on the advantages of deploying a large scale distributed honeypot platform. In: E-Crime and computer conference (ECCE 2005); 2005.

Quinlan JR. Simplifying decision trees. Int J Man–Machine Stud 1987;27(3):221–34.

Quinlan JR. C4. 5: programs for machine learning, vol. 1. Morgan Kaufmann; 1993.
R Development Core Team. R: a Language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2008.
Ramsbrock D, Berthier R, Cukier M. Profiling attacker behavior following SSH compromises. In: 37th Annual IEEE/IFIP international conference on dependable systems and networks (DSN'07); 2007. pp. 119–24.
Salles-Loustau G, Berthier R, Collange E, Sobesto B, Cukier M. Characterizing attackers and attacks: an empirical study. In: 17th IEEE Pacific rim international symposium on dependable computing (PRDC); 2011. pp. 174–83.
SANS. Sans top-20 2007 security risks (2007 annual update) http://www.sans.org/top20/; 2007.
Siegel S, Castellan Jr NJ. Nonparametric statistics for the behavioral Sciences. McGraw-Hill; 1988.
Vrable M, Ma J, Chen J, Moore D, Vandekieft E, Snoeren AC, et al. Scalability, fidelity, and containment in the Potemkin virtual honeyfarm. ACM SIGOPS Oper Syst Rev 2005;39(5):148–62.
WhiteHat. Whitehat secuirty website statistics report https://www.whitehatsec.com/resource/stats.html; 2012.
Woo S-W, Joh H, Alhazmi OH, Malaiya YK. Modeling vulnerability discovery process in Apache and IIS HTTP servers. Comput Secur 2011;30(1):50–62.
Xu K, Zhang Z-L, Bhattacharyya S. Internet traffic behavior profiling for network security monitoring. IEEE/ACM Trans Netw 2008;16(6):1241–52.
Yegneswaran V, Barford P, Paxson V. Using honeynets for Iternet situational awareness. In: 4th Workshop on hot topics in networks (HotNets IV); 2005. pp. 17–22.
Yegneswaran V, Barford P, Ullrich J. Internet intrusions: global characteristics and prevalence. ACM SIGMETRICS Perform Eval Rev 2003;31(1):138–47.

Katerina Goseva-Popstojanova is a Robert C. Byrd Associate Professor in the Lane Department of Computer Science and Electrical Engineering at West Virginia University, Morgantown, WV. Her research interests are in cybersecurity and assessment and prediction of reliability, availability, and performance of software and computer systems. She has served as a program chair of the 18th IEEE International Symposium on Software Reliability Engineering, a guest editor of the IEEE Transactions on Software Engineering special section on software dependability (May/June 2010), and as a member of program and organizing committees of numerous international conferences and workshops.

Goce Anastasovski is currently a student pursuing his M.S. degree in Computer Science in the Lane Department of Computer Science and Electrical Engineering at West Virginia University. Prior to joining West Virginia University he received his B.S. degree in Computer Science and Computer Engineering from Ss. Cyril and Methodius University in Skopje, Macedonia. His current research interests include security of web systems, data mining and machine learning.

Ana Dimitrijevikj received her M.S. degree in Computer Science from West Virginia University in 2011 and B.S. degree in Computer Science from Ss. Cyril and Methodius University, Skopje, Macedonia in 2006. Her research interests include cybersecurity, pattern recondition, and statistical modeling.

Risto Pantev received his M.S. degree in Computer Science from West Virginia University in 2011 and the B.S. degree in Computer Science from Ss. Cyril and Methodius University, Skopje, Macedonia in 2006. He is currently affiliated with Microsoft, Redmond, WA. His research interests include cybersecurity, anomaly detection, machine learning, and pattern recondition.

Brandon Miller is a Software Engineer at KeyLogic Systems, Inc of Morgantown, WV. He currently works onsite at the NASA Independent Verification and Validation (IV&V) Facility in Fairmont, WV. He earned a Bachelor of Science (2008) and Master of Science (2009) in Computer Science at West Virginia University in Morgantown, WV. His Master's thesis topic was "Analysis of Attacks on Web Based Systems." He is also a certified Project Management Professional (PMP).