2009

Analysis of attacks on Web based applications

Brandon Scott Miller
West Virginia University

Analysis of Attacks on Web Based Applications

by

Brandon Scott Miller

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

Master of Science
in
Computer Science

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

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2009

Keywords: Web 2.0, honeypots, port scans, vulnerability scans, attacks

Abstract

Analysis of Attacks on Web Based Applications

by

Brandon Scott Miller
Master of Science in Computer Science

West Virginia University

Katerina Goseva-Popstojanova, Ph.D., Chair

As the technology used to power Web-based applications continues to evolve, new security threats are emerging. Web 2.0 technology provides attackers with a whole new array of vulnerabilities to exploit. In this thesis, we present an analysis of the attacker activity aimed at a typical Web server based on the data collected on two high interaction honeypots over a one month period of time. The configuration of the honeypots resembles the typical three tier architecture of many real world Web servers. Our honeypots ran on the Windows XP operating system and featured attractive attack targets such as the Microsoft IIS Web server, MySQL database, and two Web 2.0-based applications (Wordpress and MediaWiki). This configuration allows for attacks on a component directly or through the other components. Our analysis includes detailed inspection of the network traffic and IIS logs as well as investigation of the System logs, where appropriate. We also develop a pattern recognition approach to classify TCP connections as port scans or vulnerability scans/attacks. Some of the conclusions of our analysis include: (1) the vast majority of malicious traffic was over the TCP protocol, (2) the majority of malicious traffic was targeted at Windows file sharing, HTTP, and SSH ports, (3) most attackers found our Web server through search-based strategies rather than IP-based strategies, (4) most of the malicious traffic was generated by a few unique attackers.

Acknowledgements

I first want to thank my committee chair and advisor, Dr. Katerina Goseva-Popstojanova, for allowing me the opportunity to work with her and her students. I could not have completed this thesis without her constant guidance, encouragement, and support. I have very much enjoyed working with her and appreciate the time and effort she put forth to help me achieve my goals. I also would like to thank Dr. James Mooney and Dr. Arun Ross, who also served on my graduate committee. I have had the pleasure of taking courses with all members of my committee and have learned much from them. I appreciate their willingness to be members of my committee.

I also want to thank and acknowledge the other students I worked with in the research lab, in particular Risto Pantev and Ana Dimitrijevik. They have helped me in many ways throughout this research, and I have greatly enjoyed working with them.

I want to thank the CSEE Helpdesk and CEMR IT Staff, and Dave Krovich in particular, for all the help they gave me in getting the honeypots setup and connected to the network and helping me solve general systems issues.

I would also like to thank all of the professors and teachers I have had the pleasure of working with throughout my academic career. Many professors have influenced my life and I am sincerely grateful for all they have done for me. I would like to acknowledge my high school Computer Science teacher, Dan Wallace, in particular. Mr. Wallace realized my talent in this field early in my life and encouraged me to pursue Computer Science. The wealth of knowledge he shared with me is truly endless and he continues to inspire me, even after his passing.

Finally, I want to express my gratitude to my family and my mother and father in particular. They have given me encouragement and support my entire life and have made many sacrifices so I could attend college and earn an advanced degree. I can never adequately express how much their endless support and encouragement means to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Web applications have become the primary solution for the software needs of many businesses and normal computer users, such as Email, financial planning, and news. The unique set of vulnerabilities facing Web applications, in conjunction with their popularity, make them very attractive targets for malicious users. According to the SANS Institute 2007 annual update [52], half of the top 20 security risks were vulnerabilities in open source or custom built Web applications. Another study by Gartner Group shows that 75% of attacks are on Web based applications [53]. These studies indicate that vulnerabilities in Web based systems, as well as attacks exploiting these vulnerabilities, are on the rise. As the trend to use Web applications to serve increasingly sensitive services and information continues, the security of Web applications will become a more and more critical issue.

The next generation of Web application technology, Web 2.0, is becoming more and more widespread every day. In August 2007, it was reported in a McKinsey global survey that three fourths of surveyed senior executives believe that Web 2.0 technologies are strategic and investments in Web 2.0 applications will increase [1]. However, Web 2.0 Web sites are inherently at more security risk than traditional Web sites due to their nature of high user interaction [2]. The increasing popularity of Web 2.0-based applications make them an even more attractive target for attackers.

Web 2.0 encompasses a wide range of technologies and services. There is no widely agreed upon or concise definition of "Web 2.0" [1]. Web 2.0 has been defined by some as a new way to use existing technologies, such as XML and JavaScript [2]. It has also been defined as

any application that allows direct interaction with the user. However, Web 2.0 technologies are generally considered to have some or all of the following characteristics:

- they are flexible and facilitate creative reuse and dynamic updating by user contributions
- they have responsive user interfaces
- they encourage users to collaborate through content creation and modification
- they encourage networking between people with common interests
- they allow combining of existing applications and data into a single source

Web 2.0 encompasses a number of types of applications, such as:

- *Blogs*: Web sites where users post their opinions, thoughts, or life updates
- *Wikis*: Web sites where users collaborate to generate an online "encyclopedia", either about a specific topic or about anything and everything, such as Wikipedia
- *Social Networking*: Web sites which allow users to connect with each other to share information such as messages, pictures, and videos
- *Mashups*: Web sites which collect information and services from other sources and displays them on the same page
- *RSS*: short for "really simple syndication", RSS is used to syndicate summarized content from Web pages via XML

The feature that all of these have in common is that each application depends on user interaction to generate new content and update old content [1].

These applications are based on a wide array of technologies that allow interaction with the user. Most Web 2.0 applications require a database, such as MySQL or Postgres, to store user generated data. Many developers use a technology called Asynchronous JavaScript and XML (AJAX). AJAX allows the user interface to be highly interactive and responsive to the user. Applications that use AJAX can make a server call and retrieve new data

while simultaneously updating the Web page without reloading the entire page. This allows the user to continuously interact with the application, even while it is updating the page. Common uses of this type of technology can be found in Google Maps, where the user can navigate through the map by clicking and dragging while the map updates and redisplays without reloading the entire page. Other development approaches include Adobe Flex [20], which is based on Adobe Flash, and Google Web Toolkit [21], which is an open source Java development framework for AJAX applications [1],[2]. Web 2.0 applications can be based on other technologies, such as PHP or ASP, and tend to be developed as open source software.

Web 2.0 applications are the targets of unique types of attacks [2],[27]:

- A *code/script injection* attack occurs when attackers attempt to exploit vulnerabilities in the Web application's interface with the underlying operating system in order to execute arbitrary code on the operating system. One way of executing such an attack is including operating system commands in Web application URL's in order for them to be included in execution commands.

- *Remote code inclusion* is similar to code injection attacks in that arbitrary code is executed on the Web server by the attacker. However, in this case, the attacker will try to change the value of global variables in a manner that will cause malicious code to be executed.

- *SQL injection* attacks attempt to exploit form fields or URIs that are part of database queries. By placing carefully crafted strings in these fields, attackers can gain unauthorized access to the system.

- In a *Cross-site scripting* (XSS) attack, attackers enter data into an application that is later written back to another user. This could allow the insertion of arbitrary code into an HTML page, for example, which is later executed when another user views that page. In XSS attacks, other users, not the Web application itself, are targets of the attack [11]. The code executed on the victim's machine in an XSS attack usually gives attackers high privileges to the victim's system which allows the attacker to steal data or change settings. To exploit this type of vulnerability, attackers have to get

unsuspecting users to follow carefully crafted malicious URLs, such as ones similar to ".../press-this.php/?ajax=box&i=%3Cscript(document.cookie)%3C/script" [22] (an exploit for WordPress) which in turn gives the attackers access to the user's cookies.

- *Web 2.0 Worms* propagate in the background of an unsuspecting user's Internet browser. An example of a Web 2.0 worm is the Samy worm, which propagated through MySpace user accounts. When a user viewed an infected user's profile, their profile was in turn infected [48].

- In a *Cross site request forgery* (CSRF) attack, a hacker sends unauthorized requests to a Web site from the victim's computer. The attacker can use the computer's IP address as well as stored passwords and cookies to authenticate to a given Web site and gain access to the user's financial accounts and other sensitive information. This type of attack can be executed using HTTP GET requests with specific requests. For example, links such as "http://www.example.com/EditAccount?action=update&email=malicious@any.com" suggest that the link can be accessed by an authorized user with a cookie. However, such links can also be forged.

Web application technologies, particularly those employing Web 2.0, are now very prevalent on the Internet and thus attacks on them are on the rise. Also, Internet users are putting more and more sensitive information on the Internet every day. Naturally, attackers are attempting to infiltrate this data for criminal purposes such as identity theft, extortion, and financial fraud. Web 2.0 technologies give attackers a broad new range of vulnerabilities that can be exploited to support their activities. This provides motivation to study the patterns of attacks on Web based systems in order to see how the attackers exploit vulnerabilities and attempt to infiltrate computer systems.

1.1 Honeypots

Sifting through the large amounts of data present on production Web servers presents a "needle in the haystack" problem. Finding malicious data and attempts at attacks amongst the large amount of normal data is very difficult. In order to eliminate this problem, we use *honeypot* technology. A honeypot, quite simply, is a computer that is connected to a network but is not used by any legitimate users. Therefore, if anyone attempts to use the machine, it is either an accident or most likely an attack attempt on the machine [3]. This allows us to consider all traffic to the machine as malicious once we have filtered out "legitimate" system traffic such as DNS queries and logging.

We developed and deployed two high-interaction honeypots which have real services and real Web applications following the example of GenII honeypots used by the Honeynet Project [14]. Using a high-interaction honeypot allows us to give the appearance of a real Web server with all of the expected components and also guarantees that the honeypot gives authentic responses to any attack attempts. This provides the most realistic environment for our experiment. The configuration of our honeypots is not just a collection of random, unassociated applications. We designed the honeypots to closely resemble a real Web server with its databases and applications populated with a large amount of content. This is in contrast to the honeypots in the related work which typically have little or no content and do not have an obvious, meaningful functionality.

One of the honeypots we deployed is designated as the "advertised" honeypot. This honeypot is visible to the Internet through a method called "transparent linking." To use this method, links to the honeypot were placed on the home Web page of the Computer Science Department such that the links are not visible to humans but are visible to crawlers. The other honeypot is not advertised and is therefore not visible to crawlers. We refer to this honeypot as the "unadvertised" honeypot. The only way to reach the unadvertised honeypot is to connect to it directly. The purpose of having two honeypots is to provide a control to the experiment. This allows us to study attacks that are search-based (preceded by a discovery request) versus IP-based (come directly to the IP address of the honeypot), which are defined in [11].

Since our goal in this experiment is to observe and analyze the pattern of attacks on Web based systems, we built our honeypots in the same configuration as a typical Web server:

- Web (HTTP) Server (Microsoft's Internet Information Services, or IIS)
- Application Servers (PHP, Open SSH)
- Web Applications (MediaWiki, Wordpress)
- Database Server (MySQL)

For each component, we chose old versions of applications that are widely used. The purpose of installing old versions is to provide applications with a known set of vulnerabilities. We believe this makes our honeypots more attractive targets to attackers.

1.2 Data Analysis

We define the following terms for use in the rest of this thesis:

- *Connection*: a unique tuple {source IP address, source port, destination IP address, destination port} with a maximum inter-arrival time between packets in a connection of 64 seconds (this value referred to as a *threshold*) [32],[35]
- *Vertical Visit*: a sequence of ports visited on a single server by a single source IP address within a one hour period of time [34]
- *Horizontal Visit*: a visit to the same port on several machines in a subnet from a single source IP address within a one hour period of time [34]
- *Attack Source*: a single source IP address visiting a honeypot on a single day; a unique tuple {date; source IP address} [35]
- *Port Scan*: a TCP connection used to check for open ports and running services which can be exploited [38]
- *Vulnerability Scan*: a connection used to check for specific vulnerabilities in a system [38]

- *Attack*: an attempt to exploit a specific vulnerability or breach a system

Our analysis of the data collected by our honeypots begins with *descriptive statistics* gathered through the network traffic logs. We analyze characteristics of the unique source IP addresses who visit the honeypots. We first study the country of origin of the attackers. Then, we analyze the port sequences seen in horizontal and vertical visits. We also analyze the attack sources seen on each honeypot.

We then move to studying the traffic seen on the TCP protocol. We analyze the number of connections originated from each unique source IP address on each honeypot and develop a way to classify each TCP connection as either a port scan or a vulnerability scan/attack using a supervised learning approach. We then analyze the distribution of TCP traffic across different ports and then specifically analyze connections to the ports on which our applications (IIS, SSH, MySQL) run. The configuration of our honeypots allows us to analyze port scans, vulnerability scans, and attacks on different components of the Web based system. We also analyze whether attackers who perform port scans return for further attacks afterwards. This analysis is performed via a combination of the network traffic logs, the system logs (referred to as syslogs), and the application logs. The network traffic logs contain all packets sent over the Internet to our honeypots, the syslogs contain all logs generated by the Windows operating system including authentication attempts and service restarts, and the application logs contain information such as authentication attempts and queries to the database. Finally, we analyze the traffic seen on the other transport level protocols UDP and ICMP.

1.3 Organization of Thesis

The rest of this thesis is organized as follows. In Chapter 2 we review the related work and present our contributions to the field. In Chapter 3 we present our experimental set up and the tools used to perform our analysis. We present our detailed analysis of the data collected on the honeypots in Chapter 4. Finally, we conclude the thesis in Chapter 5.

Chapter 2

Related Work & Our Contribution

In this section, we review the related work in the area of honeypots and then discuss our previous work in this research area. Finally, we present the contributions of this thesis.

2.1 Related Work

There have been several studies and projects developed in the past decade which used honeypots to collect data about malicious activity on the Internet. One of these initiatives is the Leurre.com Honeynet Project [15], which was launched in 2003 by the Institute Eurecom and is based on a globally distributed system of low-interaction honeypots spanning five continents and thirty countries. Honeypots in the Leurrecom.org project are identically configured and based upon Honeyd [17], which is a framework for developing virtual honeypots which can be configured to emulate real applications. This project also has honeypots based on ScriptGen [18], which allow for greater interactions with attackers but still implement emulated systems rather than real systems. Traffic to all honeypots in this project is stored in a central database which can be accessed by all members of the project and outside researchers by request [16].

The Leurre.com project published a paper in 2005 regarding the results they had obtained up to that time. The honeypots used in this study were based on the Honeyd platform and emulated Windows 98, Windows NT Server, and Red Hat 7.3. There were 20 platforms located in 11 countries. They observed, via fingerprinting, that 80% to 90% of the attacking

machines ran Windows operating systems and attacked Windows services. This suggests that Microsoft Windows operating systems are very attractive targets for attackers. The authors also observed that the country of origin of the attackers and ports the attackers targeted varied greatly from honeypot to honeypot (evidence suggests this is because of different geographical location of the honeypots). However, they found that the USA and China were on the top three of attacking countries on each platform [19].

Bloomfield et al [43] compared the data collected from the Leurre.com project and two honeynets deployed by the researchers; a high-interaction honeynet on a corporate network and another on an external network purchased from an Internet Service Provider. The researchers found that there were a few ports where attacks were seen in large numbers across all datasets: TCP/80, TCP/135, TCP/5900, TCP/1433, and UDP/1434. They also found that TCP ports 22, 139, 445, and 2967 were attacked in large numbers in two of the three datasets. The researchers also studied the countries from which attacking IP addresses originated and found that the United States and China had the highest numbers, but Chinese IP addresses tended to scan a large number of computers and then concentrate attacks on a few targets. They also found some overlap between the attackers found on Leurre.com and the two honeynets they deployed (60% between the corporate honeynet and Leurre.com, 34% between the external network honeynet and Leurre.com).

The Honeynet Project [12] has contributed much to the research and development of honeypots. The Honeynet Project was founded in 1999 and is an international non-profit research organization. The Honeynet Project has members all over the world whose goal is to raise awareness of threats and vulnerabilities and to provide information so others can better secure and protect their data. The Honeynet project aims to provide information regarding attackers' motives, communication methods, timing of attacks, and actions after compromising the system. "Know Your Enemy" whitepapers are provided to detail this information [12].

In one such whitepaper [11], common threats against Web applications are detailed. Multiple high-interaction honeypots were deployed in this study. A number of trends in attacks were noted. The first trend was that attackers will attempt to gain access to the Web server's operating system through Web application vulnerabilities. They also found

that some attackers tried to inject Email spam and blog spam into the Web forms on the honeypot, which indicates that attackers use automated tools for attacks and try to inject spam and/or other vulnerability exploits regardless of the type of form. They also observed many attempts to deface the Web site by overwriting the index file with Chinese characters. Other attacks observed were attempts to recruit the honeypot to a botnet and attempts to download HTML content to the honeypot and turn it into a PayPal phishing site. On another honeypot, the researchers installed 5 Web applications (Coppermine, WebCalendar, PHPBB, AWS, and Mambo). On this honeypot, they found a number of other trends. For instance, some attackers used proxy servers, which act as an intermediate between a Web server and a Web browser and effectively hide the identity of the Web browser to the Web server, to execute their attacks. They also observed that some attackers used the Google Translate service as a proxy.

Dacier et al [10] performed an experiment in which data was collected from three virtual honeypots over a four month period. During the testing period, they observed a large number of individual attackers (6,285[1]) and almost one million packets on their honeypots. They observed the vast majority of the observed traffic was TCP (97.9%) with ICMP and UDP traffic in small percentages. Despite the large number of attackers, they observed that attacks were directed towards only 164 ports. They found that the nature of attacks appears to have changed from the previous norm of probing all ports within a range to targeting a specific port. They also found that, of their three honeypots which were running Windows 98, Windows NT, and Redhat Linux respectively, the one running the most recent version of Windows saw the most traffic. This correlates with the results from the Leurre.com project which indicated Windows as the most attractive target for attackers. Dacier saw the most traffic on the same Windows related ports as the Leurre.com project with other traffic coming primarily from TCP/80 (HTTP) and TCP/21 (FTP). The results of this study in conjunction with the results from the Leurre.com project strongly suggest that using the Windows operating system in a honeypot will attract the most malicious traffic.

Alata et al studied the behavior of attackers who encounter an SSH server with weak

[1]This is the total between all three honeypots; approximately one third of all attackers visited each honeypot.

passwords in [3]. The researchers in this study deployed a high interaction honeypot using VMWare [4] which rejected all incoming connections except those to TCP port 22 (SSH). They intentionally set weak passwords for SSH user accounts. They then studied many aspects of the behavior of attackers, such as whether the attackers were humans or automated programs, the specific activities of the attackers, and the relative skill of attackers. Their results indicated that most attacks on SSH servers were only partially automated and executed by "script kiddies", which are attackers with minimal skill, whereas attacks to other ports such as TCP/139 and TCP/445 are typically completely automated and carried out by worms. Their results also indicated that attacks on their SSH server were not preceded by scanning for an open SSH port on their honeypot.

Chen et al [36] performed a study using low-interaction honeypots developed using Honeyd [17]. They had approximately 30 honeypot platforms deployed in 20 countries at the time of their study. They first studied the country of origin of the attackers seen on their honeypots and presented the results from honeypots deployed in Taiwan and France. They found a large discrepancy in the origin of attacks on the different honeypot platforms but observed that China and the United States were among the top attacking countries in each instance. They also studied the ports targeted by attackers and found that Windows related ports (TCP/135, 137, 139, and 445) were the most popular targets, as well as ports left open by well known worms and Web-related ports. Finally, they studied the port sequences scanned by attackers on each honeypot and analyzed some of the common port sequences, such as TCP/8080, 3128, 1080, 1813, 80 and TCP/5554, 1023, 9898.

McGrew et al [49] studied data collected on two low-interaction honeypots running Solaris and Windows XP and a high-interaction honeypot running Red Hat Linux. The data obtained on the high-interaction honeypot shows that similar ports to what is seen in other related work are targeted, such as TCP/1433, 80, and 22, as well as other ports such as TCP/1025, 2745, 3127, and 6129.

Berthier et al [35] compared network traffic datasets collected using two high interaction honeypots, ATLAS [40] (a globally distributed network telescope), and Internet Protect (a network alerting service). Their study focussed on correlating events[2] received on the

[2]An event is defined as malicious activity received on a honeypot or alerts raised on ATLAS or Internet

honeypots and ATLAS or Internet Protect on the same day in order to determine local events and global events. The honeypots used for this experiment were running the Windows 2003 Server operating system. The results of this study indicate that between 27% and 65% of events monitored on the honeypots could be correlated with events found on ATLAS and Internet Protect. This indicates that there are events that are both global and local in varying degrees depending on the datasets used. The researchers were also able to identify ports that were most likely attacked globally (TCP/5900, TCP/445, TCP/22, UDP/1026, UDP/1027, and UDP/1434) and ports that were most likely attacked locally (TCP/10000, TCP/139, TCP/23, TCP/42, and TCP/2968).

Mcarty [41] describes the recruitment of a compromised Windows 2000 Server honeypot into a botnet. This honeypot was deployed at Azusa Pacific University and was installed with a null administrator password. The honeypot was probed and attacked by numerous worms, such as Slammer and Code Red II. They also observed that the following ports were probed and attacked: TCP/80, TCP/139, TCP/445, UDP/137, and UDP/1434. Some of the attacks were successful, in particular the ones on TCP/445 which allowed attackers to upload files to the honeypot. An Internet Relay Chat (IRC) client was uploaded and subsequently joined the honeypot to an IRC channel consisting of approximately 5,000 other hosts. These networks constitute *botnets* and are frequently used to execute Distributed Denial of Service (DDoS) attacks on other IRC channels and servers. The honeypot in this experiment did not participate in any attacks due to the presence of a reverse firewall which limited outgoing connections [41].

Panjwani et al [38] studied an approach to determine the correlation between port scans, vulnerability scans, and attacks. They experimented with different methods to determine a way to differentiate between these three types of connections and determined that the number of packets in the connection might be sufficient to classify it. They determined that connections with 5 packets or less can be classified as a port scan, connections with 6 to 12 packets can be classified as a vulnerability scan, and connections with more than 12 packets can be classified as an attack. However, there are some problems with their result as some port scans consist of 6 packets. They also found that only 4% of port scans led to an attack

Protect.

on their system, while 21% of vulnerability scans led to a subsequent attack. However, when both a port scan and vulnerability scan were performed by the same attacker, an attack followed 71% of the time. Their results also showed that 50% of attacks were not preceded by any sort of scan.

Pang et al studied [42] "nonproductive" or background traffic; that is, traffic that is destined for addresses which do not exist, servers that are not running, or servers which do not want traffic. The researchers built application level responders which would mimic the behavior of common services, such as Microsoft IIS, to encourage further traffic from a source. Data was collected on three different networks. They found that most of the background traffic was primarily TCP (90%) with a minority coming from the ICMP (9.8%) and UDP (0.2%) protocols. They found that the background radiation was concentrated on a small number of ports, in particular TCP ports 445, 80, 123, 1025, 2745, 139, 3127, and 6129. They also studied port sequences and found that the most common port sequences scanned combinations of the ports listed previously.

There are some similarities seen in the results of the related work. One similarity is that most of the traffic seen on deployed honeypots tend to be on the TCP protocol. There also seems to be a common set of ports and applications that are targeted in each instance. Table 2.1 shows the ports to which high amounts of traffic are seen in most of the related work. Other similarities include country of origin of attackers (China and the United States are consistently seen in the highest numbers), the attractiveness of Microsoft products for attackers, and that there is still a tendency by attackers to try to exploit weak passwords.

Port	Running Service
TCP/22	SSH
TCP/80	HTTP
TCP/137	NetBIOS Name Service
TCP/139	NetBIOS Datagram Service
TCP/445	Windows SMB File Sharing
UDP/1026	Windows Messenger RPC
TCP/1433	Microsoft SQL Server
UDP/1434	Microsoft SQL Server

Table 2.1: Ports Most Visited in Related Work

2.1.1 Our Previous Work

The work presented in this thesis is an evolution of experiments we have previously conducted. In our previous work [51], we also deployed two identically configured honeypots with one being advertised in the same manner as this study while the other was unadvertised. The honeypots in the previous work were configured as a Web server, like the ones used for this thesis, but with different applications. The honeypots used in the previous work had the following characteristics:

- Ubuntu (Linux) operating system
- Apache Web server
- MySQL database server
- phpMyAdmin Web application (used to access the MySQL databases)
- SSH server
- Honeywall to collect network traffic and restrict outgoing connections

We collected data from these honeypots for approximately 4 months from June 2, 2008 until September 28, 2008. We summarize the results of that work by the following:

- A majority of the network traffic came via the TCP protocol (over 99.9% on each honeypot). UDP and ICMP traffic were minimal.
- The advertised honeypot saw more unique source IP addresses than the unadvertised honeypot (365 to 250, respectively). A majority of this difference was attributed to more unique source IP addresses visiting HTTP ports on the advertised honeypot than the unadvertised honeypot. This indicates that much of the HTTP traffic is search-based rather than IP based, while traffic to other ports is targeted blindly at a range of IP addresses.
- On each honeypot, a few unique source IP addresses contributed an overwhelming majority of the traffic. For example, on the advertised honeypot, there were only

seven sources with more than 1,000 connections, while 334 had less than 10. Similar results were found for the unadvertised honeypot.

- The majority of unique visitors to each honeypot came from IP addresses originating from the USA, China, and Taiwan. However, the countries whose attackers sent the majority of the packets and bytes to each honeypot varied from the countries with the highest number of unique visitors because a small number of attackers launched a large number of attacks against the honeypots.

- Attackers visited 71 TCP ports on the advertised honeypot and 75 on the unadvertised honeypot, respectively. Almost all of the UDP traffic on each honeypot were Windows Messenger attacks directed at ports 1026 and 1027.

- The majority of TCP traffic was password cracking attacks on ports 22 (SSH) and 3306 (MySQL).

- The advertised honeypot saw significantly more HTTP connections than the unadvertised honeypot (522 to 78). This indicates that search-based strategies are prevalent for HTTP attacks. We also found that vulnerability scans dominated the HTTP sessions at 86.44% on the advertised honeypot and 88.24% on the unadvertised honeypot. The rest of the HTTP sessions were attacks.

- There were 19 distinct port sequences (or vertical visits) were observed between the two honeypots. The most common was a Windows Messenger attack that was prevalent at the time. Other port sequences related to HTTP ports and worm exploits.

- There were 208 unique source IP addresses which scanned the same port on both honeypots (horizontal visit). Most of these horizontal visits were to TCP ports 22 (SSH), 80 (HTTP), and 1433 (Microsoft SQL).

- The majority (90%) of unique source IP addresses that attacked each honeypot attacked on only a single day. Of those that visited on more than one day, most visited TCP port 80.

2.2 The Contributions of This Thesis

The contributions of this thesis are as follows:

- We use honeypots to offer attackers targets consisting of complex Web servers featuring two popular Web 2.0-based applications populated with real data.
- We analyze characteristics of malicious connections and compare them to previous work. We confirm some observations seen previously in that we see traffic to the ports that others saw traffic and observe many attackers originating from China and the US. We also make new observations, such as large amounts of traffic to ephemeral ports.
- We develop a method of classifying TCP connections as either port scans or vulnerability scans/attacks using a supervised-learning approach.
- We differentiate between traffic observed as a result of search-based strategies and IP-based strategies.
- We analyze port scans, vulnerability scans, and attacks through multiple tiers of a Web based system, which to our knowledge has only been done in our previous work.

Chapter 3

Experimental Setup

In this section, we present the setup for this experiment. We first describe the configuration of the honeypots, including the physical architecture, the network configuration, and the applications installed on the honeypots along with their vulnerabilities. We then discuss the tools used to perform analysis on the data collected by the honeypots.

3.1 Honeypot Configuration

For this work, we deployed two high-interaction honeypots with a real operating system and real applications. This type of honeypot was used extensively by the Honeynet project[14] and also in our previous work. The operating system we selected for the honeypots is Windows XP Service Pack 2, installed with the default options but no security updates. We chose this operating system because the related work (such as [10], [19], [35], and [41]) indicates that Windows is a very attractive target for attackers and is known to have more vulnerabilities than other operating systems. At the time of the writing of this thesis, the most current version of Windows XP is Service Pack 3 but Service Pack 2 is still widely used, so the honeypots have many known vulnerabilities while still having an operating system that has a large market share. According to Security Focus [22], this version of Windows has over 250 vulnerabilities. We created only two user accounts within Windows, Administrator and an account for log transfers. Other "system accounts", such as the IUSR account needed for the Web server, are present but could not be logged into. It should be

noted that all user accounts have strong passwords 8 to 11 characters long with at least one special character, one lower case letter, and one upper case letter. We gave the user accounts strong passwords in order to prevent a simple password cracking attempt from succeeding.

The honeypots are set up in virtual machines because virtual machines allow several honeypots, as well as data collection systems, to run on the same physical machine. This also allows us to easily redeploy a honeypot in the event it becomes compromised. Many other studies, such as [3] and [10], have used virtual machines to run honeypots. We use the VMWare Server[4] virtual machine monitor software on an Ubuntu 8.10 Server (kernel: 2.6.27-11-server) host operating system. Each honeypot is assigned its own IP address and an appropriate host name.

An important part of our experimental set up is the honeywall, which acts as a bridging, reverse firewall between the honeypots and the Internet. All traffic to or from the honeypots pass through the honeywall. The honeywall logs all network level traffic with TCPDump and then forwards the traffic to its destination without modifying the hop counts so the honeywall cannot be detected. The honeywall limits outbound connections which are initiated by each honeypot to ensure the honeypots are not used to originate attacks on other machines. The honeywall also forwards all of the logged network packets to our data collection server, which resides on a separate physical machine from the honeypots and honeywall. Figure 3.1 shows an illustration of our overall experimental setup.

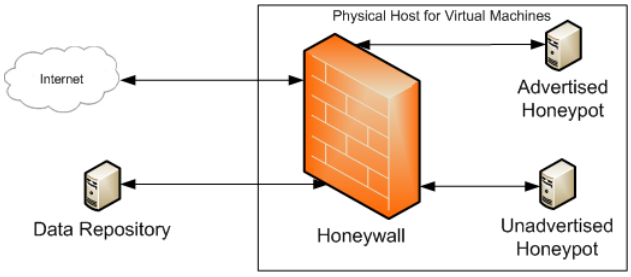

Figure 3.1: Experimental Setup

Since the goal of this research is to analyze attacks on Web applications, the honeypots are modeled as a Web server consisting of an HTTP server, application servers, and database. The configuration consists of Microsoft's Internet Information Services (IIS) Web server

version 5.1 to process HTTP requests, PHP Server version 5.0.2 to serve the PHP-based applications, and MySQL Server version 4.1 to serve as the database. According to Security Focus [22], these versions of IIS and PHP have 32 and more than 76 known vulnerabilities, respectively. Security Focus does not have vulnerability data on MySQL, however Secunia [23] has issued 23 security advisories and has recorded 26 known vulnerabilities for MySQL versions 4.x. Some of these vulnerabilities are exploitable only when an attacker is logged into MySQL, while some can also be exploited through Web applications that use MySQL or through remote login attempts.

Two Web 2.0 style applications are installed on the honeypots. The first is Wordpress (version 2.1.1) [5]. Wordpress is an PHP-based open source blogging software that is widely used across the Internet. We chose Wordpress because blogging is becoming a popular Internet activity and follows the principles of Web 2.0. According to a study by Water and Stone, Wordpress is the most downloaded open source content management system (CMS) software available online [8]. We decided to use an older version of the application in order to ensure there are a number of well-known vulnerabilities present. According to Security Focus, this particular version of Wordpress has in excess of 65 vulnerabilities, including at least 22 XSS, 2 CSRF, 10 HTML injection, and 18 SQL injection vulnerabilities. Additionally, Secunia has issued 32 advisories and reported 49 known vulnerabilities for Wordpress versions 2.x (see Figure 3.2). This presents a wide array of known vulnerabilities for attackers to exploit.

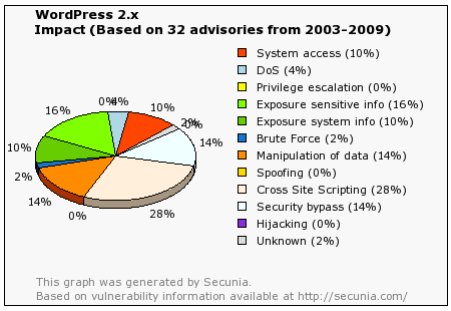

Figure 3.2: Vulnerabilities in Wordpress 2.x according to Secunia.com [23]

The second Web application we installed is MediaWiki (version 1.9.0) [6]. MediaWiki

is a PHP-based open source wiki software that is widely used across the Internet and has gained prominence as the application base for Wikipedia. We chose MediaWiki because of its standing as the dominant wiki application on the Internet. According to the study by Water and Stone referenced above, MediaWiki is the most widely used wiki software on the Internet [8]. As we did with Wordpress, we installed an older version of MediaWiki to ensure there are a number of known vulnerabilities. According to Security Focus, this version of MediaWiki has in excess of 30 vulnerabilities, including at least 18 XSS and 5 HTML injection vulnerabilities. Secunia has issued 27 advisories and recorded 29 vulnerabilities for MediaWiki versions 1.x (see Figure 3.3).

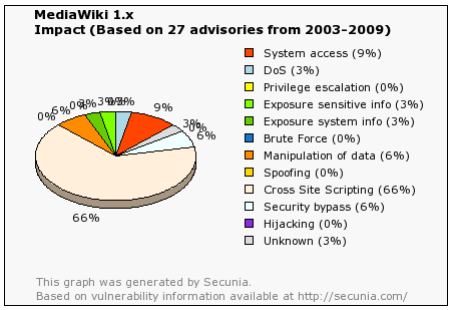

Figure 3.3: Vulnerabilities in MediaWiki 1.x according to Secunia.com [23]

We used a random text generator [24] to generate random content for each of the Web applications so it would appear to attackers that the applications were being actively used. We also created multiple user accounts in each application. In Wordpress, a total of nine user accounts were created. There are five user levels in Wordpress, each with varying degrees of usage permissions (number of user accounts created for that level shown in parentheses): administrator(1), author(3), subscriber(2), contributor(2), and editor(1). MediaWiki was configured with 5 user accounts of equal privilege level. As with the Windows user accounts, all user accounts in the Web applications have strong passwords. We configured each Web application to accept anonymous submissions (submissions from users which are not logged in). In Wordpress, anonymous users can post comments to blog entries. In MediaWiki, anonymous users have the same permission level as logged in users and can post and edit entries.

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 is to serve as the backend for the Web applications. The MySQL server contains one database for each of the Web applications as well as the system database. Seven user accounts were configured in MySQL, with varying privileges and remote login capabilities. The root account is restricted so that it can only be accessed locally or from the data collection server. Each Web application has its own MySQL account with all privileges on its own database. These user accounts are restricted to local access only so that only the Web applications may access them. There are also four other user accounts, each with remote login permissions. These accounts have varying levels of privileges: select on the application databases, select on all databases, insert and select on all databases, and select and update on all databases. The purpose of these accounts are to open the MySQL server to remote attacks. However, we want to prevent simple password cracking attempts from succeeding so all user accounts have strong passwords with the same requirements was the Windows accounts.

The "home page" of the Web server is a static HTML page which contains links to the two Web applications as well as links to other HTML pages which contain pictures. In total, there are seven picture pages each containing a different number of pictures. There are also links to two large video files (approx. 10 MB) which can be downloaded. The purpose of these pages are to provide some static HTML content with a large amount of data which we can analyze alongside the Web applications.

As mentioned in the introduction, one of the honeypots is "advertised" using a method called transparent linking. There are three links to the advertised honeypot on the Computer Science Department's Web page. Each link had different text and a different target URL, as shown in Table 3.1. We also use META tags to place appropriate keywords on each of the Web pages on our honeypots. These keywords are used by crawlers to identify the content of the Web page and index the pages for search engines.

We made some efforts to simplify the classification of crawlers as legitimate or malicious. In addition to retrieving lists of known crawlers from IPLists [26] and filtering them out of our data, we included a robots.txt file which specified that crawlers should not crawl a particular page on our honeypot. This page contains randomly generated text. Any crawlers

Text	URL Target
My Web Portal	Honeypot's Home Page
Wordpress	Honeypot's Blog
MediaWiki	Honeypot's Wiki

Table 3.1: Links to Honeypot

that crawl this page can be immediately labeled as malicious since they did not obey the robots.txt file. We also included a "favicon" file (icon which appears next the address of the Web page in most modern Web browsers). The purpose of this is to aid in identifying crawlers versus actual users. Crawlers typically will not request the favicon file, however modern Web browsers will automatically request this file. This can be used to identify traffic as coming from a crawler or a person using a Web browser.

We also installed the SSHWindows (version 3.8.1p1) [25] SSH/SFTP server on each honeypot, which is an OpenSSH server for Windows. The primary purpose of the SSH server is to enable us to automate the transfer of application logs to our data collection server via secure communication. A Windows user was created on each honeypot specifically for SSH/SFTP connections and is the only user authorized to connect to the OpenSSH server. We enabled public key encryption to allow daily scripts to be run on the data collection server without the need of keeping the password in plain text. However, we also enabled password authentication in the SSH server so that attackers can attempt to exploit any vulnerabilities. We installed the most recent version of OpenSSH rather than an older version. We did this because the primary purpose of the SSH server was data transfer and not to observe attacks. This version of OpenSSH has only 4 vulnerabilities according to Security Focus and 7 according to Secunia.

The design of our honeypots allows for attacks across multiple components of the system as well as direct attacks against certain components, much like what would be seen in a real Web server. Figure 3.4 illustrates this flow through multiple components within one of our honeypots. Attackers can launch direct attacks on the IIS server by making HTTP requests. The Web applications can only be attacked by going through the IIS server, but attackers can subsequently attack the MySQL database by going through the Web applications. Attackers can also attack the MySQL database by connecting directly to TCP port 3306. The operating

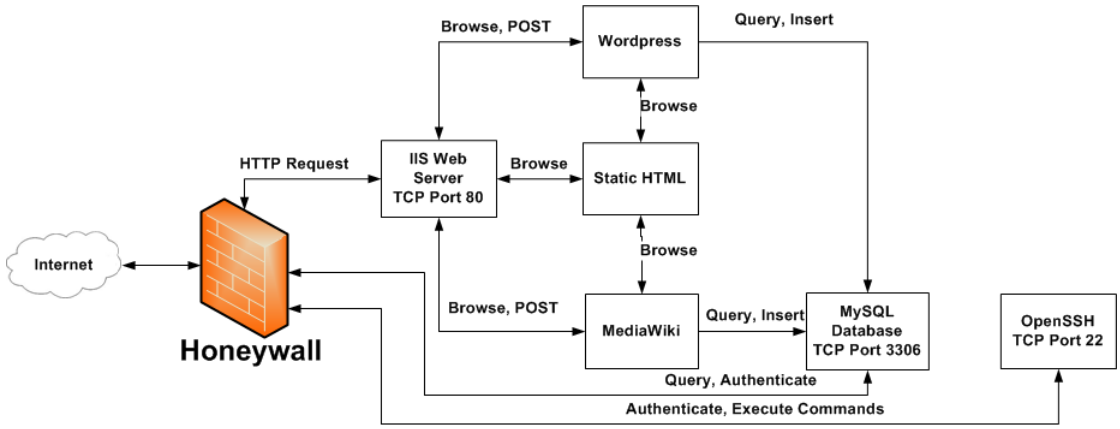

Figure 3.4: Flows Through Multiple Components on Honeypots

system can be attacked by going through IIS, MySQL, SSH, or through direct connections to ports on which Windows services run.

3.2 Analysis Tools

In order to analyze the activity of attackers, we look at the data contained in the various logs created on the honeypots. The honeywall logs all network packets sent to and from the honeypots. Windows maintains extensive system logging referred to as event logs. We installed the Eventlog to Syslog Utility [9] on the honeypots, as described in [44]. This utility automatically forwards all events generated by Windows to a remote syslog server located on the data collection server. The default logging capabilities of Wordpress and MediaWiki are minimal, so we enhanced their logging capabilities by modifying the source code of the applications. The source code was modified so that the username and password used for any login attempt are recorded and stored in a log file. These log files are copied nightly to the data collection server. All HTTP requests are logged by IIS and transferred nightly from the honeypots to the collection server. All log transfers from the honeypot to the data collection server are done via the SCP protocol to ensure secure transmission.

Analyzing these logs manually is a very time consuming task. To simplify the analysis, we custom-developed numerous scripts. We created multiple scripts to extract useful data from the network traffic logs. These scripts were originally developed for use in our previous

work and then updated and enhanced to aid in the analysis for this thesis. The network traffic related scripts we developed have the ability to:

- Combine many network traffic capture logs (pcap files) into a single file
- Filter legitimate crawlers (found on iplists.com [26]) and system traffic (such as syslogs) from the network traffic
- Compute all connections and relevant information about each connection
- Compute the amount of traffic to each port
- Compute horizontal and vertical scans
- Compute port sequences
- Compute attack sources

The output from these scripts are comma separated value files which contain the compiled data. These scripts are written in Java and optimized to provide the best performance. The scripts are written without any hard-coded IP addresses or host names so they can be used in the future without modification to the code. The analysis performed by these scripts can be performed in approximately 30 seconds of computational time and 5 - 10 minutes of real time. Most of this analysis can also be performed with a combination of the Wireshark network analysis tool [28] and Microsoft Excel. However, the analysis that can be performed in 30 seconds with our custom developed scripts take in excess of two hours to complete using the other tools.

We note here that we analyze only the incoming network traffic[1]. The outgoing traffic from our honeypots consists of direct responses to incoming traffic from attackers, so we therefore limited our analysis to only the incoming traffic.

From previous work [29][30][31], we already had custom developed tools to extract data from Web server (Apache) logs. However, the IIS logs generated by the honeypots in this experiment were not in the format that the tool from previous work required. We therefore developed a script to transform the IIS logs into the necessary format for the custom

[1]We define incoming traffic as a packet whose destination address is the address of one of our honeypots

developed tool. This tool categorizes each HTTP request into an HTTP session, which is defined as multiple HTTP requests from the same source IP address within a threshold of 30 minutes. The output of this tool is a comma separated value file which can then be imported into a MySQL database. Once the Web logs are imported into a database, SQL queries are used to extract information such as the number of fingerprinting requests and sessions, number of login attempts, etc.

We collected other logs on our honeypots, such as syslogs, MySQL logs, and logs from the Web applications, but did not develop any tools to aide in analyzing these logs. Syslogs typically contain authentication attempts, program errors, and service restarts as recorded by the operating system. We are primarily concerned with authentication attempts and used the "grep" function in Linux to extract authentication attempts from the syslogs. We then inspected them manually for the information we needed. The MySQL logs contain all authentication attempts and queries to the database. However, since the network level traffic indicated there were only 9 connections to MySQL's TCP port and the HTTP logs indicated that there were no SQL injection attacks, we felt that developing a tool to extract information from the MySQL logs was unnecessary and we simply manually parsed the MySQL log to get the information we needed from it. Finally, the authentication logs for the Web applications consisted of only 2 login attempts, so we also deemed it unnecessary to develop a tool for analyzing the Web application logs.

Chapter 4

Data Analysis

In this section, we present an analysis of the data collected by our honeypots. First, we analyze the network traffic, including the country of origin of the attackers (the country to which the IP address of an attacker resolves) and port sequences visited by attackers. Then, we will analyze the transport protocols (TCP, UDP, ICMP) and and application level traffic specifically.

Tables 4.1 and 4.2 give a summary of the incoming network traffic that was collected from March 30, 2009 to April 26, 2009 on the advertised and unadvertised honeypots, respectively. We first observe that an overwhelming majority of the traffic sent to the honeypots was sent using the TCP protocol. Similar results were also seen in our previous work and related works such as [10] and [42]. We next observe that the advertised honeypot has many more unique IP sources visiting TCP ports than the unadvertised honeypot. This difference can be attributed to traffic on TCP port 80 (HTTP). The advertised honeypot saw HTTP traffic from 95 unique IP addresses while the unadvertised honeypot saw HTTP traffic from only 21 unique IP source addresses. While a significant number of attackers still found their way to the unadvertised honeypot, this indicates that, for HTTP traffic, a majority of the unique IP sources use search-based methods rather than IP-based methods to find their targets. We also observed similar results in our previous work. Finally, we observe that the unadvertised honeypot has a significantly larger amount of UDP traffic. This can be mostly be attributed to the traffic generated by a single attacker. There was no incoming ICMP traffic to either honeypot.

Protocol	Unique IP Sources	Ports Visited	Connections	Packets	% Packets
TCP	279	41	7,244	61,437	99.12%
UDP	88	10	NA	548	0.88%
Total	367	51	NA	61,985	100%

Table 4.1: Summary of traffic on the advertised honeypot

Protocol	Unique IP Sources	Ports Visited	Connections	Packets	% Packets
TCP	219	43	7,362	43,148	96.04%
UDP	92	13	NA	1,777	3.96%
Total	311	56	NA	44,925	100%

Table 4.2: Summary of traffic on the unadvertised honeypot

4.1 Countries of Origin

In this section, we discuss the country of origin of source IP addresses that produced traffic on our honeypots. Tables 4.3 and 4.4 present a summary of this information for each honeypot. The country of origin of a particular IP address is computed using information from the IP-to-Country-Database [33]. It should be noted that the country of origin of an attacker is the country to which his or her IP address belongs. The attacker could be in one country and using a machine in another to launch attacks or could be spoofing his or her IP address.

Visitors to the advertised honeypot have IP addresses that originate from 50 different countries. As can be seen in Table 4.3, the majority of visitors are from the United States (31.01%) and China (18.16%) and the majority of the connections come from the United States (56.34%), Canada (16.49%), and Italy (13.17%). There is a noticeable difference in which countries contribute the most unique IP source addresses and which countries generate the most connections. There a couple of reasons for this difference. It was observed in [43] that Chinese IP addresses tend to scan many computers and focus their attacks on only a few. This could explain why many Chinese IP addresses visited the advertised honeypot but did not generate as high a percentage of the overall connections. We also note that attackers from Canada contribute the 2nd largest amount of connections but less than 2% of the unique attackers. This is because almost all of the connections from Canadian IP addresses

Country	% Unique IP sources	% Connections	Port with most connections
United States	31.006%	56.337%	TCP/445
China	18.156%	5.393%	TCP/22
Russia	4.190%	0.339%	TCP/28472
France	3.911%	1.264%	UDP/21616
Italy	3.352%	13.169%	TCP/80
Taiwan	3.352%	0.261%	TCP/25
Brazil	3.073%	0.638%	TCP/22
North Korea	2.793%	0.547%	TCP/22
Ukraine	2.514%	0.274%	TCP/80
Canada	1.955%	16.491%	TCP/22
Germany	1.955%	0.469%	TCP/22
India	1.955%	0.261%	TCP/80
Spain	1.676%	0.208%	TCP/80
Japan	1.366%	0.404%	UDP/21616
Philippines	1.366%	0.326%	UDP/21616
Vietnam	1.366%	0.195%	TCP/80
Argentina	1.366%	0.130%	TCP/28472
Thailand	1.366%	0.112%	TCP/80
Other (32 Countries)	15.027%	3.304%	NA

Table 4.3: Country of origin for attackers to the advertised honeypot

were generated by a single IP address which attempted to crack the password on our SSH server. This is a similar phenomenon to what we observed in our previous work where a large amount of the connections came from Romania but were attributed to a single attacker who was attempting to break the password on the MySQL server. A similar situation exists with the Italian IP addresses, except the attacker who contributes most of the traffic from Italy was searching for phpMyAdmin on the honeypot. We also observe that the port that was targeted most by the highest number of countries (6) was TCP port 80 (HTTP). Other ports on the advertised honeypot that were targeted by attackers from more than one country are TCP/22 (SSH), UDP/21669 (unassigned), and TCP/28472 (unassigned).

On the unadvertised honeypot, we observed visitors whose IP addresses originate from 50 different countries as well. As can be seen in Table 4.4, the majority of unique IP sources are again from China (40.51%) and the United States (10.29%) while the majority of connections come from the United States (48.437%), China (21.52%), and Canada (14.03%).

Country	% Unique IP sources	% Connections	Port with most connections
China	40.514%	21.515%	UDP/45342
United States	10.289%	48.437%	TCP/445
Russia	5.466%	0.342%	TCP/4586
Brazil	3.859%	0.525%	TCP/22
Italy	2.894%	11.395%	TCP/80
Taiwan	2.572%	0.171%	TCP/25
India	2.572%	0.148%	TCP/4586
Ukraine	2.251%	0.354%	TCP/80
Malaysia	2.251%	0.160%	TCP/4586
North Korea	1.929%	0.354%	TCP/22
Germany	1.929%	0.319%	TCP/22
Spain	1.929%	0.148%	TCP/4586 and TCP/34318
United Kingdom	1.929%	0.137%	TCP/4586
Canada	0.643%	14.031%	TCP/22
Other(38 Countries)	18.971%	2.053%	NA

Table 4.4: Country of origin for attackers to the unadvertised honeypot

This is similar to the advertised honeypot except that China has a much higher proportion of the unique sources as well as the number of connections. This is due to a drastically larger amount of UDP traffic to the unadvertised honeypot which originated from China. The ports targeted most by the highest number of countries (5) was TCP/4586 (Unassigned). A significant observation is that the HTTP port is targeted most by fewer countries on the unadvertised honeypot (1) than compared to the advertised honeypot (6), confirming earlier observations that much of the HTTP traffic comes from search-based strategies rather than IP-based strategies.

The country of origin analysis on the data we collected yields somewhat similar results to those found in other studies, such as [19], [36], and our previous work, in that IP addresses from China and the United States are significant contributors to malicious traffic. We also observe that there seems to be a pattern in the behavior of attackers from certain countries, such as the behavior of Chinese IP addresses to be selective in the computers they attack observed in [43] which we also observed, as well as a tendency of attackers from the United States to generate more attacks per unique IP source. However, as we also observed in our previous work, the activities of single attackers (such as the Canadian IP address who execute

password cracking attacks or the Italian IP address which searched for phpMyAdmin) can skew the results for any individual country in a particular study.

4.2 Characteristics of Malicious Traffic

In this section we analyze some general characteristics of the malicious traffic, regardless of protocol. We first analyze attackers who target multiple ports and/or multiple honeypots and then study attackers who visit the honeypots on multiple days.

4.2.1 Port Sequences

In this section we study patterns in the port sequences visited by attackers. We observed traffic that visited multiple ports on both honeypots. We call this a *vertical visit* (single source IP address targets multiple ports on the same target machine). We observed 15 vertical visits to the advertised honeypot and 22 to the unadvertised honeypot. These vertical visits target 10 unique port sequences on the advertised honeypot and 12 unique port sequences on the unadvertised honeypot. There are 11 unique source IP addresses which executed vertical visits on the advertised honeypot and 14 on the unadvertised honeypot, 10 of which executed the exact same vertical visit on each honeypot at least one time. This indicates that many of the vertical visits are part of a systematic exploration of a sequential range of IP addresses looking for specific services or specific vulnerabilities in those services.

The port sequences we observed in vertical visits are shown in Table 4.5. The most common sequence is TCP/8000, 7212. There is no apparent vulnerability attempting to be exploited by the traffic in this port sequence. TCP port 8000 is listed in some places, such as Wireshark, as the Intel Remote Desktop Management Interface. However, port 8000 is also commonly used as an alternate HTTP port, particularly for proxy servers. TCP port 7212 has no assigned usage but has been used by Ghostsurf [37], a software package that allows for anonymous, encrypted Internet surfing, according to dsheild.org. This suggests that attackers executing vertical scans using this port sequence are looking for running instances of proxy servers, perhaps Ghostsurf specifically. Since our honeypots do not have any services running on those ports, they simply reply with a RST (reset) packet and terminate the

	Port Sequence	Advertised honeypot	Unadvertised honeypot
1	TCP/8000, TCP/7212	5	4
2	TCP/445, TCP/44445	2	5
3	TCP/80, TCP/8080	2	2
4	TCP/445, TCP/139, TCP/80	1	3
5	TCP/5554, TCP/1023, TCP/9898	1	1
6	TCP/1986, TCP/8080, TCP/5946, TCP/780	1	1
7	TCP/10000, TCP/20000,	1	1
8	TCP/8090, TCP/8089, TCP/808, TCP/8088	1	0
9	UDP/3345, UDP/21616	1	0
10	TCP/1364, TCP/28472	1	0
11	UDP/54520, UDP/16451	0	2
12	TCP/24314, TCP/4586	0	1
13	TCP/8089, TCP/8090, TCP/8088	0	1
14	TCP/8083, TCP/8081	0	1
15	UDP/34252, UDP/16451	0	1
	Total	15	22

Table 4.5: Port sequences observed in vertical visits

connection.

The port sequence we saw in the second highest number was TCP/445, 44445. TCP/445 is a Windows port used for network file sharing, and we believe that TCP/44445 is an alternate port for this service. Another similar port sequence is TCP/445, 139, 80. TCP/139 is another port used for network file sharing on Windows and this sequence (sequence 4) was used by an attacker to search for open network shares on the honeypots.

We observed other port sequences in smaller numbers, some of which we can explain and some we cannot. One such sequence is TCP 80, 8080. This sequence is fairly common and is used by attackers to search for running Web servers. We saw this port sequence in our previous work as well. Another port sequence that we observed is TCP 5554, 1023, 9898. This sequence is testing for backdoor vulnerabilities left open by well-known worms [36] which we also saw in our previous work. Other port sequences, such as 9, 10, 11, 12, and 15 are to ports that are unassigned. We believe these scans were looking for peer-to-peer applications which use ephemeral ports, as described in [47], or were background noise picked up by our honeypots. These port sequences we observed have not been identified because

they are not going to ports with any known services or vulnerabilities.

We also analyzed *horizontal visits* (single source IP address targets the same port on multiple machines). Table 4.6 shows the horizontal visits that we observed. The horizontal visits are spread across 20 port sequences with 22 (SSH), 445 (Microsoft-DS file sharing), 80 (HTTP) and 1433 (MS SQL) seeing the most horizontal visits. We also observed that most attackers who execute horizontal visits execute only a single horizontal visit to a particular port (only 16 sources executed more than 1 horizontal visit). One reason for this could be that most of the ports visited are not open on our honeypots and attackers did not return after not finding what they were looking for. However, the top three are open, which is why we suspect there were more repeat visitors to those two ports than any of the other ports.

Port	Number of horizontal visits	Number of unique IP sources	Number of IP sources with only one vertical visit
TCP/22	25	21	17
TCP/445	25	1	0
TCP/80	20	15	4
TCP/1433	14	14	14
TCP/25	5	5	5
TCP/53	3	3	3
TCP/7212, 8000	2	1	0
UDP/123	1	1	1
TCP/21	1	1	1
TCP/780	1	1	1
TCP/1023	1	1	1
TCP/3050	1	1	1
TCP/3306	1	1	1
UDP/5060	1	1	1
TCP/5554	1	1	1
TCP/5946	1	1	1
TCP/8080	1	1	1
TCP/8083	1	1	1
TCP/8088	1	1	1
TCP/20000	1	1	1
Total	106	73	56

Table 4.6: Number of horizontal visits across both honeypots per port

4.2.2 Attack Sources

We also study whether attackers visited the honeypots once and then went away or returned multiple times using a statistic called *attack source* (a unique tuple {date; source IP address}). Each day that a source IP address visits a honeypot counts as an attack source, regardless of how much traffic that IP address generates to the honeypot. We are interested in seeing if sources revisit the honeypots on different days. Figures 4.1 and 4.2 show the number of attack sources per unique source IP address on the advertised and unadvertised honeypot. On each honeypot, we observe that the majority (more than 90%) of attackers visit the honeypot on only a single day. We believe these attackers were either unsuccessful in attacking the honeypots or simply did not find the service or vulnerability they were looking for and never returned. Almost all of the attackers who visited on more than one day were visiting port 80 which suggests they were malicious crawlers probing the Internet and our honeypots for potential targets.

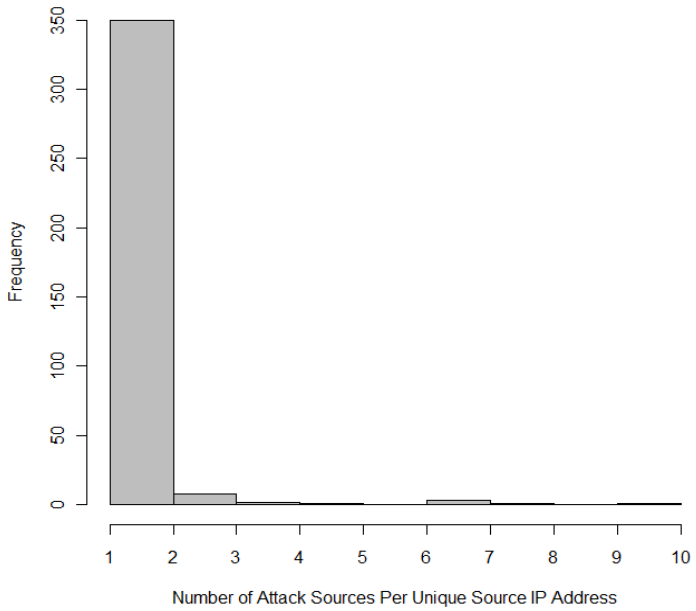

Figure 4.1: Histogram of Attack Sources Per Unique Source IP Address on Advertised Honeypot

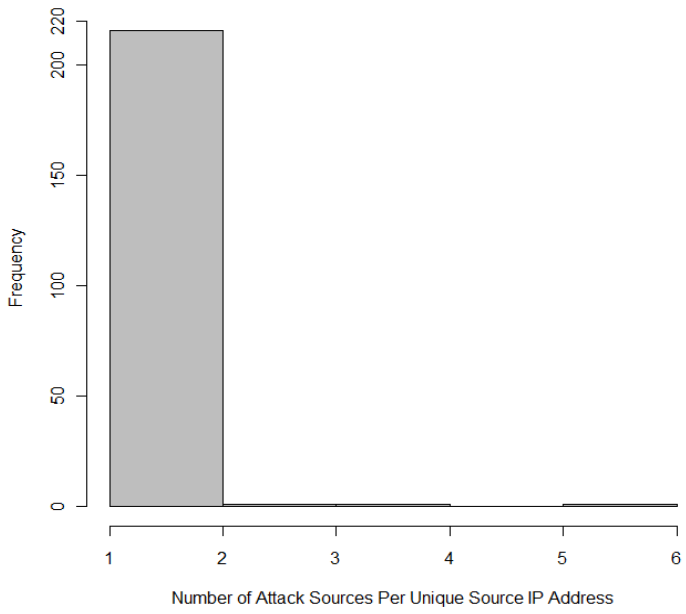

Figure 4.2: Histogram of Attack Sources Per Unique Source IP Address on Unadvertised Honeypot

4.3 Analysis of TCP Traffic

In this section we present an in-depth analysis of the TCP traffic. We first study the number of packets and connections from unique attackers and the traffic distribution across TCP ports. We then present a supervised-learning approach to classifying connections as port scans or vulnerability scans/attacks. Finally, we present a detailed analysis of the traffic to the applications and application level protocols that run on our honeypots: HTTP, SMB, SSH, and MySQL.

4.3.1 TCP Connections and Packets per Unique Source IP Address

We are interested in seeing how the TCP traffic is distributed amongst the unique sources that visited TCP ports on the honeypots. As shown in Tables 4.1 and 4.2, there were 279 unique IP sources who sent at least one packet to TCP ports on the advertised honeypot and 219 on the unadvertised honeypot. There were 108 sources who visited both honeypots.

Figure 4.3 shows a histogram of the number of connections per unique source IP address on the advertised honeypot. We can see that a majority of the sources connected to the

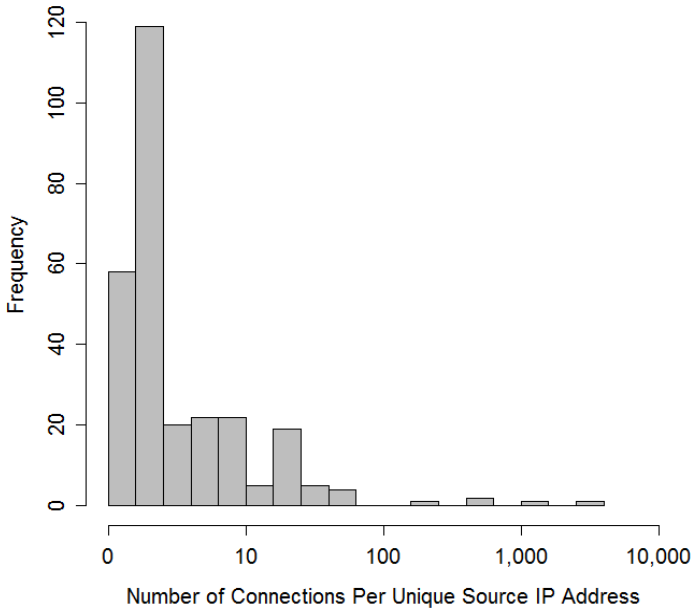

Figure 4.3: Number of TCP Connections Per Unique Source IP Address on the Advertised Honeypot

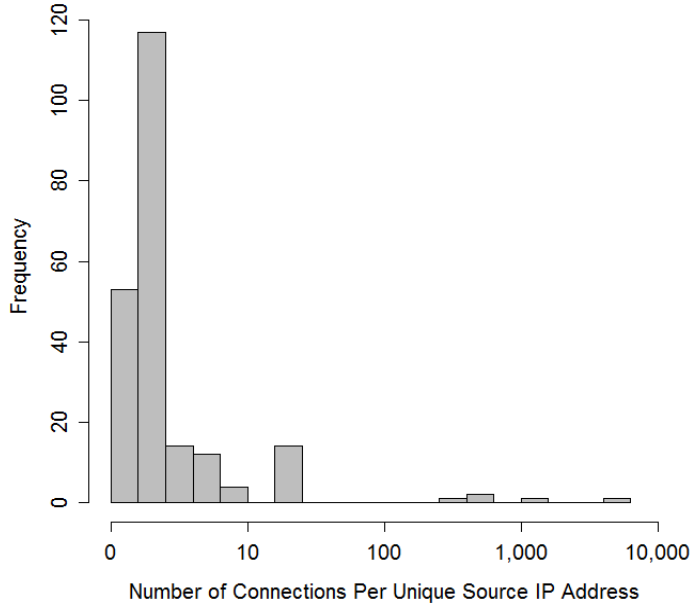

Figure 4.4: Number of TCP Connections Per Unique Source IP Address on the Unadvertised Honeypot

honeypot less than 10 times, while only a very few connected more than 100 times. Of the 279 unique IP sources who visited the advertised honeypot, only 5 connected more than 100 times while 58 connected only once and 237 connected less than 10 times. This indicates that a few source IP addresses constitute a majority of the connections to the honeypot.

We saw similar behavior on the unadvertised honeypot. Figure 4.4 shows a histogram of the number of connections per source IP address on the unadvertised honeypot. Of the 219 unique IP sources who visited the unadvertised honeypot, only 5 connected more than 100 times while 53 only connected once and 199 connected less than 10 times. We also saw this behavior in our previous work.

The packets sent per source IP exhibit similar behavior to the connections per source IP. Figures 4.5 and 4.6 show a histogram of the number of packets per source IP address on each honeypot. We can see that a majority of sources sent less than 1,000 packets to the honeypots while only a very few sent more than 1,000. This indicates that a few source IP addresses send a majority of the packets to the honeypot.

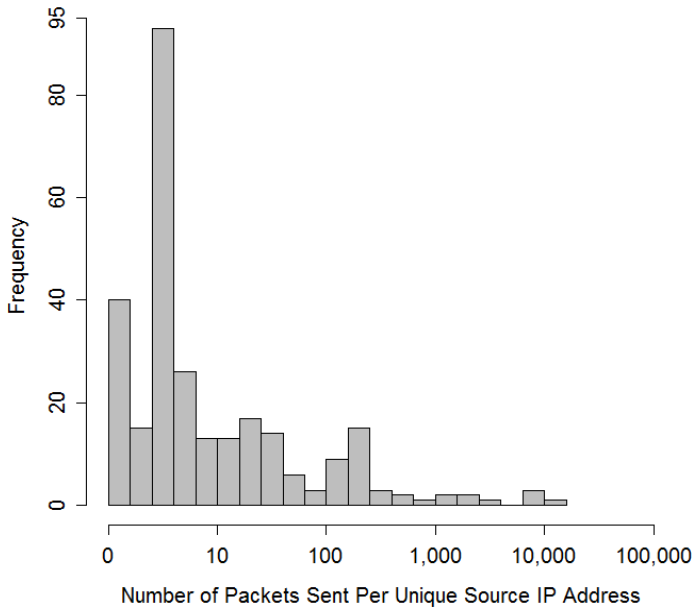

Figure 4.5: Number of TCP Packets Per Unique Source IP Address on the Advertised Honeypot

4.3.2 Traffic Across TCP Ports

Now, we discuss the distribution of TCP traffic across the different TCP ports. Attackers connected to 41 TCP ports on both the advertised honeypot and 43 unadvertised honeypot. There were 36 TCP ports that were visited on both honeypots. Table 4.7 summarizes the distribution of TCP traffic to TCP ports. We make the following observations about the

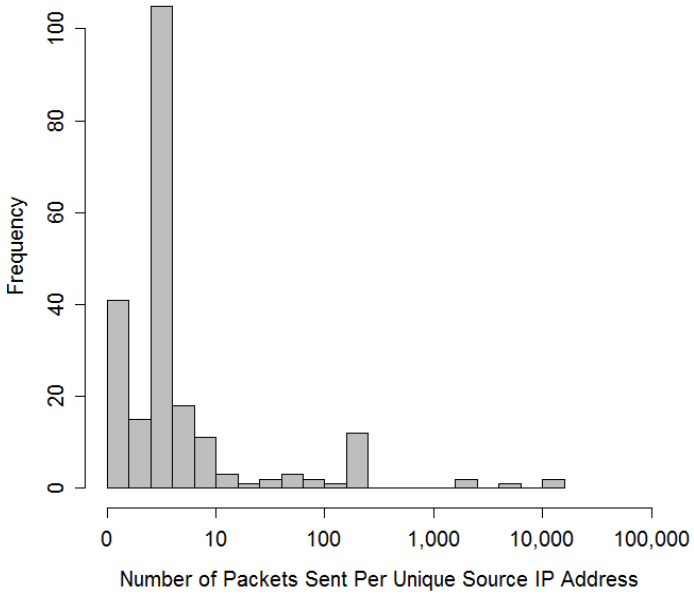

Figure 4.6: Number of TCP Packets Per Unique Source IP Address on the Unadvertised Honeypot

TCP traffic:

- More connections were made to TCP port 445 than any other port on each honeypot, while this port contributed to a significant yet smaller percentage of the total packets received on each honeypot. TCP port 445 is designated for the Server Message Block protocol (SMB) and is used primarily for file sharing on Windows machines. TCP ports 137 and 139 can also be used for this purpose.

- HTTP traffic to port 80 is the second most popular target port for TCP traffic. It is interesting to note that there are 40 more HTTP connections on the unadvertised honeypot than the advertised honeypot. However, there are over 20,000 more packets sent to port 80 on the advertised honeypot than the unadvertised honeypot. This is because there was more diversity in the HTTP requests on the advertised honeypot (attackers browsing/crawling the site) while most HTTP requests on the unadvertised honeypot were targeted at fingerprinting a specific vulnerability. This will be discussed further in Section 4.3.4

- SSH traffic was seen in the third highest number on each honeypot. Over 90% of the SSH traffic to each honeypot was password cracking attempts. We identified 73 port

Port	Advertised honeypot				Unadvertised honeypot			
	Connections		Packets		Connections		Packets	
SMB (445)	2,698	37.24%	4,985	8.11%	2,612	35.49%	6,718	15.57%
HTTP (80)	2,369	32.70%	34,776	46.21%	2,409	31.78%	13,712	24.88%
SSH (22)	1,812	25.28%	21,062	34.28%	1,882	25.57%	21,942	50.86%
Unknown (28472)	83	1.16%	134	0.22%	0	NA	0	NA
Unknown (4586)	0	NA	0	NA	118	1.60%	176	0.41%
NetBIOS (139)	58	0.80%	120	0.20%	109	1.48%	218	0.51%
MS SQL (1433)	42	0.58%	53	0.08%	42	0.58%	53	0.12%
Unknown (41653)	0	NA	0	NA	28	0.38%	45	0.10%
Unknown (56087)	27	0.37%	40	0.07%	0	NA	0	NA
Unknown (34318)	0	NA	0	NA	26	0.35%	36	0.08%
Unknown (64125)	24	0.33%	36	0.06%	0	NA	0	NA
Unknown (29229)	24	0.33%	35	0.06%	0	NA	0	NA
Unknown (56380)	0	NA	0	NA	18	0.24%	25	0.06%
Unknown (21669)	0	NA	0	NA	17	0.23%	26	0.06%
Unknown (41347)	12	0.16%	16	0.03%	0	NA	0	NA
HTTP-ALT (8080)	11	0.15%	16	0.03%	11	0.15%	16	0.04%
SMTP (25)	10	0.14%	31	0.05%	10	0.14%	31	0.07%
Ghostsurf (7212)	10	0.13%	17	0.03%	10	0.13%	17	0.04%
MySQL (3306)	9	0.12%	37	0.06%	9	0.12%	37	0.09%
HTTP-ALT (8000)	10	0.13%	17	0.03%	9	0.12%	17	0.04%
Other	26	0.62%	62	0.10%	52	0.71%	79	0.11%
Total	7,244	100%	61,437	100%	7,362	100%	43,148	100%

Table 4.7: Statistics about TCP ports visited on each honeypot

scans to port 22 on the advertised honeypot and 69 to the unadvertised honeypot. Port scans were identified using the Perceptron based classification system we developed for this thesis.

- MS SQL traffic was seen on each honeypot, even though we did not have Microsoft SQL running on either honeypot. The nature of the MS SQL traffic was an attempt to establish a connection which was immediately replied to with a RST packet by our honeypots since we have no service running on that port. Traffic to the MS SQL port was also seen in our previous work and other related work such as [43] and [35].

- On each honeypot, we observe significant amounts of traffic to ports that do not seem to be assigned to anything (such as 4586, 28472, 41653, etc.). While we cannot defini-

tively explain what this traffic is, we suspect it is attackers searching for peer-to-peer applications to connect to. Peer to peer applications commonly use ephemeral ports [47], so we believe this traffic is attackers attempting to find running peer-to-peer applications. We also note that the traffic to these ports is always directed at a single honeypot, not both, which suggests that these attacks are not attackers searching the same port on a range of IP addresses. The intent of the attackers generating malicious traffic to these ports is not entirely clear. However, the traffic to these ports seem to be simple port scans consisting of SYN packets. Since our honeypots do not have any services running on these ports, a RST packet is immediately sent in response and the connection terminated.

4.3.3 Port Scans versus Vulnerability Scans/Attacks

We now present a method of classifying TCP connections as either (1) a port scan or (2) a vulnerability scan/attack. Panjwani et al [38] used the number of packets in a connection to distinguish between port scans, vulnerability scans, and attacks. The metric they used was 5 packets or less represents a port scan (except for a special case in which there could be 6 packets in a port scan), 6 - 12 packets represents a vulnerability scan, and 13 or more packets represents an attack. We believe that the number of packets in a connection is a useful metric but are not convinced that these threshold values will work well for the data being analyzed in this thesis. We analyze only incoming traffic while Panjwani analyzed incoming and outgoing traffic. Therefore, we decided to design our own system to classify TCP connections.

We developed a supervised-learning based pattern classification system to classify TCP connections as either a port scan or a vulnerability scan/attack. The tool used for classification is written in the R programming language [45] and uses a batch Perceptron algorithm to compute a linear boundary between the two classes (port scan, vulnerability scan/attack). The basis of the Perceptron algorithm is a weight vector which is used to classify data. The weight vector is multiplied by the vector representing a data point and the point is classified in class one if the result is positive and in class two otherwise. The batch Perceptron

algorithm works by adjusting this weight vector incrementally until the error rate for the classifier is less than a designated threshold. The presence of a threshold allows the batch Perceptron algorithm to converge even if the classes are not linearly separable[1]. This contrasts the fixed increment single sample Perceptron algorithm which will only converge if the classes are linearly separable. The Perceptron algorithm is discussed further in [46]. We considered other machine learning approaches to solving this problem, such as those used in [47], but we decided to use the Perceptron because of its simplicity and effectiveness.

Two features are used to classify the connections: (1) the number of packets in the connection and (2) the duration of the connection. As discussed earlier, the number of packets in a connection has been used for this purpose previously, and we also believe that the duration of the connection is a reasonable statistic to use because we observed in our data that vulnerability scans and attacks in particular tend to have longer durations. We experimented with using the bytes transferred in a session as another feature but we felt that it did not contribute anything to the accuracy of the classifier, so we decided not to use that feature. Distinguishing between vulnerability scans and attacks proved to be a much more difficult problem to solve and we therefore decided to restrict ourselves to classifying port scans versus vulnerability scans and attacks.

In order to obtain the proper weight vector, the Perceptron algorithm requires training data where each data point is manually classified. A test set of data is also needed to evaluate the performance of the classifier. Therefore, we manually classified approximately 250 TCP connections as either a port scan or a vulnerability scan/attack. These manually classified connections were then randomly partitioned into a training set (70% of data points) and a testing set (30% of training points). We decided to use 70% of the manually classified connections as the training set because this percentage proved to give the best accuracy without using 90% or more of the data. Figure 4.7 shows the accuracy of the Perceptron in classifying the test set as a function of the size of the training data.

The weight vector found by the Perceptron algorithm using the training data was

[1]Linearly separable means that a linear boundary can be used to separate the data points in each class

(3,870.667, -34.33, -1,288.00), giving a linear decision boundary of

$$y = -0.0267x - 3.0052.$$

We used this weight vector to classify all TCP connections and evaluated its accuracy using the test set. The weight vector accurately classified 98.667% of the test data set. This weight vector misclassified only 1 port scan and misclassified no vulnerability scans/attacks. We feel that an error rate of under 2% demonstrates very acceptable performance. We also tested the classifier on the HTTP traffic, where we can classify connections as port scans by looking at connections that appear in the network traffic logs but not in the IIS logs, and found that our system correctly classified all HTTP port scans.

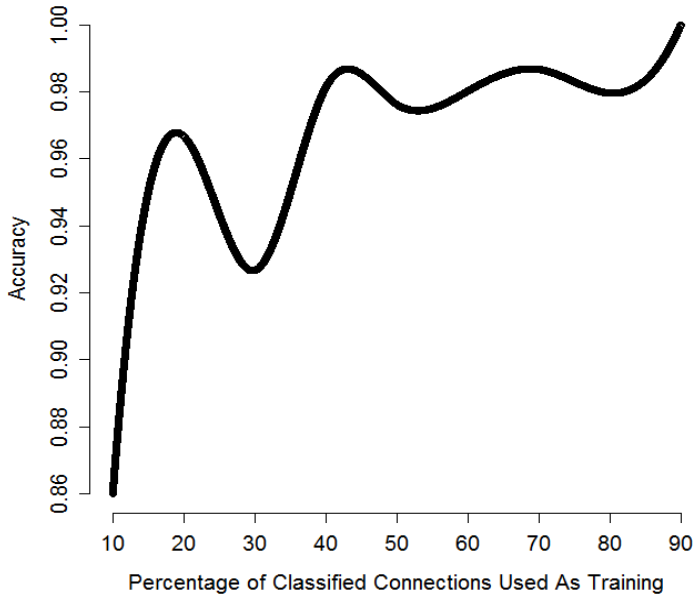

Figure 4.7: Accuracy of Perceptron Classifier As a Function of Training Set Size

Table 4.8 shows the number of port scans and vulnerability scans/attacks found on each honeypot by the classifier. We found that 3,276, or 45.22%, of TCP connections to the advertised honeypot were port scans while 2,960, or 40.21% of TCP connections to the unadvertised honeypot were port scans (with the rest being either a vulnerability scan or attack). These results are similar and the difference in the percentage of port scans between the two honeypots can be attributed to the actions of attackers who only visited a single honeypot. Overall, we can see that slightly less than half of the TCP traffic is simply

port scans looking for running services while slightly over half of the TCP connections are attackers actively searching for a vulnerability or actually attacking the honeypot.

	Port Scans	Vulnerability Scans/Attacks
Advertised Honeypot	3,276	3,968
Unadvertised Honeypot	2,960	4,402

Table 4.8: Number of Port Scans and Vulnerability Scans/Attacks Observed on Each Honeypot

Figure 4.8 illustrates the classification of the TCP connections on the advertised honeypot. This figure shows the decision boundary found by the Perceptron algorithm as the black line, the connections classified as port scans as red circles, and the connections classified as a vulnerability scan/attack as blue triangles. We also show the training set as filled in data points. It is important to note that many connections have the same vector (duration, packets sent), so there is some overlap in the figures. Figure 4.9 shows the same information for the unadvertised honeypot.

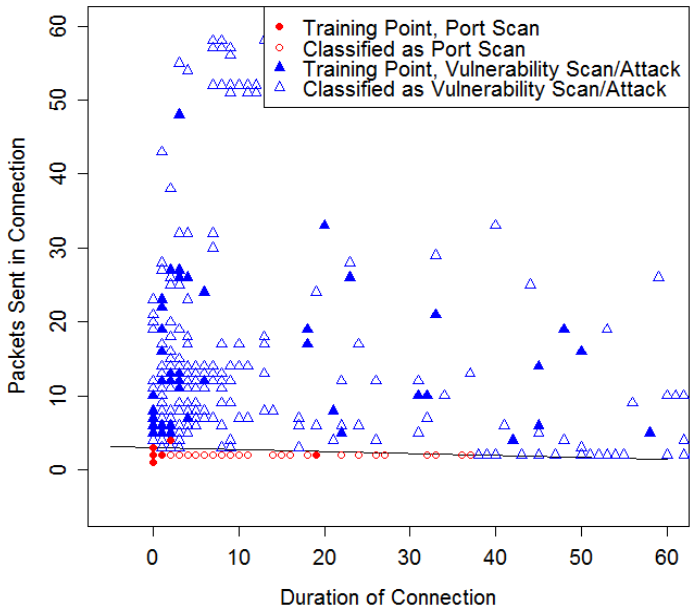

Figure 4.8: Classification of Port Scans and Vulnerability Scans/Attacks Using Perceptron on Advertised Honeypot

We observe from the results that the decision boundary is a line that is close to horizontal,

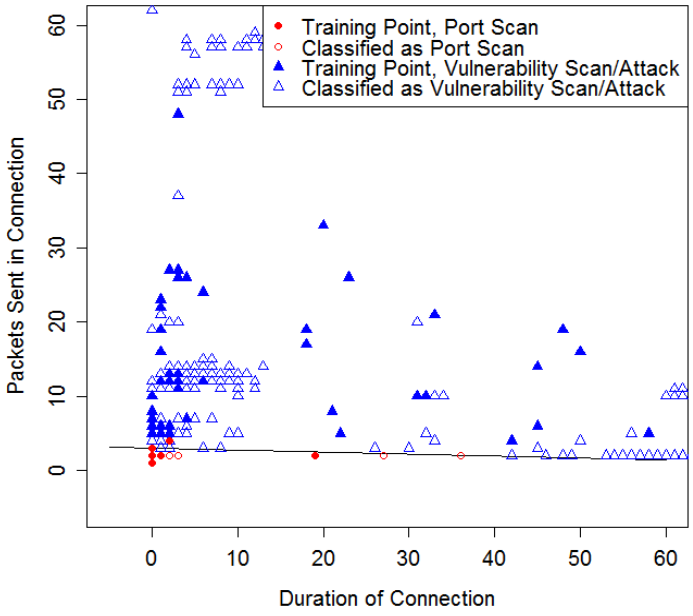

Figure 4.9: Classification of Port Scans and Vulnerability Scans/Attacks Using Perceptron on Unadvertised Honeypot

indicated that the number of packets per connection is the dominate feature in the classifier and the duration of the connection is of little consequence. This indeed requires more research and experimentation, and thus the work performed on this classifier is considered preliminary and, in future work, we plan to expand the use of machine-learning techniques to classify connections. We would like to increase size of the training and testing sets for use with the Perceptron, develop a method of distinguishing between vulnerability scans and attacks, and evaluate the use of clustering techniques to identify categories of connections.

4.3.4 HTTP Traffic

We now look specifically at the HTTP traffic, which includes both the TCP packets sent to port 80 and the requests logged by the IIS Web server running on the honeypot. The first statistic we look at is *port scans*, which are used simply to determine if a Web server is running on our honeypots. We determined which connections are port scans by looking for connections that appear in the network traffic logs but do not appear in the IIS logs (the Perceptron based classifier described in Section 4.3.3 classified all of these connections as port scans as well). Port scans represents a very small percentage of the HTTP connections.

In total, there were 14 unique source IP addresses that executed HTTP port scans on the advertised honeypot and 9 that executed port scans to the unadvertised honeypot. Five of the attackers who performed HTTP port scans to the unadvertised honeypot also performed them on the advertised honeypot.

We found that there were 19 port scans to the advertised honeypot and 14 to the unadvertised honeypot. Table 4.9 shows the activities of attackers after they performed a port scan to TCP port 80. We see that all but 6 of the attackers who performed HTTP port scans on the honeypots returned later to do something else. An interesting observation here is that none of the attackers which returned to the unadvertised honeypot fingerprinted IIS[2] or crawled the site. This is because fingerprinting and crawling are consistent with search-based strategies, which are not possible on the unadvertised honeypot since there is no visible link to it. However, IP-based traffic, such as searching for a particular service (phpMyAdmin) or vulnerability (DFind), was observed on both honeypots. This indicates that even private Web servers, which are not "visible" by a link to the general Internet, are still the target of malicious traffic searching for particular services and vulnerabilities.

Activity Performed After Port Scan	Sessions on Advertised Honeypot	Sessions on Unadvertised Honeypot
Searched for phpMyAdmin	4	5
Fingerprinted IIS	3	0
Crawled Web site	5	0
Other Vulnerability Scan	3	3
No return visit	6	6
Total	19	14

Table 4.9: Subsequent actions of attackers who performed port scan to TCP port 80

We note that, in order to differentiate between traffic generated from search-based strategies as opposed to IP-based strategies, we compared the traffic seen on each honeypot. Traffic going to the advertised honeypot only is considered search-based because the traffic is not directed at a range of IP addresses (or it would have reached the unadvertised honeypot as well) and must have been generated by the link to the Web page or by a search engine's

[2]Fingerprinting IIS means making an HTTP request that will cause the Web server to reply with information about the Web server, including the application name and version

index. Traffic seen on both honeypots or the unadvertised honeypot only is considered IP-based since the only way to reach the unadvertised honeypot is by directly connecting to the IP address.

We now analyze application level data collected through the IIS logs[3]. We analyzed both request level and session level traffic. An HTTP session is defined as a sequence of requests from the same source IP addresses with a maximum time of 30 minutes between any two requests. Table 4.10 gives an overview of the application level traffic we observed on each honeypot. Requests from legal crawlers (requests that originated from an IP address listed by IPlists [26] as that of a legal crawler or robot) are filtered out of the results. *Vulnerability scans* are requests designed to find vulnerabilities on the Web server and Web applications and *attacks* are requests that are direct attacks on some part of our system.

	Advertised honeypot				Unadvertised honeypot			
	Sessions		Requests		Session		Requests	
Vulnerability scans: Total	195	98.98%	2,191	98.96%	19	95%	2,381	99.96%
DFind	6	3.04%	6	0.27%	6	30%	6	0.25%
XMLRPC.PHP	6	3.04%	23	1.04%	0	0%	0	0%
Toata Scanner	2	1.02%	3	0.14%	2	10%	3	0.13%
Fingerprinting								
phpMyAdmin	3	1.52%	985	44.49%	3	15%	1,002	42.07%
PROPFIND	1	0.51%	661	29.86%	3	15%	1,364	57.26%
IIS Fingerprint	34	17.26%	45	2.03%	4	%	5	0.21%
IIS Fingerprint&Browse	25	12.69%	233	10.52%	0	0%	0	0%
Browse Site Only	118	59.90%	235	10.61%	1	5%	1	0.04%
Attacks: Total	**2**	**1.02%**	**23**	**1.04%**	**1**	**5%**	**1**	**0.04%**
Password Cracking on Blog	1	0.51%	22	0.99%	0	%	0	0%
CVE-2008-5619	1	0.51%	1	0.05%	1	5%	1	0.04%
Total	**197**		**2,214**		**20**		**2,382**	

Table 4.10: Breakdown of HTTP application level traffic

We make the following observations about the data in Table 4.10:

- The unadvertised honeypot received more HTTP requests than the advertised honeypot but substantially less HTTP sessions. The reason the unadvertised honeypot saw more HTTP requests is due to a few attackers who were searching for particular vul-

[3]Port scans are not part of the IIS logs

nerabilities searching more on the unadvertised honeypot than the advertised honeypot for unknown reason. However, the fact that the advertised honeypot saw substantially more HTTP sessions than the unadvertised honeypot (182 to 20) confirms our earlier observations that HTTP traffic tends to come from search-based strategies rather than IP-based strategies.

- The advertised honeypot and unadvertised honeypot saw very similar vulnerability scans (except for IIS fingerprinting and Browse/Crawl types). These vulnerability scans were looking for specific services (such as phpMyAdmin) or specific vulnerabilities (such as DFind, XMLRPC, soapCaller.bs, RoundCube), many of which were not present on our honeypots. This indicates that attackers searching for a specific service or vulnerability to exploit employ IP-based schemes to find their targets.

- The advertised honeypot saw much more IIS fingerprinting and regular visits (browsing the pages of our Web-based applications) than the unadvertised honeypot. In fact, almost none of the traffic to the unadvertised honeypot involved browsing the Web pages; it was all directed towards vulnerability scans and fingerprinting phpMyAdmin.

Vulnerability scans constituted the majority of the HTTP traffic to each of our honeypots. The vulnerability scans can be classified into the following categories:

- *Dfind* is a tool which allows attackers to scan for vulnerabilities on computers running Microsoft Windows operating systems. Dfind can be used to search for certain running services or vulnerabilities [39]. Dfind scans are characterized by HTTP requests like "GET /w00tw00t.at.ISC.SANS.DFind:)". All IP addresses that scanned the advertised honeypot using Dfind also scanned the unadvertised honeypot. However, none were successful in finding a vulnerability to exploit.

- *XMLRPC.PHP* is a set of remote procedure calls which allows software (usually Web-based software) to make calls over the Internet [50]. XML-RPC is used in many PHP based Web applications, including Wordpress. XML-RPC has a number of vulnerabilities but did not generate any attacks despite requests for that file on our honeypots being responded to with the "200" status code (OK). Four attackers fingerprinted

XMLRPC.PHP on each honeypot a total of 6 times. We believe these attackers were looking for a specific version of XMLRPC, did not find it, and therefore did not attempt any further attacks.

- The *Toata Scanner* (complete name: Toata+dragostea+mea+pentru+diavola) is another tool used to look for vulnerabilities in Web applications. In the case of our honeypots, the scanner made requests on the page "/roundcube/bin/msgimport", which belongs to the RoundCube Web mail application, which is not installed on the honeypots, or to "moodle/README.TXT", which indicates a scanner searching for an installation of Moodle, which is an open source Web application which is not installed on our honeypots.

- The final category of vulnerability scans is *fingerprinting*. The most prevalent fingerprinting performed on our honeypots was attackers searching for the phpMyAdmin application, which is not installed. We observed three sessions in which attackers searched up to 500 different paths in an attempt to find phpMyAdmin on each honeypot. Examples of the paths searched are "/phpMyAdmin/main.php", "/admin/phpMyAdmin/main.php", "/websql/phpMyAdmin/main.php", etc. We also observed a number of scans of the form "GET / HTTP/1.*" These types of requests are typically used by attackers to get information on the Web server installed on the system. However, in the case of our honeypots, the index.html page was returned to the attacker. The rest of the traffic designated as "fingerprinting" consisted of crawlers and/or humans browsing the site.

We observed many "PROPFIND" requests from a single attacker. This attacker had an internal IP address and is the same attacker who generated all of the traffic to port 445 on both of our honeypots. PROPFIND is a method of Web-based Distributed Authoring and Versioning (WebDAV) and is basically a set of extensions to HTTP which allow the management of files on remote Web servers. The version of the IIS Webserver installed on our honeypots has a single PROPFIND related vulnerability (CAN-2003-0226) that causes denial of service when excessively long requests are made using PROPFIND [22]. However, we do not believe that this vulnerability was being exploited in this case. The PROPFIND

requests to our honeypots were requests for "/ADMIN$", "/c$", "/d$", "/d$", "/f$", "/g$", and "/h$" which is quite similar to the requests this attacker made using the SMB protocol. We believe these PROPFIND requests were another attempt by that attacker to locate available shared drives and folders on our honeypot.

Table 4.11 shows a breakdown of the activity of attackers who browsed the Web pages located on the honeypots, including a breakdown of sessions generated by automated scripts (crawlers) and humans. We classify a session as being automated when it does not request the favicon file and as being non-automated when it does. We can see that there was very little actual browsing on the unadvertised honeypot and only one attacker actually viewed a Web page on the unadvertised honeypot. However, on the advertised honeypot, we saw a lot of traffic across all parts of the Web site located on the honeypot. Most attackers (18) visited both the blog and the wiki, while a smaller number visited only the blog (11), only the wiki (12), or only the picture pages (10) and 3 attackers visited all 3. We also observed that most attackers who requested the robots.txt file, which is used to tell robots which parts of the site they may crawl and may not crawl, proceeded to crawl other pages on the Web site. Only 5 of the attackers who requested the robots.txt file did not browse further on the Web site. None of the crawlers which requested the robots.txt file proceeded to crawl the disallowed page.

	Advertised honeypot		Unadvertised honeypot	
Session Type	Automated	Non-Automated	Automated	Non-Automated
1 Blog Visit Only	36	4	1	0
2 Wiki Visit Only	39	2	0	0
3 Picture/Video Visit	26	0	0	0
4 Blog and Wiki Visits	31	2	0	0
5 Blog, Wiki, and Picture/Video Visit	4	1	0	0
6 Robots.txt Only	2	0	0	0
7 Robots.txt and Other Pages	68	0	2	0

Table 4.11: Summary of "Browse/Crawl Site" Sessions from Table 4.10

We can also see the overwhelming majority of sessions on each honeypot are crawlers,

with the only non-automated sessions coming on the advertised honeypot. This indicates that a vast majority of activity on HTTP ports is completely automated rather than executed by humans using Web browsers.

The rest of the traffic we observed is labeled as an *attack*:

- The first attack we observed was a password cracking attack on the advertised honeypot. This attack was a login (password cracking) attempt to the Wordpress blog. The attacker tried logging in with two username/password combinations, then tried the "Forgot password?" link[4] before giving up.

- *CVE-2008-5619* was executed by the Morfeus scanner, which is a common tool used to look for vulnerabilities in PHP based applications, to each honeypot. The scans to our honeypots were looking for "/user/soapCaller.bs", which is associated with the RoundCube Web mail application. CVE-2008-5619 is a remote code inclusion vulnerability. Since RoundCube is not installed on our honeypots, this attack had no affect.

We can draw a number of conclusions from the HTTP traffic:

- Overall, HTTP traffic is dominated by search-based strategies, as we see many more HTTP sessions on the advertised honeypot versus the unadvertised honeypot.

- Attackers searching for a specific service or vulnerability are more likely to use IP-based strategies than search-based strategies.

- The traffic on the advertised honeypot and unadvertised honeypot is significantly different. The traffic on the advertised honeypot is more diverse and consists of vulnerability scans, fingerprinting for specific applications and services, and browsing the site. Traffic to the unadvertised honeypot is almost entirely comprised of vulnerability scans and fingerprinting.

- Attackers who perform a port scan on port 80 are likely to come back and probe the Web server located at that IP address if one is found.

[4]The Forgot Password link generally allows users to have their password emailed to them. However, we did not configure the applications on our honeypots for email.

4.3.5 SMB (Windows File Sharing) Traffic

We now will discuss the traffic to TCP port 445 (SMB). This analysis utilizes the network traffic logs and the syslogs. All of the SMB traffic to our honeypots came from a single, internal IP address. This IP address connected to our honeypots on ports 80, 139, and 445 on eight separate days. On port 445, this IP address attempted to authenticate with the advertised honeypot 74 times and the unadvertised honeypot 176 times with user names such as "Administrator", "admin", "Guest", and "sa". This attacker also tried to find shared folders and connect to them by browsing to paths such "C″, "*D*", "E″, "*F*", "ADMIN″, "*IPC*", etc (there were no shared files or folders on the honeypot, so the attacker was searching in vain). The attacker also attempted numerous times to locate the "eraseme" virus on the honeypots by searching for files with the name "eraseme_%.exe" where % represents a number. None of the attempted intrusions to our honeypot were successful. We tried, unsuccessfully, to locate the computer from which these attacks were being launched. We believe there is a possibility that machine is itself compromised and being used to launch attacks on our network from within the network. We therefore notified the IT Staff in the department so that they could investigate further. The internal IP address which launched all connections to TCP port 445 on our honeypots is also the same attacker who launched a series of "PROPFIND" requests to our IIS servers.

4.3.6 SSH and MySQL Traffic

The analysis of malicious SSH and MySQL traffic is based upon the network traffic logs, syslogs, and MySQL application log. The SSH port was the third most visited on each honeypot. As mentioned earlier, there were 73 port scans to the SSH port on the advertised honeypot and 69 to the unadvertised honeypot. Port scans were executed by all 24 unique IP sources to visit port 22 on the advertised honeypot and all 23 unique IP sources who visited port 22 on the unadvertised honeypot. The rest of the connections, over 90%, were password cracking attempts. The pattern observed in the password cracking attempts follows the pattern we observed in our previous work: each connection consists of an attempt to log in with a single user name and password combination. There were

1,739 password cracking attempts on the advertised honeypot and 1,813 on the unadvertised honeypot. We identified password cracking attempts by checking the syslogs (which contain the SSH authentication logs) and the network traffic (SSH password cracking attempts follow a predictable pattern, as we found in our previous work). Password cracking attacks on the advertised honeypot were executed by 17 unique source IP addresses, all of which preceded their password cracking attacks with port scans to port 22 while password cracking attacks on the unadvertised honeypot were executed by 18 unique source IP addresses, all of which also preceded their password cracking attempts with port scans to port 22. This contrasts with the results of Alata's study [3] where it was found that no password cracking attacks on SSH were preceded by a port scan.

We did not observe much malicious traffic to the MySQL server. There were 9 connections to port 3306 observed on each honeypot, all initiated by the same attacker. The connections consisted of 2 port scans followed by 7 password cracking attacks (attempts to log in as root), which failed. The MySQL logs in conjunction with the network traffic were used to identify the password cracking attempts.

4.4 UDP and ICMP Traffic

	Advertised Honeypot		Unadvertised Honeypot	
Port	Packets	% Packets	Packets	% Packets
Unassigned (45342)	0	NA	1,298	73.17%
Unassigned (21616)	505	92.15%	0	NA
Unassigned (16451)	0	NA	434	24.46%
Unassigned (13681)	19	3.47%	0	NA
Unassigned (15833)	0	NA	14	0.79%
Unassigned (56377)	13	2.37%	0	NA
Unassigned (54520)	0	NA	10	0.56%
NTP (123)	4	0.73%	6	0.34%
SIP (5060)	1	0.18%	1	0.06%
Other	6	1.09%	14	0.79%
Total	548	100%	1,777	100%

Table 4.12: Statistics about UDP ports visited on each honeypot

We observed little UDP and ICMP traffic to the honeypots. Only 0.88% of the incoming traffic to the advertised honeypot and 3.96% of the incoming traffic to the unadvertised honeypot was on the UDP protocol. We observed more UDP traffic to the unadvertised honeypot because of a single attacker who sent more than 1,200 packets to UDP port 45342 on the unadvertised honeypot but not the advertised honeypot. We observed no incoming ICMP packets to the honeypots. However, the advertised honeypot sent 540 ICMP "Destination unreachable (port unreachable)" packets in response to UDP packets received by the honeypot. The unadvertised honeypot sent 1,771 such packets. Overall, the honeypots replied to 99% of the UDP packets with ICMP destination unreachable packets. Table 4.12 shows the distribution of UDP packets to UDP ports.

Over 99% of the UDP traffic was directed towards ports which are not assigned to any service and do not have any obvious vulnerabilities. Most of these packets (more than 90%) have payloads of exactly 33 bytes. Also, the payloads are all very similar, differing in only a few bytes, and end with the string "LIME". While we could not find any specific information on UDP packets of this type, we hypothesize that this could be some sort of vulnerability scan or attack on the LimeWire peer-to-peer application.

Chapter 5

Conclusion

In this thesis, we have presented an analysis of the activity of malicious users on typically configured Web servers based on data collected on two high interaction honeypots. The results have shown the malicious traffic is largely dependant on the services running on the deployed machines.

We first analyzed the characteristics of the attackers who visited our honeypots. We found that most of the traffic to our honeypots originated from IP addresses residing in China and the United States and we also noted that attackers from different countries tend to have specific patterns of traffic. We also studied the types of visits performed by attackers and identified the port sequences seen in vertical visits and horizontal visits. We found that there was little correlation between the port sequences seen in vertical visits and the services running on our honeypots. However, a correlation does exist in the horizontal visits since the top 3 ports visited in horizontal visits are ports on which our honeypots have running services (TCP/22, 80, 445).

TCP traffic dominated the traffic to our honeypots. Over 96% of all traffic was to TCP ports, with UDP comprising the rest. The observed UDP traffic went to unassigned ports and we could not definitively classify it.

Our analysis of the TCP traffic indicates that the majority of the TCP traffic is generated by only a few unique sources. We observed that the ports to which most TCP traffic was directed were ports on which we had running services, such as HTTP, SSH, and Windows file sharing (SMB). However, we also noticed that a significant amount of traffic was directed

towards "ephemeral" ports, which are ports to which there are no assigned services. We suspect this traffic was port scans to find running peer-to-peer applications.

We then presented a supervised-learning based pattern classification system (based on the batch Perceptron algorithm) which classifies TCP connections as either port scans or vulnerability scans/attacks. We found that between 40% and 45% of the TCP connections to the honeypots were port scans and the rest were vulnerability scans or attacks.

Next, we performed a detailed analysis of the HTTP traffic. This analysis indicated that the majority of HTTP traffic is search-based rather than IP-based but that attackers who look for specific vulnerabilities tent to use IP-based strategies. We also observed that, despite having popular Web 2.0-based applications on our honeypots, we saw very little attempts to exploit specific Web 2.0 vulnerabilities. We suspect that more time is needed for data collection in order to see these types of attacks.

We observed a significant amount of traffic to the other applications and protocols running on our honeypots. Traffic to the SMB Windows file sharing port dominated the number of connections to each honeypot. This traffic was generated by a single internal IP address searching for open network shares. We also observed many port scans and password cracking attempts on our SSH servers and a minimal amount on our MySQL servers. We observed that all password cracking attempts on the SSH and MySQL servers were preceded by port scans.

We believe that having honeypots which run real services and applications is essential to attracting attackers. It is also important that these services have some sort of meaningful purpose when aggregated on the same system, as was the case with our honeypots that were modeled after Web servers.

The methods used to analyze the data collected by the honeypots utilized for this thesis can be easily reapplied to further implementations. The main value of this thesis lies not only in the results of the analysis but in the tools and methods developed to analyze the data. The experience gained through the work in this thesis will be extremely valuable in further iterations of this line of research.

5.1 Future Work

For future work, we would like to expand the data collection time period in hopes of observing Web 2.0 related attacks and other unique activity. We also would like to expand our research in the area of using machine-learning techniques to classify network and application level traffic. The Perceptron based classifier we presented in this thesis represents just the preliminary research into this type of classification. We would like to improve the Perceptron classifier and experiment with other approaches, such as clustering, to classify our data.

References

[1] S. Murugesan, "Understanding Web 2.0", *IT Professional, Volume 9, Issue 4*, 2007, pp. 34 - 41

[2] G. Lawton, "Web 2.0 Creates Security Challenges", *IEEE Computer, Volume 40, Issue 10*, 2007. pp. 13 - 16

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

[4] VMWare Inc, http://www.vmware.com/

[5] Wordpress, http://www.wordpress.org/

[6] Mediawiki, http://www.mediawiki.org/

[7] MySQL, http://www.mysql.com

[8] R. Shreves, *The 2008 Open Source CMS Market Share Report.* Water & Stone http://www.waterandstone.com, 2008 https://engineering.purdue.edu/ECN/Resources/Documents/UNIX/evtsys

[9] Eventlog to Syslog Utility,

[10] M. Dacier, F. Pouget, H. Debar, "Honeypots: Practical Means to Validate Malicious Fault Assumptions", *Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing*, 2004, pp. 383 - 388

[11] J. Riden, R. McGeehan, B. Engert, M. Mueter, "Know Your Enemy: Web Application Threats", *The Honeynet Project & Research Allicance, http://www.honeynet.org*, 2007

[12] The Honeynet Project, http://www.honeynet.org/

[13] PHPShell, http://sourceforge.net/projects/phpshell/

[14] The Honeynet Project, "Know Your Enemy: GenII Honeynets", *The Honeynet Project & Research Alliance, http://www.honeynet.org*, 2005

[15] Leurrecom.org Honeypot Project, http://www.leurrecom.org/

[16] C. Leita, V. H. Pham, O. Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, M, Dacier, "The Leurre.com Project: Collecting Internet Threats Information using a Worldwide Distributed Honeynet", *WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, IEEE. 2008, pp. 40 - 57

[17] Honeyd, http://www.honeyd.org/

[18] C. Leita, K. Mermoud, and M. Dacier. "Scriptgen: an automated script generation tool for honeyd", *In Proceedings of the 21st Annual Computer Security Applications Conference*, 2005.

[19] F. Pouget, M. Dacier, V. H. Pham, "Leurre.com: on the Advantages of Deploying a Large Scale Distributed Honeypot Platform", *E-Crime and Computer Conference*, 2005.

[20] Adobe Flex, http://www.adobe.com/products/flex/

[21] Google Web Toolkit, http://code.google.com/webtoolkit/

[22] Security Focus, http://www.securityfocus.net/

[23] Secunia, http://www.secunia.com/

[24] Random text generator, http://www.randomtextgenerator.com/

[25] SSHWindows, http://sshwindows.sourceforge.net/

[26] IP Addresses of Search Engine Spiders, http://www.iplists.com/

[27] D. Illiyev, K.H. Choi, K.J. Kim, "Dangers of Applying Web 2.0 Technologies in E-commerce Solutions", *2008 International Conference on Information Science and Security*, IEEE. 2008, pp. 17-25.

[28] Wireshark, http://www.wireshark.org/

[29] K. Goseva-Popstojanova, S. Mazimdar and A. D. Singh, "Empirical Study of Session-based Workload and Reliability for Web Servers", *15th IEEE International Symposium on Software Reliability Engineering*, Nov. 2004, pp. 403-414.

[30] K. Goseva-Popstojanova, A. Singh, S. Mazimdar and F. Li, "Empirical Characterization of Session-based Workload and Reliability for Web Servers", *Empirical Software Engineering Journal*, Vol.11, No.1, Jan. 2006, pp. 71-117.

[31] K. Goseva-Popstojanova, F. Li, X. Wang and A. Sangle, "A Contribution Towards Solving the Web Workload Puzzle", *36th Annual IEEE/IFIP International Conference on Dependable Systems & Networks (DSN 2006)*, 2006, pp. 505-514.

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

[33] IP-to-Country-Database, http://ip-to-country.webhosting.info/

[34] V. Yegneswaran, P. Barford and J. Ullrich, "Internet Intrusions: Global Characteristics and Prevalence", *2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2003, pp. 138-147.

[35] R. Berthier, D. Kormann, M. Cukier, M. Hiltunen, G. Vesdonder and D. Sheleheda, "On the Comparison of Network Attack Datasets: An Empirical Analysis", *11th IEEE High Assurance Systems Engineering Symposium (HASE 2008)*, Dec. 2008, pp. 39-48.

[36] P. T. Chen, C. S. Laih, F. Pouget and M. Dacier, "Comparative Survey of Local Honeypot Sensors to Assist Network Forensics", *1st International Workshop on Systematic Approaches to Digital Forensic Engineering*, 2005, pp. 120-132.

[37] Ghostsurf, http://www.tenebril.com/consumer/ghostsurf/

[38] S. Panjwani, S. Tan, K. M. Jarrin, M. Cukier, "An Experimental Evaluation to Determine if Port Scans are Precursors to an Attack", *2005 International Conference on Dependable Systems and Networks*, 2005, pp. 602 - 611.

[39] Symantec, http://www.symantec.com/

[40] Active Threat Level Analysis Sytem (ATLAS), http://atlas.arbor.net/

[41] B. McCarty, "Botnets: Big and Bigger", *IEEE Security and Privacy, July/August 2003*, pp. 87 - 90.

[42] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, L. Peterson, "Characteristics of Internet Background Radiation", *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, 2004, pp. 27 - 40.

[43] R. Bloomfield, I. Gashi, A. Povyakalo, V. Stankovic, "Comparison of Empirical Data from Two Honeynets and a Distributed Honeypot Network", *19th International Symposium on Software Reliability Engineering*, 2008, pp.219 - 228.

[44] The Honeynet Project, *Know Your Enemy: Learning About Security Threats, 2nd Edition*, Addison-Wesley, 2004.

[45] The R-Project, http://www.r-project.org/

[46] C. Bishop, *Pattern Recognition and Machine Learning*, New York: Springer Science+Business Media LLC, 2006, pp. 192 - 196.

[47] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, "Offline/realtime Traffic Classification Using Semi-supervised Learning", *Performance Evaluation*, 2007, pp. 1194 - 1213.

[48] U. Erlingsson, B. Livshits, Y. Xie, "End-to-end Web Application Security", *Microsoft Research.*

[49] R. McGrew and R. B. Vaughn, "Experiences with Honeypot Systems: Development, Deployment, and Analysis", *39th Annual Hawaii International Conference on System Sciences (HICSS 06)*, 2006, pp. 220a-220a.

[50] XML-RPC, http://www.xml-rpc.com/

[51] K. Goseva-Popstojanova, B. Miller, R. Pantev, A. Dimitrijevik, J. Lynch, "Empirical Analysis of Attackers' Activity on Multi-Tier Web Systems", *28th IEEE Symposium on Reliable Distrubuted Systems*, 2009, Submitted to be published.

[52] SANS Top-20 2007 Security Risks (2007 Annual Update), http://www.sans.org/top20/

[53] http://www.gartner.com/