An Experimental Study of SSH Attacks by using Honeypot Decoys

Esmaeil Kheirkhah[1], Sayyed Mehdi Poustchi Amin[2*], Hediyeh AmirJahanshahi Sistani[2] and Haridas Acharya[3]

[1]Department of Computer Engineering, Mashhad Branch, Islamic Azad University, Iran; E.kheirkhah@gmail.com
[2]Department of Computer Studies and Research, Symbiosis International University, Pune, India;
Poustchi@outlook.com; Hediehamirjahanshahi@yahoo.com
[3]Allana Institute of Management Science, Pune University, Pune, India; Haridas.undr@gmail.com

Abstract

We studied Brute-force SSH attacks carried out on six different universities campus networks by using Honeypot Techniques. Brute-force password guessing attacks against SSH, FTP and telnet servers are the most common form of attack to compromise servers facing the internet. A key factor to avoid disruption of these networks is to defend it against Brute-force attacks. We focused on the attempts to gain remote access to our SSH Honeypots Plus Tools and techniques employed.

There are striking similarities in the methods used to attack these dissimilar systems. The evidence shows that, pre-compiled lists of usernames and passwords that are widely shared form the basis for brute-force attacks. When the passwords were analysed, it was found that in the event of actual malicious traffic what was commonly understood to be strong password did not protect the systems from being compromised. The data from the study were used to evaluate the efficacy of a variety of techniques designed to defend the systems against these attacks. Table 17 lists some commonly recommendation for the protection of SSH servers.

Keywords: Honeypot, Honeynet, Internet Attacks, Network Security, SSH, Brute-force, Malware.

1. Introduction

The researchers in the nascent fields of digital and network forensics constantly require new tools and techniques to stay abreast of the latest trends in attack on the systems. The use of honeypots has become more extensive in the computer security industry because the attack vectors have specially shifted into new domains. To quote the leader of the Honeynet Project, Lance Spitzner, a Honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource [1]. A honeypot is a decoy computer system, designed to look like a legitimate system, which tempts an intruder to break into it while unknown to the intruder, he is being covertly observed. The attackers are unaware of the exact existence of the Honeypots and hence the honeypots become precisely effective [2].

RFC 4251 defines Secure Shell (SSH) as "a protocol for secure remote login and other secure network services over an insecure network." [3] Nowadays, majority of administrators use SSH as a secure replacement for Telnet and r commands like rsh and rcp. SSH supplies similar functionality while providing encrypted communications, password-less logins via public key authentication, and host-based verification.

**Corresponding author:*
Sayyed Mehdi Poustchi Amin (Poustchi@outlook.com)

In recent years, several studies on SSH attack have been undertaken [4–6]. In some cases, the study of SSH attack has been a part of a larger study. The larger study included observation of intruder activities following the compromise of a system or brute-force attacks. This research has narrowly focused on the malicious login traffic, with the aim of developing a deeper understanding of the tools and techniques employed by such brute-force SSH intruders. They still, by many accounts, represent a significant threat to networked Linux systems [7].

Attacks against SSH are typically attempts to gain remote access to a system or to cause a denial of service condition since SSH supplies mechanisms for remote access or remote file transfer. Basically, SSH service listen for connection attempts on TCP port 22. After the three-way handshake completes setting up the TCP connection between the client and server, both client and server transpose SSH version information and cryptographic encryption keys. The remote user can seek authentication after the encrypted channel is established. If this authentication fails, the connection will be closed. The user is granted remote access to the system, if the authentication succeeds.

There are two classes of Honeypots. The first class depends upon the use of honeypots; in other words the purpose of the honeypots and its production is purely for research. The second class of honeypot is built depending upon the level of interactivity that they provide to the attackers; in other words the can be termed low or high interaction honeypots. By low-interaction honeypots, one refers to those that simulate only some parts of the system (the network stack).

The low-interactions honeypots do not implement real services as much as the high-interaction honeypots. They are utilized for collection of information at a higher level (to learn about network probes and worm activity). Hence, low-interaction honeypots simulate some services that prevent an attacker full access to the honeypots. Thus the attacker can't control the system. With the aid of patched system calls from loadable kernel modules, a high interaction honeypot is able to capture a large amount of data, including encrypted communications.

We have set up a low-interaction honeypot system in six different networks (different Internet IP spaces) to investigate malicious activities that occur on these networks. These systems are low-interaction honeypots based on Kojoney project [8]. Here an attacker can interact with honeypot as in any other system on the network. For the attacker, there appears to be no discernible difference between the honeypot and other computer systems. It must be borne in mind however, that the honeypot is closely monitored through the IPTables firewall. The system events are recorded on the honeypot itself via its logging facility. In fact, we improve the Kojoney honeypot to allow the attacker to interact with our simulated shell environment. Add support for XMPP (Extensible Messaging and Presence Protocol) [14] to honeypot allow the admins to receive alerts about ongoing attack very quickly. We also instituted a central logging mechanism for aggregation and analyzing attacks to create a database center and warn the other parties about ongoing attacks.

The honeypot ran a standard server configuration of Linux CentOS 5 on a regular computer system. They were equipped with the following hardware:

- Pentium IV CPU 2+ GHz , 1 GB RAM
- NIC Cards 100/1000 Mbps , 80+ GB HDD

Each honeypot system has three components. In simple terms there are three layers with the processor or CPU as the central or lowest layer, the application as the outermost or highest layer and OS as a layer between them.

To run a 64-bit operating system you need support from the lower level: the 64-bit CPU. To run a 64-bit application you need support from all lower levels: the 64-bit OS and the 64-bit CPU.

This simplification works to explain what occurs when the 32-bit and 64-bit parts in our honeynet design are combined. The thumb rule is that 32-bit will run on a lower level 64-bit component. The main reason that our 32-bit honeypot will always run on 64-bit is that the 64-bit components have been designed to work that way. So the newer 64-bit systems are backward-compatible with the 32-bit systems and the choices in the platforms warrants the same result regardless of whether you select 32-bit or 64-bit platform.

We have installed Open VZ, in addition to the software provided by the above type of installation. Open VZ is a container-based virtualization for Linux. Also included are the Kojoney honeypot and P0f (a passive OS fingerprinting tool) [9].

First, we configured the Kojoney with default settings but after few days we made some changes in Kojoney to log more details about the attacks. Secondly, we configured the firewall so the SSH service of the server was accessible via its public static IP address from the Internet and finally, we install the p0f fingerprinting OS tools to identify the operating system on machines that connect to our box.

The honeypots were brought online on August 20th 2011 and taken offline on October 6th 2011 for full 47 days. The SSH honeypots servers face several attacks during this period of study, but were not successfully compromised because of the nature of the low-interaction honeypots. In this paper, we will focus on the attempts to gain remote access to a system using data collected from our SSH honeypots plus tools and techniques employed.

The other parts of this paper are organized as follows:

Section 2 provides a general view of the project. It includes the experimental test bed to observe the attacks and also a summary of usernames and passwords used in such attacks. It also looks at the origin of these attacks and their fingerprints on the OS. In Section 3, there is a detailed analysis of malicious traffic which provides insight into the methods and tools used by attackers. Section 4, gives recommendations about commonly used defenses against brute-force SSH attacks. Section 5 gives recommendation for future research in the same field. (Figure 1).

2. General View of the Project

2.1 Experimental Test Bed

SSH Honeypots were deployed in six different networks to collect as much data as possible about actual attacks on different network IP spaces. These six networks that were hosting the honeypots, were completely separated and had no explicit or logical links to inter connect them. In addition, a different ISP was used by each network (Figure 2).

The honeypots consisted of low-end PCs with CentOS Linux operating system. Each system constantly running two SSH services. The first service was a patched version of Kojoney SSH honeypot version 0.0.4.2 [10] that listened for attack traffic on TCP port 22. The second service, was intended for maintenance and control of the honeypots, ran the SSH server software provided with the Linux distribution and listened on a nonstandard high TCP port. (Figure 2)

Kojoney honeypot is written in Python [11] using the Twisted Conch Libraries [12] and report scripts are written in Perl [13].

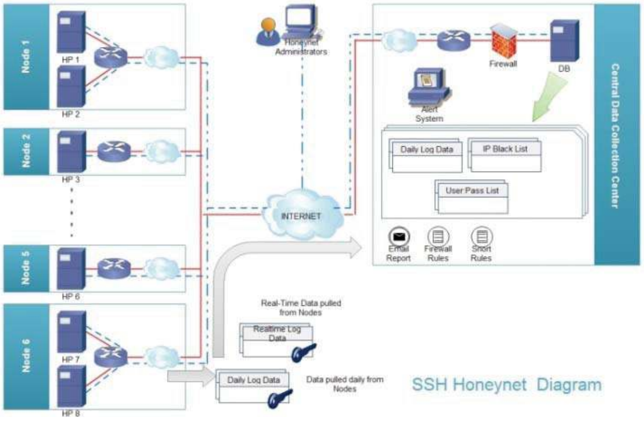

Figure 1. SSH honeynet network.

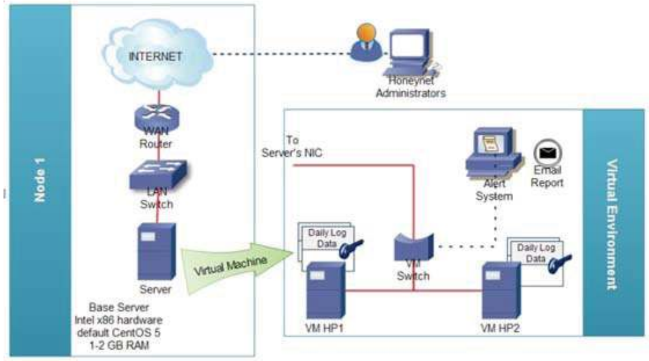

Figure 2. SSH honeypot in detail.

We have done few modifications to the Kojoney Software for more effectiveness which is as follows:

1. Add password logging function to the authentication mechanism to log the passwords used in all login attempts.
2. Add user agent detection function to find out which client software was used by attackers.
3. Add support for XMPP (Extensible Messaging and Presence Protocol) [14] to create warning system that should be able to alert the system administrator about ongoing attack activities.
4. Add support of P0f as an OS fingerprinting tools.
5. Upgrade the IP geolocation function to provide accurate information about attackers' origination.
6. Upgrade shell prompt mechanism to make the system more realistic.
7. In addition, a collection of scripts were written to extract attack data from the honeypot log files and insert them into a local database. For aggrega-

tion and analysis, the local databases were regularly synchronized with a central database server.

The source code of our modified version of SSH honeypot is available on http://code.google.com/p/kojoney-patch/.

We operated the honeypots from August 20th 2011 through October 6th 2011 for full 47 days. During this period we collected ~160 MBytes raw log data's (near 1206113 DB records), ~10 MBytes OS fingerprint data and ~50 malware samples.

2.2 Raw Data File

A large amount of raw data is generated by honeypots. Network security professionals use them for analysis and investigation. This raw data has typically rich content and high volume. On our low interaction honeypot this data run to hundreds megabytes.

Table 1 is a simplified example of raw logged data from our honeypot. When a piece of malware has been downloaded by the attackers, then this data can be extracted from the logged file records.

Each rows of log file formats as date and time is expressed as YYYY-MM-DD HH:MM:SS.MS, [Service function, Session ID, Attacker IP], Log Message. The rich records supply a large amount of alternate modes for analysing data.

2.3 Attack Attempts

This section provides a general view of the attacks in this research. Over the course of approximately 7 weeks, 8 honeypots were received to nearly 98,180 connection requests which were originated from 1153 IP addresses and 79 countries.

Across the honeypots, during each attack, a number of login attempts were observed. These ranged from one to two to hundreds or even thousands of attempts (Table 2). The highest number of attempts observed during a single attack session was 8,496. This attack, originated from Poland (Warsaw).

Of the 1153 IP addresses involved in attacks across the eight systems, on 6 honeypots only 3 addresses were consistently observed. Similarly, on more than 50% of all honeypots, 50 IP addresses were involved in the attacks. 177 IP addresses were observed in attacks on at least 2 honeypots and 976 IP addresses were observed in attacks on only one honeypot. Overall statistics are presented in Table 3; there is a break down data for each individual honeypot.

Table 1. Raw data from our honeypot (simplified)

```
2011-09-25 06:36:20.977 new connection: 61.x.x.x:45231
[session: 931]
2011-09-25 06:36:20.978 detected os: Linux 2.6 (newer, 3)
2011-09-25 06:36:20.978 detected connection: ethernet/
modem
2011-09-25 06:36:21.970 remote SSH version: SSH-2.0-
libssh-0.1
2011-09-25 06:36:21.970 kex alg,key alg: diffie-hellman-
group1-sha1 ssh-rsa
2011-09-25 06:36:23.535 root trying auth password
2011-09-25 06:36:23.631 login attempt root:sales succeeded
2011-09-25 06:36:23.831 got channel session request
2011-09-25 06:37:23.952 getting shell
2011-09-25 06:38:44.631 command is : uname -a
2011-09-25 06:39:11.551 command is : cd /var/tmp
2011-09-25 06:42:13.351 command is : wget eduteam.home.ro/
mech.gz
2011-09-25 06:44:25.151 saved the file /log/kojoney/ro_
mech_gz643 requested by the attacker.
2011-09-25 06:45:25.126 connection duration (msec): 544143
2011-09-25 06:45:25.127 remote compression type: none
2011-09-25 06:45:25.127 connection lost
```

Table 2. Top 10 session count

Country	City	Session count
Poland	Warsaw	8496
United States	Statesboro	4993
Panama	Panama City	3898
Iran	Tehran	3153
Canada	Toronto	2694
United States	Los Banos	2269
Indonesia	Jakarta	1875
Russian Federation	Yekaterinburg	1451
United States	Richmond	1291
Romania	Bucharest	1267

Table 3. Overall honeypot attack activity

	Connection	Successful authentication	Failed authentication	Source IP
HP 1	11174	628	9575	225
HP 2	3635	196	3221	72
HP 3	27571	5019	21625	414
HP 4	20312	2654	15342	209
HP 5	11228	913	9529	205
HP 6	8710	504	7942	137
HP 7	9652	268	3248	72
HP 8	5898	264	5393	136
Total	**98180**	**10446**	**75875**	**1153**

2.4 Login Attempts

Our SSH honeypots were configured to allow logins from any location using common username and password combinations. Login attempts for less common usernames were recorded even if they were unable to successfully authenticate to the system. As expected, the most username observed in malicious login attempts was Linux super user privileges called root which is a well-known account for UNIX, Linux, and BSD-based systems. Overall, the root account was targeted in just over a 66% of all login attempts.

The commonly targeted usernames are those often associated with temporary accounts which were test, guest or user. The constantly targeted accounts were the system accounts. Table 4 presents the "Top 15" usernames and also shows the respective percentages of total login attempts on these usernames. Interestingly, the list of system accounts seem to be dominated by the database system accounts.

According to our honeypots data, passwords based on username accounts were by far most commonly attacked. In fact, username:password pairs that were identical (e.g. root:root, oracle:oracle, test:test) were attempted in 19.0% of login attempts across all honeypots. Simple variations on the username to create passwords were used in another 5.83% of attempts. The most common variation was to simply append "123" to the username to form the password (e.g. mysql:mysql123). An alternate form of the username became other variations of passwords (the password "Leland321" used with username "Leland"). Another common variation was to simply double or triple the username to form the password, such as forming the password testtest from username test.

Table 5 lists the total percentage of login attempts along with list of passwords that where most frequently used in attacks on our honeypots.

Passwords similar to the username discussed above, are represented by %username%.

Near 70% of passwords used in attacks had less than 8 characters length and only 4.2% of them had more than 16 characters (Table 6). The longest password, observed was mt13hzxwUXu8PsT6KYExvLu5zgGEpC0vtmhVjg7-KIWknhzfCalwVinh3rqyh7Ui.

The results presented thus far correlate very well with those of earlier studies of malicious SSH login attempts [4, 5]. In their analysis, the studies focused on the most frequently observed usernames and passwords. This was a prelude to the study undertaken to comprehend the actions taken by the attackers to gain access to high-interaction honeypots. This research focuses on developing and evaluating recommendations for defense against brute-force attacks.

Table 4. Common username observed by honeypot

Username	Percentage used
Root	66.42
Test	1.27
Oracle	1.09
Admin	0.77
Nagios	0.61
User	0.48
Bin	0.44
Guest	0.40
Postgres	0.37
Mysql	0.31
Tester	0.25
Webmaster	0.20
Teamspeak	0.19
Testing	0.18
test123	0.18

Table 5. Common password observed by honeypot

Password	Percentage used
%Username%	19.0
%Username*%	5.83
123456	1.9
Password	1.0
Root	1.0
1234	0.80
12345	0.60
123	0.60
Changeme	0.59
Qwerty	0.58
Test	0.49
1q2w3e	0.45
abc123	0.44
Oracle	0.43
111111	0.40

The t-test is a statistical method used to compare group means. The independent sample t-test enables us to compare password lengths between successful login attempts and unsuccessful login attempts. Application of this statistical test to data tells us that the means of password lengths for two respective groups, successful and unsuccessful login attempts, are 5.44 and 8.33. We also see that the sig score is 0.000 (which is less than 0.05); therefore there is a significant difference between the means of the successful and unsuccessful login attempts.

While the independent sample t-test is limited to comparing the means of two groups, the one-way ANOVA (Analysis of Variance) can compare more than two groups. ANOVA uses F statistic for testing a significant difference between group means. In this paper, Duncan Multiple Range Test is used to identify which group differs from the others.

ANOVA for the comparison means of password lengths between continents shows that F (495.278) is significant (sig = 0.000 and less than 0.05) therefore there is a significant difference between the means of password lengths within continents (Consider the successful and unsuccessful login attempts). For the alpha = 0.05, Duncan's MRT (Table 7) indicated that members of Subset 4 (Attackers from North America) scored highest means of password length. Subset 2 (Attackers from Europe and South America) scored means of password length around 7.30 and were not significantly different from each other. Subset 1 (Africa) has the lowest password length with the mean of 6.32.

We also observed 339 username:password pairs were used in attacks involving all 8 of the honeypots. As shown in Table 8, the usage of simple usernames was found to be common. This designed list was aimed to explore, in a very short period, the large number of potentially vulnerable servers.

Table 9 shows the passwords used with usernames mentioned in Table 8.

Note: Passwords are equal by the username discussed above has been removed.

2.5 Sources of Attacks

Geographic distance has little relevance when dealing with hosts that are logically located on the Internet, which is physically worldwide. Regardless of the physical location of an attacker, the Internet makes everyone close enough to break in and steal things. Thus it can be important to examine the source of login attempts. Table 10 lists the

Table 6. Password lengths

Password Llength	Percentage used
<= 2	1.64
>2 AND <=4	11.59
>4 AND <=6	29.15
>6 AND <=8	27.82
>8 AND <=10	14.19
>10 AND <=14	10.0
>=16 AND <=30	3.70
>30	0.5

Table 7. Duncan test for password length (successful and unsuccessful attacks)

Continent	N	Subset for alpha = 0.05			
		1	2	3	4
Africa	863	6.32			
Europe	30600		7.29		
South America	3352		7.30		
Asia	23569			8.31	
North America	27936				8.65
Sig.		1.000	.0931	1.000	1.000

Table 8. Common usernames

Username				
admin	dev	Nagios	samba	upload
ant	ftpuser	News	scan	user
apache	git	ftp	staff	web
backup	guest	Office	student	webmaster
bin	info	Operator	support	www
cvsroot	jboss	Oracle	temp	www-data
cyrus	marketing	PlcmSpIp	test	zabbix
daemon	master	Postgres	teste	
david	mike	Qwerty	tester	
db2inst1	mysql	Root	tomcat	

Table 9. Common passwords

Password				
!@!@!@	1234pass	computer	guestguest	r00t123
!@#	1234qwer	connect!	hacker	r00tr00t
!@#$	123root123	Internet	iloveyou	root12345
#a1s2d3f4	123zxcasd	Login	p@$$w0rd	s3arch
#asd1234	Asdasd	Mario	p@$$w0rd1234	s3rver
*Q*W*E*R*T*Y*	Asdf	Matrix	p@ssw0rdmns	setup
000000	Attack	Core	p@ssword	t3mp
012345	Babe	Default	Private	t3st
0123456789	ch4ng3m3	demo123	professional	zaqxsw
0p3r4t0r	Cocacola	Foobar	qweqwe	zxcvbn

"Top 15" countries which attacks were originated from with their overall percentages of total attacks attempts. As you can see, two countries in particular stand out: the United States (U.S.) and China.

The chi-square test in contingency table (cross table) analysis used to determine whether the variables are statistically independent or if they are associated.

Pearson chi-square sig < 0.05 interpreted that continent and login status variables are associated (95% confidence level).

With respect to dependencies among variables and consider to the percentage of each cell we find that, attackers from Europe are more successful than their competitors in other continents with 56.60% success rate (Figure 3). Although, only 19.30% of all attacks originate from Europe was a success (Figure 4).

What we can gain from the geographic data is the ability to report the remote IP addresses of those who accessed the honeypot to the proper individuals who can fix the compromised hosts. Knowing where the system is physically located can provide a route for resolving the problem for those countries that have country-level computer response teams to handle network security issues within their jurisdictions [17].

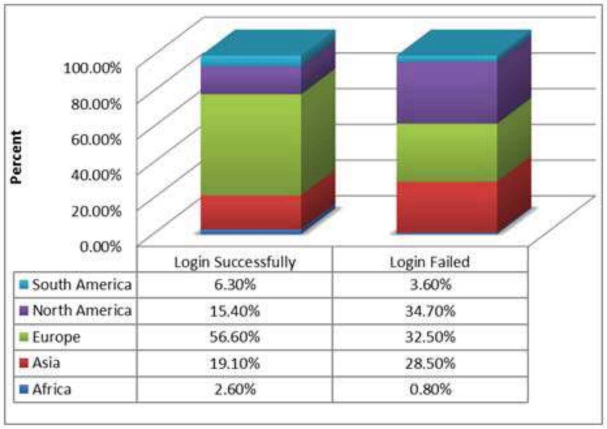

Figure 3. Login attempt stat by continent.

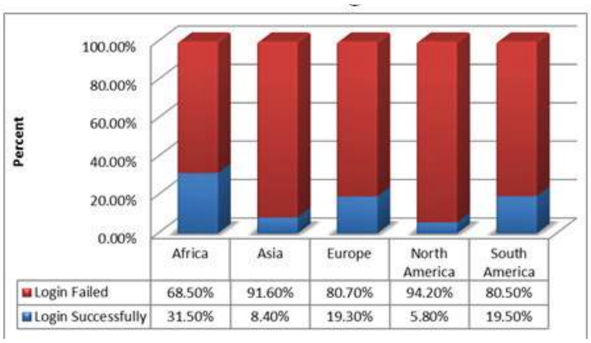

Figure 4. Login attempt stat within continent.

2.6 OS used in Attacks

Any Operating System can be used for attacks. Linux may be slightly more prone because, here, the micro-kernel approach is implemented differently from the micro-kernel of the windows OS. In Linux, the very basic functionalities are in the micro-kernel while most of the functionalities (including hardware and port access) are deployed in a kernel that is controllable in the user space. So, it provides more freedom to intercepting network connections. Other OS like Windows give only restricted access and also support much more identification over the network [18].

For our SSH honeypots, more than 82% of connections were established from a Linux system and only 3% was from Windows machine. Table 11 presents the "OS used" in attacks, along with their respective percentages.

Factorial Analysis of Variance (FANOVA) tested the effects of the OS used in attacks and the continents where attacks are originated from the length of password used in attacks.

Table 10. Source of attacks

Country	Frequency
United States	17.9
China	10.0
Poland	9.1
Canada	6.6
Argentina	6.1
Panama	4.1
Russian Federation	4.1
Germany	4.0
Iran	3.9
India	3.9
Romania	3.0
Thailand	2.8
Korea, Republic of	2.7
United Kingdom	2.5
Indonesia	2.5
Others (65 countries)	16.8

Table 11. OS used in attacks

OS	Percent
Linux	82.4
Unknown	14.4
Windows	3.2
FreeBSD	.0

Results indicated a significant main effect for the OS factor, F(2,86305) = 9.774, sig < 0.01. As hypothesized, those who used Linux OS in attacks used longer passwords (M = 8.07) compared to those who did not. There was also a significant main effect for the continent for which attacks originated from, F(4,86305) = 16.062 , sig < 0.01. Attackers from North America used longer passwords to breaking into the systems (M = 8.65) than those from Africa (M = 6.32).

The two main effects, OS and Continent, were qualified, however, by a significant interaction between the two factors, F(8,86305) = 35.573, sig < 0.01, indicating that the OS effects were not the same for different continents. Figure 5 shows the estimated marginal means of password length.

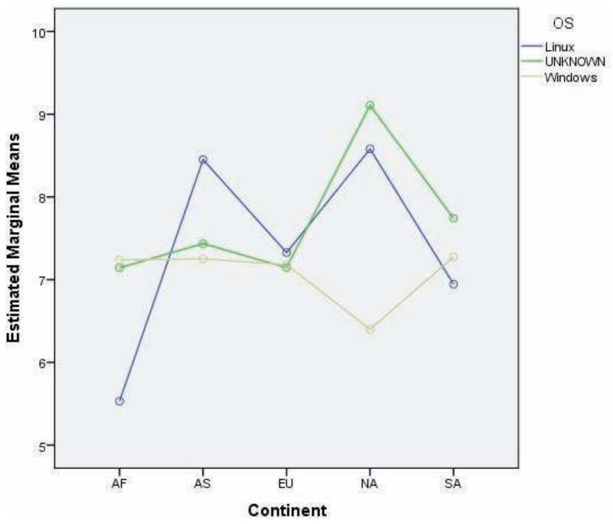

Figure 5. Estimated marginal means of password length.

2.7 User Agents used in Attacks

A user agent is an SSH client software which uses the secure shell protocol to connect to a remote computer.

More than 99% of connections to SSH honeynet were established by using libssh. Libssh is a multiplatform C library implementing the SSHv2 and SSHv1 protocol on client and server side. With libssh, you can remotely execute programs, transfer files, and use a secure and transparent tunnel for your remote applications [19]. Table 12 presents the "User agents used" by attackers, along with their respective percentages.

Table 12. User agents used in attacks

User Agents	Percent
SSH-2.0-libssh-0.1	85.3
SSH-2.0-libssh-0.2	10.3
SSH-2.0-libssh2_1.0	3.3
SSH-2.0-dropbear_0.47	.4
SSH-2.0-libssh-0.11	.4
SSH-2.0-dropbear_0.49	.1
SSH-2.0-PuTTY_Release_0.60	.1
Others	.1

3. Activities Inside the Honeypot

In this section, a deeper look into the activities was done. The different kinds of commands used by attackers on the honeypot were examined and some interesting malware was collected further discussion.

3.1 Attacker's Favourite Commands

After a user logged in, he or she entered in an emulated environment that allowed observation of all actions that the user attempted after the successful login. In general, users attempted to gain knowledge about the server–such as the processes running on the system and the version of the operating system–and could download files that allowed them to increase their system privileges or run software such as an IRC network bouncer. These shell sessions were of three categories:

- The first category were those intruders who left immediately after they had logged in, probably with the intension to return later or assuming that the server they wanted does not exist or is not available.
- The next category were intruders who run some basic fingerprinting command on the server, using **w**,**/proc/cpuinfo**, **uptime** and **uname - a** to figure out the basics of the server. Most hackers also used wget to download a large file to test the download rate of the server. This trend was more common in the Windows 2000 SP3. Probably this could be because it's a large file which is hosted by a fast server (Microsoft) and also because it is still a hotlink.
- The last category of these users was those some fingerprinting and proceeded to download and run malware. After the execution has been intercepted, some bogus error messages surfaced and the intruders were usually force to disconnect.

Table 13 presents the most commands observed, along with their respective percentages of total commands.

3.2 Malware Collected

Some significant challenges for investigation and defense of malware come from an elusive type of malware called Kernel Rootkits. The most notable are Persistent Kernel Rootkits. This is a special type of kernel rootkits that implant persistent kernel hooks to tamper with the kernel execution paths and allow them to hide their presence.

By design, rootkits attempt to hide their presence from all rootkit detection software. This includes various system utility programs (e.g., ps and ls) that manipulate OS kernel control-flows. The persistent kernel rootkits, by their nature implant kernel hooks on corresponding kernel-side execution paths to invoke the security programs.

US-CERT in 2008 confirmed that attackers are actively targeting Linux-based computing infrastructures, using stolen SSH keys to gain access to systems and install a rootkit known as "phalanx2". The rootkit itself isn't using kernel bug or an SSH flaw but CERT issued an advisory because it might happen. Fortunately, our proposed honeypot system doesn't allow such activity in first place because of the nature of low-interaction honeypot; our proposed system is completely immune against rootkits such as Phalanx2 and Adore. We have to note that this behavior is not the system limitation rather they have been designed to work this way.

Although, The purpose of SSH honeypots is not collecting malicious software (malware)–like what we can do with Nepenthes [20]–but collecting and analyzing them at the early stages may allow us to deal with them before they spreads massively and causes severe damage. Table 14 presents the most common malware observed.

After analyzing these malwares and our honeypots logs, one observes that:

- There has been multiple repacking and rewriting of the malware. Most of the tools were outdated and contained leftover files from previous installations.
- The main reasons for hacking a server seems to be:
  - i. Installing tools to hack more servers. e.g. WuScan, Unixcod, BBDoS
  - ii. Installing hidden web servers.
  - iii. Installing IRC daemons or bouncers. e.g. psyBNC
  - iv. Installing IRC bots. e.g. ZmEu, Shellbot
- and it is a surprise that no attempt to nuke the system was carried out by any of the intruders.
- Some intruders realized that a honeypot was in place by taunting the system with fake error messages.
- By appearances, the intruders seem to follow a kind of memorized script. They used the same tools and same

Table 13. Commands used inside honeypot

Commands	Percent
W	13.61
uname–a	9.47
Wget	5.68
Id	5.56
Ls	5.44
cat/proc/cpuinfo	5.44
Uptime	3.91
ps x	2.13
ls–a	2.01
Passwd	1.89
Whoami	0.59
Halt	0.12
Help	0.12
History	0.12
Netstat	0.12
php–v	0.12
Ifconfig	0.12

Table 14. Malware collected by honeypots

Malware	Detected as
ZmEu	Mal/Behav-183
wunderbar_emporium	None!
psyBNC_2_3_2_7	IRC-Bouncer
SynScan	HackTool.Linux.Small.af
Checkmech	IRC bot
Unixcod	Los Banos
udp	Perl/BBDoS.MN!tr
Ferry.tgz	Linux/Rst.B
vicssh	Linux/CleanLog.H
alexe	Unix.Mech.g
canim	Backdoor.Perl.Shellbot.a
goshNEW	HackTool.Linux.WuScan.b
coffe	Linux/ProcHider.C
blackenergy	Spyware.Unix.Mech.A
cobrel	Exploit.Linux.Small.f
Pinky	Exploit.Linux.Local.h
Solid	Backdoor.Perl.Shellbot.B

mode of fingerprinting in most of their attempts and executions. This brings in the question of the existence of a "SSH hacking school" in cyberspace or in more concise terms a "hacking tutorial" on a forum which doles out commands for execution of hacking.

From a malware perspective, currently the vast majority of existing malware in the wild, targets 32-bit platforms. Many of these older platforms are consumer machines that are poorly patched, and they make an easy target for botnet candidates. Attackers generally go for low-hanging fruit, so it makes sense for them to continue targeting 32-bit platforms rather than expend time and energy trying to defeat the more powerful security defences of 64-bit platforms. For example, researches show that 64-bit ASLR (Address Space Layout Randomization) is a strong protection against brute force attacks (buffer overrun attacks) and most of the working brute force exploits for a x86 architecture will not succeed on a x64 machine [21, 22].

3.3 Bots Involved

By going deep into the downloaded malwares, we revealed bunch of bot nets (channels, username, password and etc). Most of bots hosted on undernet.org (Table 15).

Table 16 lists the discovered bot channels.

4. Conclusions and Recommendations

It is common wisdom that Linux is superior to Windows with regard to security issues. With the growing popularity of open source software for the consumer (Android phones) and the enterprises (Linux runs the 10 fastest supercomputers in the world, according to Wikipedia); it's time to push past the reluctance and look at safety issues in a holistic manner.

Table 15. Server used by bots

208.83.20.130	SantaAna.CA.US.Undernet.org	ced.dal.net
94.125.182.255	Vancouver.BC.CA.Undernet.org	twisted.dal.net
Tampa.fl.us.undernet.org	ede.nl.eu.undernet.org	irc.emory.edu
Budapest.hu.eu.undernet.org	193.109.122.67	irc.gigabell.de
bucharest.ro.eu.undernet.org	66.186.59.50	irc.solidstreaming.net
195.197.175.21	irc.plur.net	irc.homelien.no
195.144.12.5	irc.concentric.net	irc.powersurfr.com
82.76.255.62	irc.du.se	irc.mcs.net
us.undernet.org	irc.psychoid.net	irc.mindspring.com

"Linux has been more widely deployed, which has certainly made it a bigger target to hackers in general," said Charlie Belmer, founder and CEO of security vendor Golem Technologies. But in terms of overall security it is still far superior to Windows.

The greatest advantage with regard to security for Linux rest with its huge, highly-skilled and diligent open source community". The open source nature of Linux allows for more peer review of the code to find and fix the code before zero day hacks can be done," said Williams. However, this does not make Linux invulnerable. It faces increasing threats as it gains in popularity though, it has only limited set of security solutions available to it. The number of vulnerabilities that require patching has been growing at a fast pace too [23].

Linux faces greatest threats due to social engineering and poorly configured systems. Like Windows, passwords are a serious liability to Linux.

In light of the insights gained from our research, it is now possible after collecting and analyzing a large amount of data from our honeypots, to offer a variety

Table 16. Bot channels and operators discovered

Channels	Operators
#op_op	*!*@koffe.users.undernet.org
#muieall	*!*@GTI.users.undernet.org
#system	*!*@shortty.users.undernet.org
#moarte	-psyBNC!psyBNC@lam3rz.de
#!@##@!	-Ryo-psyBNC!Ryo-Psybnc@
#@r@d	Anakbodoh.com
#FCU	*!*@Sinai.users.undernet.org
#FreeWay	*!*@bLaCkEnErGy.users.undernet.org
#kRs-oNe	*!*@axelinho.users.undernet.org
	**!*@Bummer.users.undernet.org
	!@sp00f3d.users.undernet.org
	!@n2u.users.undernet.org
	!@sinned.users.undernet.org
	!@Winder.users.undernet.org
	!@haihui.users.undernet.org
	!@Pimpologyst.users.undernet.org
	!@ZmEu.users.undernet.org
	!@newbies.users.undernet.org
	!@ZmEu.WhiteHat.ro
	!@Costi.users.undernet.org

Table 17. Suggestion and recommendation ordered by priority/effectiveness

Order	Suggestion	Description
1	Turning off the daemon service	There is likely no need for remote access via SSH if the computer system is a client machine and not a server [16].
2	Install a firewall	To restrict access to the SSH server from only authorized machines and networks [16].
3	Use the hosts.allow and hosts.deny	To restrict daemon access to specific hosts [16].
4	Restrain the SSH server to only authenticate particular users or groups	To restrict access to the SSH server from only authorized users and groups [16].
5	Use port knocking mechanism	Port knocking is a method of establishing a connection to a networked computer that has no open ports [24].
6	Use an alternate authentication methods	PublicKey and GSS-API
7	Implement One Time Password mechanism (OTP) or Single Packet Authorization (SPA)	SPA is essentially next generation port knocking, where only a single "knock" is needed, consisting of an encrypted packet. The one-time password system ensures that a password can't be reused. So, even if the password is captured in transit, it's worthless to an attacker once you've logged in with it [26].
8	Move the listening port	It will significantly reduce the likelihood of finding your SSH daemon [16].
9	Use intrusion detection/prevention systems	Through protocol analysis, content searching, and various pre-processors, Intrusion Detection System detects thousands of worms, vulnerability exploit attempts, port scans, and other suspicious behavior
10	Use ECC and RSA cryptographic algorithm for generating SSH keys	Elliptic Curve Cryptography (ECC) is proving to be a very secure, robust and light crypto algorithm [25]. RSA key size 1024 bits ~ NIST ECC key size 192 bits RSA key size 2048 bits ~ NIST ECC key size 224 bits RSA key size 3072 bits ~ NIST ECC key size 256 bits
11	Don't use a blank passphrase on your keys	If your keys are not protected by passphrases, then after scouring your shell history, or SSH config for hosts to connect to, they're in the SSH server with little effort [16].
12	Expiring and regenerating SSH keys periodically	The biggest reason to change your private key is if you have a reason to suspect it has been compromised or is no longer secure.
13	Limit the number of clients you SSH from	If an attacker can compromise your client, then they can get access to your SSH keys, as they are stored on the file system.
14	Use network level monitoring and security tools	Monitor log files - Remote Logging

of evaluated mitigation techniques. Table 17 lists some common recommendation for the protection of SSH servers. The study data can also provide additional defense strategies. Furthermore, many projects focus upon checking the strength of passwords too. Both cracklib [27] and OpenWall's pam_passwdqc [28] provide helper tools that transparently perform password checking as users change their passwords on Unix-based systems. Based on our findings, we can offer passwords list to those projects. We would like to continue to provide updates to collected information on a monthly, weekly, daily basis and even in real-time.

5. Future Work

Currently, work on developing more effective set of software tools that could support automatic consolidation and analysis of honeypot data at a central server is on. It could also readily support a variety of analysis activities that help collect and aggregate username:password pair data, as well as highlight the specific kinds of intruder activities. These can be designed to lower the volume of brute-force SSH attacks. In future, it is also possible to envision a centralized database of username:password pairs commonly used in malicious login attempts. These can be similar to the central DenyHosts [15] database of malicious IP addresses.

6. Acknowledgement

The authors are highly grateful to the Mashhad Branch, Islamic Azad University, Iran, for giving all types of support in conducting this research. This paper is based upon the information available in the project entitled “Design and Deployment of Honeynet for Analysis of Internet Attacks in Islamic Azad University, Mashhad Branch” with project number 90351/409/13.

The authors are sincerely grateful to the anonymous reviewer(s) who helped to improve the quality of this paper.

7. References

1. Spitzner L (2002). Honeypots, tracking the hackers. Available from: http://www.tracking-hackers.com
2. Scottberg B, Yurcik W et al. (2002). Internet honeypots: protection or entrapment, In IEEE International Symposium on Technology and Society (ISTAS), 387–391.
3. Lonvick C (2006). The Secure Shell (SSH) protocol architecture, IETF RFC 4251. Available from: http://www.ietf.org/rfc/rfc4251.txt
4. Ramsbrock D, Berthier R et al. (2007). Profiling attacker behavior following SSH Compromises, Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 119–124.
5. Seifert C (2006). Analyzing malicious SSH login attempts. Available from: http://www.securityfocus.com/infocus/1876
6. Alata E, Nicomette V et al. (2006). Lessons learned from the deployment of a high-interaction honeypot, In Proceedings of Dependable Computing Conference (EDCC06), 39–46.
7. SANS Institute (2007). SANS Top-20 2007 Security Risks (2007 Annual Update). Available from: http://www.sans.org/top20/2007/
8. Kojoney Project (2011). Available from: http://kojoney.sourceforge.net
9. The new P0f (2011). Available from: http://lcamtuf.coredump.cx/p0f.shtml
10. Kojoney-patch (2011). Available from: http://code.google.com/p/kojoney-patch
11. Python (2011). Available from: http://www.python.org
12. Event-driven networking engine written in Python (2011). Available from: http://twistedmatrix.com
13. Perl (2011). Available from: http://www.perl.org
14. The XMPP Standards Foundation (2011). Available from: http://xmpp.org
15. Welcome to DenyHosts (2011). Available from: http://denyhosts.sourceforge.net
16. Toponce A (2011). OpenSSH best practices. Available from: http://pthree.org/2011/07/22/openssh-best-practice
17. Observations of login activity in an SSH honeypot (2011). Available from: http://www.cisco.com/web/about/security/intelligence/ssh-security.html
18. Most of the hackers use Linux (2011). Available from: http://www.mylot.com/w/discussions/2101732.aspx
19. Libssh - the SSH library (2011). Available from: http://www.libssh.org
20. Nepenthes–finest collection (2011). Available from: http://nepenthes.carnivore.it
21. Ormandy T, and Tinnes J (2009). Linux ASLR curiosities. Available from: http://www.cr0.org/paper/to-jt-linux-alsr-leak.pdf
22. Shacham H, and Page M (2004). On the effectiveness of address-space randomization, CCS’04 Proceedings of the 11th ACM conference on Computer and Communications Security, 298–307.
23. Baker P (2011). Is linux really more secure than windows? Available from: http://www.esecurityplanet.com/trends/article.php/3933491/Is-Linux-Really-More-Secure-than-Windows.htm
24. Krzywinski M (2003). Port knocking: network authentication across closed ports, SysAdmin Magazine, vol 12(6), 12–170.
25. Rogers P, and Hering R (2011). RACF and digital certificates. Security Server RACF Security Administrator's Guide, 14th Edn., IBM Redbooks, USA, 571–631.
26. Single Packet Authorization with Fwknop (2005]). Available from: www.cipherdyne.org/fwknop/docs/SPA.html
27. CrackLib (2008). Available from: http://sourceforge.net/projects/cracklib
28. Password/passphrase strength checking and enforcement (2010). Available from: http://www.openwall.com/passwdqc