A Study of Passwords and Methods Used in Brute-Force SSH Attacks

Jim Owens and Jeanna Matthews
Department of Computer Science
Clarkson University
8 Clarkson Avenue, MS 5815
Potsdam, NY 13699
{owensjp, jnm}@clarkson.edu

ABSTRACT

In its Top-20 Security Risks report for 2007, the SANS Institute called brute-force password guessing attacks against SSH, FTP and telnet servers "the most common form of attack to compromise servers facing the Internet." A recent study also suggests that Linux systems may play an important role in the command and control networks for botnets. Defending against brute-force SSH attacks may therefore prove to be a key factor in the effort to disrupt these networks. In this paper, we report on a study of brute-force SSH attacks observed on three very different networks: an Internet-connected small business network, a residential system with a DSL Internet connection, and a university campus network. The similarities observed in the methods used to attack these disparate systems are quite striking. The evidence suggests that many brute-force attacks are based on pre-compiled lists of usernames and passwords, which are widely shared. Analysis of the passwords used in actual malicious traffic suggests that the common understanding of what constitutes a strong password may not be sufficient to protect systems from compromise. Study data are also used to evaluate the effectiveness of a variety of techniques designed to defend against these attacks.

1. INTRODUCTION

Major security threats to networked computer systems appear to be reaching crisis proportions in recent years. For example, Barracuda Networks, a major supplier of email and Web security appliances, estimates that spam email accounted for between 90 and 95 percent of all email sent during 2007 [2]. In addition, new phishing attacks increased by 18% during the first half of 2007 [27], and by the final quarter of last year phishing incidents accounted for nearly 60% of all security incidents reported [29]. Commercial malware kits such as MPack [24], including maintenance and support agreements for client hackers, are now being offered for sale on the Internet for as little as $500. These trends have continued to grow since Bruce Schneier told the audience at the Hack in the Box Security Conference in Kuala Lumpur, Malaysia that in his estimation the security war was being lost [19].

Perhaps the single biggest security threat for networked systems going forward is represented by *botnets*, defined as collections of compromised computer systems used for a variety of criminal activities, including distributed denial-of-service attacks, spamming, traffic sniffing, keylogging, identity theft, and click fraud [7]. The most highly publicized botnet of 2007 was the Storm worm botnet, which is estimated to control as many as 50 million computers [5].

For most of the recorded history of botnets, dating back to 1999, the robot computers, or *zombies,* that populate them have been understood to consist primarily of compromised systems running a version of the Microsoft Windows operating system [7,22]. Propagation of zombie code has been observed to occur through a number of Windows-specific worms, viruses, Trojans, and other forms of malware [3]. More recently, vulnerabilities in Linux machines are being recognized as an important part of the problem. In October 2007 Dave Cullinane, chief information and security officer at eBay, announced at the Trust Online conference that an internal investigation of the security threats faced by the online auction service had been traced to "rootkitted Linux boxes." [20] Alfred Huger, vice president for Symantec Security Response, echoed Cullinane's comments, saying that compromised Linux machines were frequently observed to make up a large portion of the command and control networks for botnets.

While it is true that computers running Linux are not subject to the many worms, viruses, and other malware that target Windows platforms, the Linux platform is known to be vulnerable to other forms of exploitation. A 2004 study conducted by the London-based security analysis and consulting firm mi2g found that Linux systems accounted for 65% of "digital breaches" recorded during the twelve-month period ending in October 2004 [6].

Recent studies of vulnerability trends point to two primary attack vectors: brute-force attacks against remote services such as SSH, FTP, and telnet, and Web application vulnerabilities [4,25]. In its Top-20 2007 Security Risks report, the SANS Institute called brute-force password guessing attacks against SSH, FTP and telnet servers "the most common form of attack to compromise servers facing the Internet." The report notes that unpatched flaws such as buffer overflow vulnerabilities in the authentication functions of these services can allow arbitrary code execution; however, the report also points up a much more mundane threat. Weak passwords are specifically identified as a potential Achilles heel in these systems, since "brute forcing passwords can be a used as a technique to compromise even a fully patched system."

In this paper, we focus specifically on brute-force SSH attacks. In particular, we analyze data collected from a large number of SSH brute-force attacks against Linux systems connected to different kinds of networks. We discuss patterns in the passwords used in these attacks, as well as the methods employed. We also use the data we collected to evaluate the effectiveness of various countermeasures that have been suggested for protecting systems against SSH brute-force attacks.

The remainder of this paper is organized as follows. Section 2 provides an overview of the project, including the experimental setup, an overview of attack activity, and a high-level summary of

usernames and passwords used in attacks. In Section 3, malicious traffic is analyzed in detail, providing insight into the methods used by attackers. In Section 4, we evaluate a number of commonly recommended defenses against brute-force SSH attacks. Section 5 describes related work, followed by a description of future work in Section 6. We conclude in Section 7.

2. PROJECT OVERVIEW

2.1 Experimental Setup

In order to collect as much data on actual attacks as possible, from a variety network types, we deployed SSH honeypots in three very different network environments:

- An Internet-connected small business network
- A residential system with a DSL Internet connection
- Our campus network

The honeypots consisted of low-end PCs with minimal Linux server installations. Each system ran two SSH servers. The first was a patched version of OpenSSH Server version 4.7 [10] that listened for attack traffic on TCP port 22. The second server, intended for maintenance and control of the honeypots, ran the SSH server software provided with the Linux distribution and listened on a nonstandard high port. The three networks hosting the honeypots are completely separate, with no explicit or logical links to connect them. In addition, each network used a different Internet service provider.

We implemented and applied two modifications to the SSH server software for the honeypots. First, we added a line to the password authentication function to log the passwords used in all login attempts. Second, we also hard-coded the function's return value to always indicate a failed login attempt, as we were not interested in allowing attackers to access the honeypots. In addition, we wrote a collection of scripts to extract attack data from the honeypot log files and insert it into a local database. The local databases were regularly synchronized with a central database server for aggregation and analysis.

We operated the honeypots in two phases, for periods of 5-6 weeks each. The first phase ran from mid-July through late-August 2007. The second phase ran from mid-December 2007 until early-February 2008.

2.2 Overview of Attack Activity

In this section, we begin with a basic overview of the brute-force attacks we observed. Over the course of approximately 11 weeks, the three honeypots were subjected to nearly 300 separate attacks, consisting of more than 103,000 login attempts, originating from 279 IP addresses.

The number of login attempts observed during each attack varied widely across the honeypots, from 1 or 2 up to hundreds or even thousands of attempts. The largest number of attempts observed during a single attack session was 9,311. This attack, observed on the honeypot located on the residential DSL connection, lasted for 117 minutes and accounted for nearly half of the login attempts observed on this honeypot.

Of the 279 IP addresses involved in attacks across the three systems, only 8 addresses were observed in attacks on more than one of the honeypots. No IP addresses were observed in attacks on all three. Thus, we recorded a total of 271 distinct IP addresses in our research. Overall statistics are presented in Table 1, broken down by individual honeypot.

Table 1. Overall honeypot attack activity

	Campus	Business	Residence	Totals
Attacks	109	125	64	298
Login attempts	42,031	43,131	18,669	103,831
Source IPs	96	119	64	279

2.3 Common Usernames and Passwords

As one might expect, the username observed most often in malicious login attempts was root. Overall, the root account was targeted in just over a quarter of all login attempts. Other usernames commonly targeted are often associated with temporary accounts, such as `test`, `guest` or `user`. System accounts were also commonly targeted. Table 2 presents the "top ten" usernames observed, along with their respective percentages of total login attempts. Interestingly, database systems appear to dominate the list of system accounts.

Beyond the root, system and temporary account names, the vast majority of usernames used in attacks were first names (e.g. `michael` or `cheryl`). We saw very little effort to target usernames such as those used in many U.S. organizations, which often combine all or part of the user's surname with the first and sometimes the middle initial. In fact, a search for such usernames based on the top ten American surnames from the 2000 U.S. Census [28] yielded just nine examples among all the usernames collected in our research.

Table 2. "Top 10" usernames observed in SSH attacks

Username	% Used
root	25.7
admin	2.1
test	1.6
a	0.9
guest	0.9
user	0.6
oracle	0.4
postgres	0.4
webmaster	0.3
mysql	0.3

Passwords based on account usernames were by far the most common in the attacks on our honeypots. In fact, identical username/password pairs (e.g. `root/root`, `guest/guest`, `michael/michael`) were used in nearly 49 percent of login attempts across all three honeypots. Passwords based on simple variations to the username were observed in another 8 percent of attempts. The most common variation was simply appending "123" to the username to form the password (e.g. `root/root123`). Other variations included passwords that were alternate forms of the username, such as the password `walter`

used with username `walt`, or the opposite male-female form, such as the password `samantha` used with the username `sam`. Another common variation was to simply double or triple the username to form the password, such as forming the password `testtest` from username `test`. Dictionary words accounted for just over 11 percent of all passwords collected. Table 3 lists the passwords seen most frequently in attacks on our honeypots, along with their overall percentages of total login attempts. Passwords based on the simple variations of the username discussed above are represented by *%username%*.

Table 3. "Top 10" passwords observed in SSH attacks

Password	% Used
%username%	56.9
123456	3.6
password	1.4
test	0.8
12345	0.6
test123	0.5
123	0.5
1234	0.5
passwd	0.4
admin	0.4

The results presented thus far correlate very well with those of earlier studies of malicious SSH login attempts [23,26]. These studies tended to focus on the most frequently observed usernames and passwords in their analyses, as a prelude to the study of the actions taken by attackers who gained access to high-interaction honeypots. In our research, we have chosen to focus on developing and evaluating recommendations for defending against brute-force attacks. We present the results of that analysis in the next section.

3. ATTACK PATTERNS

In this section, we dig deeper into the attack patterns we observed. We begin with an examination of the different types of passwords used in the attacks on the honeypots, followed by a discussion of some interesting attack scenarios.

3.1 Passwords and Attack Dictionaries

For SSH servers that permit password authentication, the passwords themselves are an obvious area of vulnerability. So we begin our analysis with an examination of the different kinds of passwords and dictionaries used in the attacks on our honeypots.

3.1.1 Passwords

One of the first questions raised in our analysis concerned the degree commonality that might exist in the passwords used in attacks across the honeypots. In the previous section, we presented the overall "Top 10" list of passwords collected, which was headed by passwords that were variations on the username. Of course, these passwords vary with the username. Putting passwords based on the username aside, we generated a list of the most frequently occurring passwords collected in each of the honeypots and compared them side-by-side. We found the similarity among these lists rather astonishing. Figure 1 below presents the 20 passwords seen most frequently in each honeypot. The passwords in the bold font are those that were found among the top 20 in all three honeypots. The passwords in italics were recorded in two of the lists. When evaluating these lists, we again point out that these passwords were generated in attacks originating from 279 IP addresses. Only eight of these IP addresses were observed in attacks on more than one honeypot.

Overall, 12 passwords were found in the top 20 list among all three honeypots, with another 5 occurring in two of the lists. These results might have been even more striking were it not for the presence of three of the longest passwords found in the Business honeypot's list:

```
asutcmhack123@
40232046bad
!@#asutcmhack!@#
```

These passwords were used hundreds of times each in combination with different usernames in a single attack on the Business honeypot. These passwords are also the strongest found in this list. In fact, the password `asutcmhack123@` received a "Best" rating when tested with Microsoft's online Password Checker tool [8], while the remaining two were rated as "Medium."

Campus	Business	Residence
123456	**123456**	**123456**
password	**password**	**password**
12345	**test**	**test**
test	**admin**	**12345**
admin	**test123**	**123**
1234	asutcmhack123@	**1234**
123	*passwd*	**test123**
root	40232046bad	*passwd*
qwerty	!@#asutcmhack!@#	1
abc123	**root**	12
administrator	**12345**	**root**
12345678	**qwerty**	**admin**
user	**1234**	changeme
linux	*mysql*	abc123
test123	**123**	**qwerty**
guest	*apache*	**guest**
mysql	*master*	1q2w3e
1234567	**user**	**user**
apache	*linux*	newpass
master	**guest**	asdfgh

Figure 1. The "Top 20" passwords from each honeypot.

3.1.2 Attack Dictionaries

The striking similarity we observed among the passwords most commonly used in attacks on the three honeypots led us to suspect that attackers might be using shared dictionaries of usernames and passwords. In fact, by examining the number of login attempts involved in attacks on the three honeypots and manually comparing the individual usernames and passwords used in each attack, we found evidence of at least five such dictionaries.

The criteria we used to identify these attack dictionaries were quite strict. Specifically, we considered two attack sessions to be

using the same dictionary only if they used exactly the same username/password pairs in precisely the same order. We also observed numerous partial runs of similar username/password lists; however, these were not counted.

Table 4 below provides some statistics on the frequencies with which the dictionaries we identified were used in attacks. We named the dictionaries according to the number of username/password pairs contained in each. The total of 51 attacks using these dictionaries accounted for 17 percent of all the brute-force SSH attacks observed on the honeypots. Given the strict criteria used to define each dictionary, we find this result quite striking. Additional information on the individual dictionaries is provided in the following paragraphs.

Table 4. Username/password dictionaries used in SSH attacks

	Campus	Business	Residence	Total
Dictionary-9	7	4	6	17
Dictionary-66	1	2	0	3
Dictionary-168	8	6	10	24
Dictionary-363	1	1	2	4
Dictionary-373	2	0	1	3
Totals	19	13	19	51

3.1.2.1 Dictionary-9

The smallest of the 5 dictionaries we observed, including 9 username/password pairs, was used in a total of 17 attacks involving all 3 of the honeypots. As shown in Figure 2 below, the usernames and passwords used are quite simple. This dictionary was clearly designed to permit exploration of a large number of potentially vulnerable servers in a very short period. The average time required to complete the 17 attacks observed using this dictionary was just under 22 seconds.

Usernames	Passwords
test	test
guest	guest
admin	admins
user	user
root	password
root	root
root	123456
test	123456

Figure 2. Usernames/passwords included in Dictionary-9.

3.1.2.2 Dictionary-66

All username/password pairs contained in this dictionary were specifically directed at the root account. The passwords used include a small number of the sort found in the Top 20 lists above, as well as some simple phrases like `changeme` and `trustno1`. However, the majority of the passwords found in this dictionary are based on simple keyboard patterns:

```
qazwsxedc
qpwoeiruty
1q2w3e4r
!@#$%^
```

3.1.2.3 Dictionary-168

This dictionary proved to be the most popular choice for attacks on the honeypots. It includes a large variety of usernames including root; various system accounts; generic and/or temporary account names such as staff, sales, and recruit; as well as proper names. The included passwords are quite simple throughout, with the vast majority being limited to the username or a simple variation thereon. We identified three distinct versions of this dictionary, each of which individually met the criteria described above for defining dictionaries. That is, each version was used in attacks on multiple honeypots, using the exact same username/password pairs occurring in precisely the same order. Each version incorporated a small number of modifications (10 or fewer) to the usernames, passwords, or both from other versions. Interestingly, despite these minor differences, each version of Dictionary-168 contained the same number of username/password pairs.

3.1.2.4 Dictionary-363 and Dictionary-373

These dictionaries include a diverse collection of usernames and passwords and may simply represent a conglomeration of smaller dictionaries. The root account and various system accounts are well represented, with passwords of varying types including common English words, proper names, keyboard patterns, and "leets," which replace letters with numbers or symbols that resemble the replaced letter. For example, these dictionaries include these variations on the word password:

```
p@ssw0rd
p@ssword
passw0rd
pa$$word
pa55word
pa55w0rd
```

Both of these dictionaries also include more than a hundred identical username/password pairs based on proper names.

3.2 Attack Methods

As noted in the previous section, the number of login attempts observed during individual attack sessions varied widely. More than a third consisted of ten or fewer login attempts, while other attackers attempted hundreds or even thousands of logins in a single session. In fact, in about 10 percent of attacks, more than 1,000 login attempts were recorded.

While the vast majority of attacks seemed fairly straightforward, we recently observed a small number of attacks that appear specifically designed to evade detection by intrusion prevention systems. We provide details of these attacks in the paragraphs below.

3.2.1 A Slow-motion Brute-force SSH Attack

Beginning on January 1 and continuing through January 8, 2008, we observed a total of 21 separate attack sessions on a single honeypot originating from the same IP address. The number of logins attempted during each session varied somewhat, but the number of logins attempted during a single session never exceeded nine. The total number of login attempts over the eight days was 130, all of which targeted the root account.

The passwords used in the initial 50 or so attempts over the first 3 days were quite simple. They consisted mostly of common English words, proper names, and simple phrases such as `newuser`, `stuffedturkey`, and `youareok`. The passwords used in the next session, consisting of nine login attempts, consisted mostly of "leets" such as `c4bl3m0d3m` (cablemodem), `c4l3nd4r` (calendar), and `c4lif0rni4` (california).

Beginning with session number 11 and continuing throughout the remaining attacks sessions, the passwords were much stronger. In fact, of the passwords used in the last 73 login attempts, 53 percent were rated as "Strong" by Microsoft Corporation's Password Checker tool [8]. A representative sample of these passwords is presented in Figure 3 below.

U50s8AdF
OxZBA4xOMd
35t3K6OZ
Zh59EPu5mQxq
8Nv9YUpQu0v
K48v87GR8Rf
QcxC3OuZUH
848TmMf57
bC28s9R7Weg
nezBh57yi1jm
Kqr17tJ89Tan

Figure 3. "Strong" passwords used during a slow-motion brute-force SSH attack on a single honeypot.

3.2.2 A Distributed Brute-force SSH Attack

We observed another attack apparently designed to evade detection by intrusion prevention systems. This attack consisted of a coordinated series of login attempts originating from 10 consecutive IP addresses from the same Class C network. A total of 33 logins were attempted in just over 3 minutes, with no more than 5 attempts originating from a single IP address. The sequence of login attempts is shown in Figure 4 below. Interestingly, the username/password pairs used in this attack are identical to the first 32 pairs found in one version of the attack dictionary designated as Dictionary-168 in the previous section. Although distributed among 10 different source IPs addresses, the username/password pairs used in this attack were in exactly the same order as in other attacks originating from a single IP.

Time	Username	Password	IP Address
10:42:34	staff	staff	aaa.bbb.ccc.131
10:42:39	sales	sales	aaa.bbb.ccc.136
10:42:44	recruit	recruit	aaa.bbb.ccc.131
10:42:51	alias	alias	aaa.bbb.ccc.137
10:42:58	office	office	aaa.bbb.ccc.137
10:43:03	samba	samba	aaa.bbb.ccc.137
10:43:08	tomcat	tomcat	aaa.bbb.ccc.131
10:43:13	webadmin	webadmin	aaa.bbb.ccc.136
10:43:21	spam	spam	aaa.bbb.ccc.138
10:43:29	virus	virus	aaa.bbb.ccc.134
10:43:36	cyrus	cyrus	aaa.bbb.ccc.139
10:43:41	oracle	oracle	aaa.bbb.ccc.136
10:43:46	michael	michael	aaa.bbb.ccc.134
10:43:51	ftp	ftp	aaa.bbb.ccc.137
10:43:57	test	test	aaa.bbb.ccc.135
10:44:05	webmaster	webmaster	aaa.bbb.ccc.138
10:44:10	postmaster	postmaster	aaa.bbb.ccc.134
10:44:15	postfix	postfix	aaa.bbb.ccc.139
10:44:21	postgres	postgres	aaa.bbb.ccc.139
10:44:26	paul	paul	aaa.bbb.ccc.131
10:44:32	root	root	aaa.bbb.ccc.131
10:44:38	guest	guest	aaa.bbb.ccc.133
10:44:43	admin	admin	aaa.bbb.ccc.139
10:44:49	linux	linux	aaa.bbb.ccc.138
10:44:54	user	user	aaa.bbb.ccc.140
10:45:00	david	david	aaa.bbb.ccc.139
10:45:06	web	web	aaa.bbb.ccc.136
10:45:11	apache	apache	aaa.bbb.ccc.137
10:45:17	pgsql	pgsql	aaa.bbb.ccc.132
10:45:22	mysql	mysql	aaa.bbb.ccc.134
10:45:30	info	info	aaa.bbb.ccc.138
10:45:35	tony	tony	aaa.bbb.ccc.135
10:45:45	core	core	aaa.bbb.ccc.138

Figure 4. A distributed brute-force SSH attack.

We believe that these attacks represent fledgling efforts to lower the volume of brute-force SSH attacks, and thereby avoid detection. We fully expect to see more sophisticated attacks using these and similar methods to extend the time periods between login attempts and distribute the attempts among a greater number of IP addresses. In fact, distributed SSH attacks would seem to be a likely application for large, distributed botnets.

4. EVALUATION OF COMMON DEFENSES AGAINST SSH ATTACKS

Having collected and analyzed a large amount of data on brute-force SSH attacks, we now offer an evaluation of a variety of mitigation techniques that are commonly recommended for protecting SSH servers, in light of the insights gained from our research. We also suggest some additional defense strategies based on our study data.

Enforcing strong passwords with password checking programs or libraries. Much has been written on what constitutes a strong password. A quick Web search turns up a long list of sites offering advice on this topic. One such site is Microsoft Corporation's page: "Strong passwords: How to create and use them" [9]. The advice offered on this page reflects the broad consensus of the criteria that constitute a strong password:

- Make it lengthy
- Combine letters, numbers, and symbols.
- Use words and phrases that are easy for you to remember, but difficult for others to guess

Microsoft's site also offers a six-step tutorial for creating a strong, memorable password. The final step includes a link to Microsoft's Password Checker tool [8], a utility that helps users determine the strength of candidate passwords.

While many resources are available for helping users choose strong passwords, the challenge for many system administrators is to get their users to actually select and use strong passwords.

Fortunately, password checking libraries that can prevent users from choosing weak or vulnerable passwords are readily available. Perhaps the most commonly used are the Openwall Project's pam_passwdqc PAM module [17] and the cracklib library [18].

The pam_passwdqc module is simple to install, highly configurable, provides support for passphrases, and subjects candidate passwords to a number of checks including minimum password length and the presence of weak substrings. The pam_passwdqc module can also generate random passwords.

The cracklib module provides for similar checking. Candidate passwords are tested for strings related to the username and GECOS data, as well as simple patterns and dictionary words. Administrators can also incorporate checks against password lists. The cracklib project Web site provides one such list, which currently contains more than 1.6 million words culled from a variety of sources, including the passwords captured in our honeypots.

We believe that enforcing strong passwords is arguably the most important step system administrators can take to protect SSH servers from brute-force password attacks. As noted in the SANS Institute's most recent Security Risks report, even fully patched systems are vulnerable to brute force password-guessing attacks. Password-checking libraries such as cracklib can prevent users from inadvertently choosing vulnerable passwords such as those based on their usernames. Cracklib's ability to check password choices against restricted systematic approaches to generating passwords is every bit as important, we believe. Our research shows that a significant percentage of malicious login attempts are based on dictionaries of usernames and passwords. While the majority of these passwords are obviously weak by any standard, we observed a significant percentage of "strong" passwords being used in some attacks. Collecting and using attack dictionaries in password checking can help users avoid selecting passwords vulnerable to compromise, regardless of their perceived strength.

Avoiding easily guessed usernames. Our results show that the usernames in malicious login attempts that target the accounts of real users consist almost exclusively of first names. The use of account names based on combinations of surnames with initials, or similar schemes that produce less easily guessable account names can do much to complicate the job of brute-force attackers.

Disabling logins via SSH for the root account. It has long been considered good security practice to disable logins via SSH for the root account. One of the first challenges faced by attackers engaged in brute-force SSH attacks is that of obtaining or guessing valid user account names. The root account is an obvious target, since it is known to exist on all Unix/Linux systems. By disabling SSH logins to root, system administrators complicate the job of the attacker. Even when root logins via SSH are disabled, login attempts fail silently. So the attacker has no way of knowing whether these attempts have any chance of succeeding. If a non-privileged account is compromised, the attacker gains a foothold on the system and may be able to gain full privileges through a local root exploit.

Our results show that the root account was targeted in more than 25 percent of all malicious login attempts. Therefore, by disabling access to this account, system administrators can render useless a significant percentage of malicious traffic. Successfully targeting other user accounts requires some research, a bit of luck on the attacker's part, a high volume of login attempts, or a combination of all three.

Running the SSH server on a non-standard high port. SSH servers traditionally listen on TCP port 22, but there is nothing to prevent system administrators from configuring SSH servers to listen on any other unused port among the 65,535 ports provided by the TCP protocol. All the SSH servers we are aware of can be readily configured to listen on alternative ports. We believe this situation creates a great opportunity to hide the SSH service from attackers, much like the proverbial needle in a haystack. Commonly-used port scanning tools such as Nmap [13] scan just over 1,600 ports by default, leaving the vast majority unexplored. Moreover, a recent study of the relationship between port scans and attacks [21] concluded that more than 50 percent of the observed attacks were not preceded by a port scan. Some will argue that this method is an example of "security by obscurity." However, we believe that running an otherwise well-secured SSH server on a nonstandard high port can help reduce its vulnerability to brute-force attacks without exposing the server to additional risk. We also note that all three honeypots used in this study ran a second SSH server on a high port, which was used for maintenance and control purposes. No malicious login attempts directed at the servers running on these ports were observed during the same period that over 100,000 attacks were observed on the default SSH port. Asking legitimate users to remember the non-standard port can be a small inconvenience.

Using TCP Wrappers or iptables to block IP addresses after repeated failed login attempts. A number of intrusion prevention tools, such as DenyHosts [14], BlockHosts [15], and fail2ban [16], have been introduced over the past several years to help defend against brute-force password-guessing attacks. These tools work by parsing system log files for failed login attempts on a periodic basis, and then taking action to lock out attacking IP addresses using iptables, TCP Wrappers, or null routing rules. The DenyHosts tool is focused on protecting the SSH service, while BlockHosts can be used to protect both SSH and FTP servers. The fail2ban tool is more flexible in that it can be configured to protect SSH, FTP, and Web servers.

In addition to parsing log files for attacking IP addresses on the local machine, DenyHosts also provides a synchronization function through which blocked IP addresses on individual servers running the software worldwide can be synchronized with a central server. Using this system, participating servers can be configured to periodically synchronize their /etc/hosts.deny files with the central server. In this way, attacks by many blocked hosts can be prevented before the attacker has the chance to initiate even one login attempt.

We found that over 93 percent of the 271 malicious IP addresses collected in our study were listed in the /etc/hosts.deny file a local server synchronized with the DenyHosts central database. Servers using this service would therefore have been protected from the vast majority of the attacks observed in our study. On the other hand, we observed a small number of attacks that appear to be specifically designed to thwart these systems, based as they are on the attacker's IP address. These efforts do not yet seem highly effective; however, we anticipate they will improve over the coming months.

It should also be noted that there may be some administrative overhead associated with managing systems like DenyHosts. Initial installation and configuration are quite straightforward, in our experience. On the other hand, depending on the number of users involved, the effort required to restore service for legitimate users who inadvertently lock themselves out of systems after repeated login failures could be significant.

Using iptables to restrict access to the SSH port by source IP address. System administrators can restrict network access to the SSH port (and those of other services) to specific source IP addresses or networks by adding source address restrictions to iptables firewall rules. A well-written set of iptables rules, designed to limit access to an SSH server to a set of authorized IP addresses, can be quite effective is preventing brute-force attacks. For server installations where the source IP addresses are known in advance, this method should work well. In many installations, however, restricting access to a set of known IP addresses may not be feasible and would prevent authorized users from logging in from unexpected locations. It should also be noted that writing iptables rules can be a complex undertaking, and poorly crafted rule sets may inadvertently leave servers vulnerable to attack.

Using port-knocking or single packet authorization to restrict access to the SSH server port. Iptables firewall rules can also be adjusted on the fly, using tools such as `knockd` [11] or `fwknop` [12], to allow SSH server access to specific IP addresses. Access is granted based on predetermined sequences of ICMP packets or a specially-crafted UDP packet, respectively. Access attempts from IP addresses that do not provide the required authorization packets are filtered. In situations where the source IP addresses of authorized users is not known in advance, port knocking or SPA can provide added flexibility. These methods require client software with the correct configuration to be installed on all systems used to connect to the SSH server. This additional overhead and the inconvenience it poses for users may limit the feasibility of this method in some organizations.

Requiring public-key authentication in place of passwords. SSH servers such as OpenSSH [10] support a variety of authentication methods. One commonly-used method that virtually eliminates the threat of brute-force password guessing attacks is public-key authentication. To use this method, users must generate a public/private key pair and place the public key in the appropriate file on the destination server. The private key, in turn, must be stored on each client system from which the user wishes to log in to the server. To provide protection against brute-force password attacks, the server's system administrator must also disable all password-based SSH authentication.

While public-key authentication is not always feasible because of the overhead involved in generating and distributing keys, SSH servers configured in this way are virtually immune to brute-force attacks, provided all password-based authentication is disabled.

Summary of recommendations. Overall, we find that a number of the recommended techniques for defending against brute-force attacks can be quite effective, especially when used in combination. For installations in which password-based authentication is a necessity, we believe that enforcing strong passwords is the most effective method for defending against brute-force SSH attacks. Such a strategy should include not only systems that rate the strength of passwords based on length and character choice, but also by using a system such as cracklib with dictionaries of passwords actually captured in honeypots. We also recommend avoiding usernames based on simple first names. Where possible, our data indicated that running the SSH server on non-standard ports is also quite effective. Combining password checking with other techniques designed to lower the profile of the server or to reduce the volume of malicious login attempts should help to greatly reduce the likelihood of system compromise by means of brute-force SSH attacks.

5. RELATED WORK

Several studies of SSH attack traffic have been undertaken in recent years [1,23,26]. In most cases, the study of SSH attack traffic is part of a larger study, which includes attacker activities following system compromise. In our research, we were narrowly focused on the malicious login traffic itself, with the goal of developing a deeper understanding of the tools and techniques employed in brute-force SSH attacks which, by many accounts, continue to represent a significant threat to networked Linux systems [25]. We were not interested in observing successful compromises. In fact, we patched the OpenSSH server to prevent successful logins via the standard SSH port, and we instituted a number of safeguards to protect the honeypots from compromise.

Microsoft offers a Web-based tool [8] that allows users to test the strength of candidate passwords without sending their passwords over the Internet. We used the Microsoft tool to test the strength of a number of passwords collected in our research activities.

There are a number of projects focused on password checking, as well. Both cracklib [18] and OpenWall's pam_passwdqc [17] provide helper tools that transparently perform password checking as users change their passwords on Unix-based systems. Based on our early findings regarding the widespread use of attack dictionaries of common usernames and passwords, we reached out to the maintainers of the cracklib project to offer the passwords collected in our research for inclusion in cracklib-words. We continue to provide updates to this list on a monthly basis.

6. FUTURE WORK

Deploying and managing low-interaction honeypots such as those fielded in our study is a fairly straightforward process. The work of aggregating and analyzing the data collected is more labor intensive. We are currently working to develop a set of software tools to support automatic consolidation and analysis of honeypot data at a central server, which would readily support a variety of analysis activities to support collection and aggregation of username/password data, as well as highlighting the specific kinds of attack activities designed to lower the volume of brute-force SSH attacks. We envision developing a toolkit that system administrators could easily download, install, and configure to collect data on malicious activity at their own sites.

We can also envision a centralized database of usernames/ passwords commonly used in malicious login attempts, similar to the central DenyHosts database of malicious IP addresses.

7. CONCLUSIONS

The armies of compromised computer robots, known as botnets, have received a lot of attention over the past few years. To date, most of that attention has been focused on the compromised Windows machines thought to populate the ranks of botnet armies. Until the results of eBay's recent study of internal security threats were publicized last fall, little attention was paid to the

role compromised Linux systems might play in supporting botnets.

Compared with systems running the Windows operating system, Linux systems face a unique threat of compromise from brute-force attacks against SSH servers that may be running without the knowledge of system owners/operators. Many Linux distributions install the SSH service by default, some without the benefit of an effective firewall. Thus, otherwise conscientious system administrators who keep their systems fully patched may fall prey to a system compromise caused by a carelessly chosen password.

As our study results show, not all vulnerable passwords can be considered weak, based on commonly-held beliefs of password strength. Attackers are using and sharing attack dictionaries of username/password pairs that incorporate a significant percentage of apparently strong passwords. Using a password checking tool, especially one that restricts systematic approaches to password selection, can provide an extra measure of protection against malicious login traffic, especially when combined with other protective measures designed to reduce the visibility of Internet-facing servers.

8. ACKNOWLEDGEMENTS

Thanks to Nathan Neulinger, maintainer of the Cracklib project, for his excellent efforts and the valuable feedback he provided on our paper.

9. REFERENCES

[1] E. Alata, V. Nicomette, M. Kaaniche, M. Dacier, M. Herrb. "Lessons learned from the deployment of a high-interaction honeypot", in Proc. Dependable Computing Conference (EDCC06), Coimbra, Portugal, October 18-20, 2006, pp. 39 - 46.

[2] Barracuda Networks. December 12, 2007. Barracuda Networks Releases Annual Spam Report. Available at: http://www.barracudanetworks.com/ns/news_and_events/index.php?nid=232.

[3] Canavan, J. 2005. White Paper: Symantec Security Response; The Evolution of Malicious IRC Bots. Available at: http://www.symantec.com/avcenter/reference/the. evolution.of.malicious.irc.bots.pdf.

[4] Christey, S & Martin, R. May 22, 2007. Common Weakness Enumeration. Vulnerability Type Distributions in CVE. Available at: http://cwe.mitre.org/ documents/ vuln-trends/index.html.

[5] Gaudin, S. September 6, 2007. InformationWeek. Storm Worm Botnet More Powerful Than Top Supercomputers. Available at: http://www.informationweek.com/news/ showArticle.jhtml?articleID=201804528.

[6] Hochmuth, P. November 11, 2004. LinuxWorld. Linux is 'most breached' OS on the Net, security research firm says. Available at: http://www.linuxworld.com.au/index.php/id;188808220;fp;2;fpid;1.

[7] The Honeynet Project and Research Alliance. Know Your Enemy, Tracking Botnets. http://honeynet.org/papers/bots, March 2005.

[8] http://www.microsoft.com/protect/yourself/password/checker.mspx.

[9] http://www.microsoft.com/protect/yourself/password/create.mspx

[10] http://openssh.org.

[11] http://www.zeroflux.org/knock/

[12] http://www.cipherdyne.org/fwknop/

[13] http://nmap.org

[14] http://denyhosts.sourceforge.net

[15] http://www.aczoom.com/cms/blockhosts

[16] http://www.fail2ban.org

[17] http://www.openwall.com/passwdqc/

[18] http://sourceforge.net/projects/cracklib

[19] Lemon, S. September 20, 2006. ComputerWorld Security. Bruce Schneier: We are losing the security war. Available at: http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9003477.

[20] McMillan, R. October 5, 2007. ComputerWorld. eBay: Phishers getting better organised, using Linux. Available at: http://computerworld.co.nz/news.nsf/scrt/CD0B9D97EE6FE411CC25736A000E4723.

[21] S. Panjwani, S. Tan, K. Jarrin, and M. Cukier, "An Experimental Evaluation to Determine if Port Scans are Precursors to an Attack," in Proceedings of the International Conference on Dependable Systems and Networks (DSN-2005), Yokohama, Japan, June 28-July 1, 2005, pp. 602-611.

[22] Moheeb Abu Rajab , Jay Zarfoss , Fabian Monrose , Andreas Terzis, "A multifaceted approach to understanding the botnet phenomenon," in Proceedings of the 6th ACM SIGCOMM on Internet measurement, October 25-27, 2006, Rio de Janeriro, Brazil.

[23] Ramsbrock, D. Berthier, R. & Cukier, M. 2007. "Profiling Attacker Behavior Following SSH Compromises," in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp.119-124.

[24] Sachs, M. June 20, 2007. MPack Analysis. Available at: http://isc.sans.org/diary.html?storyid=3015.

[25] SANS Institute. 2007. SANS Top-20 2007 Security Risks (2007 Annual Update). Available at: http://www.sans.org/ top20/ 2007/.

[26] Seifert, C. September 11, 2006. SecurityFocus. Analyzing Malicious SSH Login Attempts. Available at: http://www.securityfocus.com/infocus/1876.

[27] Symantec December 17, 2007. Symantec Looks Back at the Internet Security Trends and Threats of 2007. Available at: http://www.symantec.com/about/news/resources/press_kits/detail.jsp?pkid=endofyear.

[28] US Census Bureau. Frequently Occurring Surnames From Census 2000. Available at: http://www.census.gov/ genealogy/ www/freqnames2k.html.

[29] US-CERT. December 3, 2007. Quarterly Trends and Analysis Report, Volume 2, Issue 4. Available at: http://www.us-cert.gov/press_room/trendsanalysisQ407.pdf