



Ethical Hacking Lab Series

Lab 8: Using John the Ripper to Crack Linux Passwords

**Certified Ethical Hacking Domain:
System Hacking**

Document Version: 2013-07-18 (Beta)

Organization: Moraine Valley Community College
Author: Jesse Varsalone

Copyright © National Information Security, Geospatial Technologies Consortium (NISGTC)

The development of this document is funded by the Department of Labor (DOL) Trade Adjustment Assistance Community College and Career Training (TAACCCT) Grant No. TC-22525-11-60-A-48; The National Information Security, Geospatial Technologies Consortium (NISGTC) is an entity of Collin College of Texas, Bellevue College of Washington, Bunker Hill Community College of Massachusetts, Del Mar College of Texas, Moraine Valley Community College of Illinois, Rio Salado College of Arizona, and Salt Lake Community College of Utah. This work is licensed under the Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/3.0/> or send a letter to Creative Commons, 444 Castro Street, Suite 900, Mountain View, California, 94041, USA.



The Center for Systems Security and Information Assurance (CSSIA), in partnership with the Network Development Group (NDG) is given a perpetual worldwide waiver to distribute per US Law this lab and future derivatives of these works.

Contents

Introduction	3
Domain: System Hacking	3
Pod Topology	4
Lab Settings	5
1 Cracking Linux Passwords with John the Ripper	6
1.1 Locating and Cracking Linux Passwords	6
1.2 Conclusion	18
1.3 Discussion Questions.....	18
2 Creating an Additional Account with root Level Permissions	19
2.1 Creating another 'root'	19
2.2 Conclusion	28
2.3 Discussion Questions.....	28
3 Using the SSH Keys to Break into Linux	29
3.1 SSH Keys	29
3.2 Conclusion	32
3.3 Discussion Questions.....	32
References	33

Introduction

This lab is part of a series of lab exercises intended to support courseware for Ethical Hacker training. The development of this document is funded by the Department of Labor (DOL) Trade Adjustment Assistance Community College and Career Training (TAACCCT) Grant No. TC-22525-11-60-A-48.

In this lab, students will become familiar with the location where Linux passwords are stored and learn about tools and techniques for breaking Linux passwords.

This lab includes the following tasks:

- [1](#) – Cracking Linux Passwords with John the Ripper
- [2](#) – Creating an Additional Account with root Level Permissions
- [3](#) – Using the SSH Keys to Break into Linux

Domain: System Hacking

Passwords help to secure systems running Linux and UNIX operating systems. If an attacker is able to get the root password on a Linux or UNIX system, they will be able to take complete control of that device. The protection of the root password is critical.

passwd – User accounts on a Linux system are listed in the passwd file which is stored in the /etc directory. The passwd file has less restrictive permissions than the shadow file because it does not store the encrypted password hashes. On most Linux systems, any account has the ability to read the contents of the passwd file.

shadow – The shadow file also stores information about user accounts on a Linux system. The shadow file also stores the encrypted password hashes, and has more restrictive permissions than the passwd file. On most Linux systems, only the root account has the ability to read the contents of the shadow file.

auth.log – This log file tracks SSH, or Secure Shell, connections. It provides information such as IP addresses, and date and time stamps. It also tracks other events related to security, such as the creation of new user's accounts and new group accounts.

John the Ripper – John the Ripper is an extremely fast password cracker that can crack passwords through a dictionary attack or through the use of brute force.

SSH – The SSH protocol uses the Transmission Control Protocol (TCP) and port 22. Credentials and files that are transferred using SSH are encrypted. Most Linux systems have native SSH client capabilities. Some Linux systems also come packaged with an SSH server, often referred to as sshd, or Secure Shell Daemon. Microsoft Windows systems do not have the built in capability to use ssh natively. However, there are third party ssh client utilities, like putty, and ssh server utilities that can be utilized for Windows. The Cisco IOS also has a built in ssh client and has the capability of running an SSH server.

Pod Topology

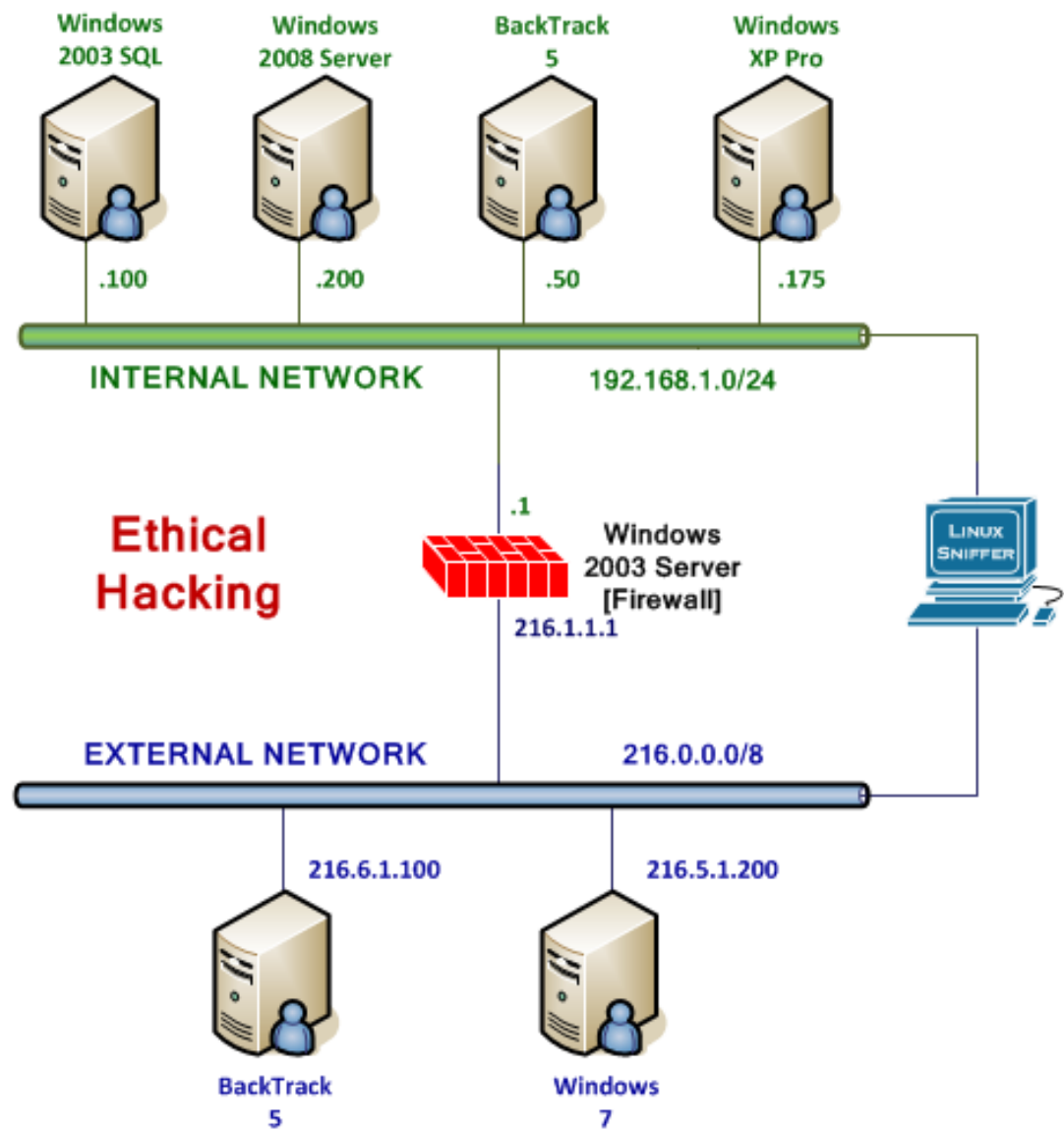


Figure 1: Lab Topology

Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Required Virtual Machines and Applications

This lab requires the use of the Linux *Internal* and *External* **BackTrack 5** machines.

<i>Internal</i> BackTrack 5	192.168.1.50
<i>Internal</i> BackTrack 5 root password	toor
<i>External</i> BackTrack 5	216.6.1.100
<i>External</i> BackTrack 5 root password	Toor
Windows 7	216.5.1.200
Windows 7 student password	password

1 Cracking Linux Passwords with John the Ripper

Passwords help to secure systems running the Linux operating system. If an attacker is able to get the root password on a Linux system, they will be able to take complete control of that device. The password hashes on a Linux system reside in the shadow file. John the Ripper is an extremely powerful password cracker. It comes loaded by default on all versions of BackTrack, but can be downloaded at www.openwall.com/john/.

Keep in mind that **Linux commands are case sensitive**. The commands below must be entered exactly as shown.

1.1 Locating and Cracking Linux Passwords

Open a Terminal on the *External* BackTrack 5 System

1. Open a terminal on the **External BackTrack 5** Linux system by clicking on the picture to the right of the word **System** in the task bar in the top of the screen.

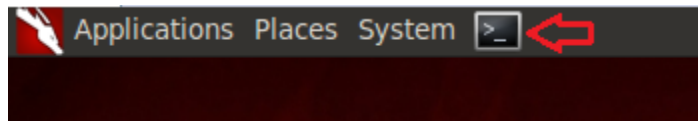


Figure 2: The Terminal Windows within BackTrack

After you click on the shortcut to the terminal, the terminal window will appear below.

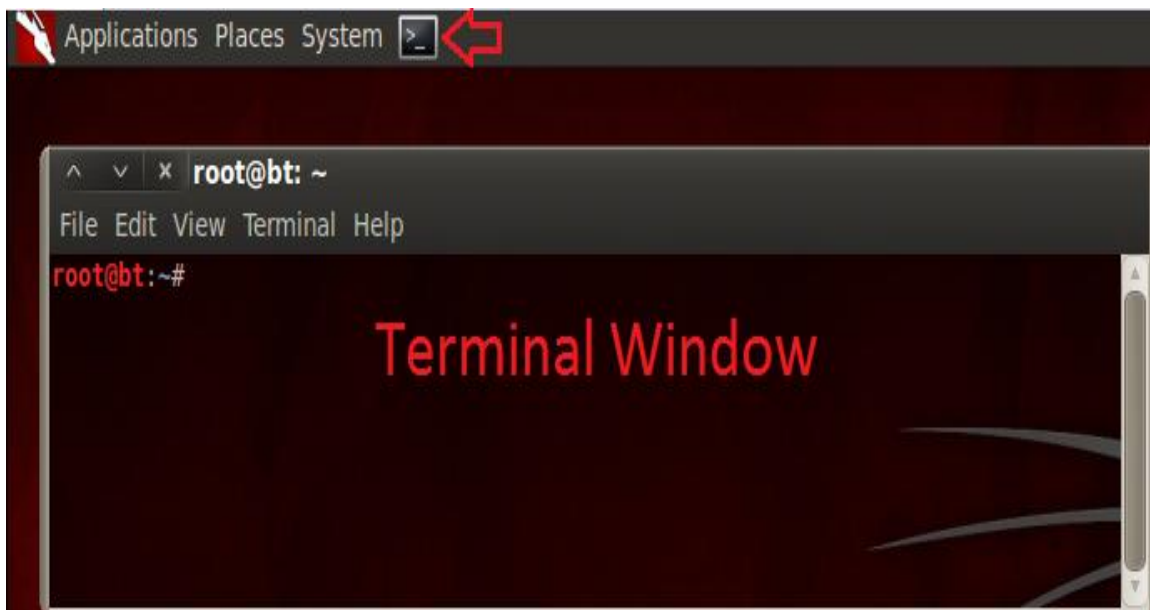


Figure 3: The BackTrack Terminal will appear

First, we will examine the passwd file, which contains the list of all of the user accounts on the Linux system. The passwd file is located within the /etc directory.

2. To view the contents of the passwd file, type:

```
root@bt:~#cat /etc/passwd
```

```
root@bt:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
sshd:x:102:65534::/var/run/sshd:/usr/sbin/nologin
landscape:x:103:108::/var/lib/landscape:/bin/false
messagebus:x:104:112::/var/run/dbus:/bin/false
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
mysql:x:105:113::/var/lib/mysql:/bin/false
avahi:x:106:114::/var/run/avahi-daemon:/bin/false
snort:x:107:115:Snort IDS:/var/log/snort:/bin/false
statd:x:108:65534::/var/lib/nfs:/bin/false
usbmux:x:109:46::/home/usbmux:/bin/false
pulse:x:110:116::/var/run/pulse:/bin/false
rtkit:x:111:117::/proc:/bin/false
festival:x:112:29::/home/festival:/bin/false
postgres:x:1000:1000::/home/postgres:/bin/sh
ftp:x:113:121:ftp daemon,,,:/srv/ftp:/bin/false
hax0r:x:1001:1001::/home/hax0r:/bin/sh
```

Figure 4: The passwd file

3. View the permissions on the /etc/passwd file by typing the following command:

```
root@bt:~# ls -l /etc/passwd
```

```
root@bt:~# ls -l /etc/passwd
-rw-r--r-- 1 root root 1434 2012-09-12 10:45 /etc/passwd
```

Figure 5: The Permissions on the passwd file

Notice that all users have at least read permissions. Only root has write permissions. At one time, the password was stored in the passwd file. However, due to the fact that the passwd file does not have very restrictive permissions, the password is no longer stored there. Instead, there is an X present, which designates that it is stored in the shadow file.


```
root@bt:~# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Figure 6: Details of passwd

4. To view the contents of the shadow file, type:

```
root@bt:~#cat /etc/shadow
```

```
root@bt:~# cat /etc/shadow
root:$6$EVCJcswz$gs9NsKYLw.gIRFQBfcmPho4xcxSo5AIPQYqNnd9gH4n2jokGqD7T5caWQUQ3ZZsr3wt.hvnlPbcIks
hVa4LTa.:15562:0:99999:7:::
daemon:x:15562:0:99999:7:::
bin:x:15562:0:99999:7:::
sys:x:15562:0:99999:7:::
sync:x:15562:0:99999:7:::
games:x:15562:0:99999:7:::
man:x:15562:0:99999:7:::
lp:x:15562:0:99999:7:::
mail:x:15562:0:99999:7:::
news:x:15562:0:99999:7:::
uucp:x:15562:0:99999:7:::
proxy:x:15562:0:99999:7:::
www-data:x:15562:0:99999:7:::
backup:x:15562:0:99999:7:::
list:x:15562:0:99999:7:::
irc:x:15562:0:99999:7:::
gnats:x:15562:0:99999:7:::
libuuid:x:15562:0:99999:7:::
syslog:x:15562:0:99999:7:::
sshd:x:15562:0:99999:7:::
landscape:x:15562:0:99999:7:::
messagebus:x:15562:0:99999:7:::
nobody:x:15562:0:99999:7:::
mysql:!:15562:0:99999:7:::
avahi:!:15562:0:99999:7:::
snort:!:15562:0:99999:7:::
statd:!:15562:0:99999:7:::
usbmux:!:15562:0:99999:7:::
pulse:!:15562:0:99999:7:::
rtkit:!:15562:0:99999:7:::
festival:!:15562:0:99999:7:::
postgres:!:15562:0:99999:7:::
ftp:!:15595:0:99999:7:::
hax0r:$6$uFnJF8V3SYFTyaoBvV8NQB9ztEVeBcwKZ96Mrj1DF6V2uGM6lmHF7HgZr.zk87LyaNu2M1XA2fvRKJgrJ0ZIq
Gq7VcyHT0:15595:0:99999:7:::
```

Figure 7: The shadow file

The two accounts, root and hax0r, that have passwords have password hashes. If we create some additional accounts, we can see how the passwd and shadow files are altered. We can also view the information about account changes within the secure log.

5. To create a new user named yoda, type the following command in the terminal:

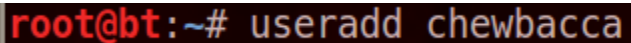
```
root@bt:~# useradd yoda
```

```
root@bt:~# useradd yoda
```

Figure 8: Adding an account

- To create a new user named chewbacca, type the following command in the terminal:

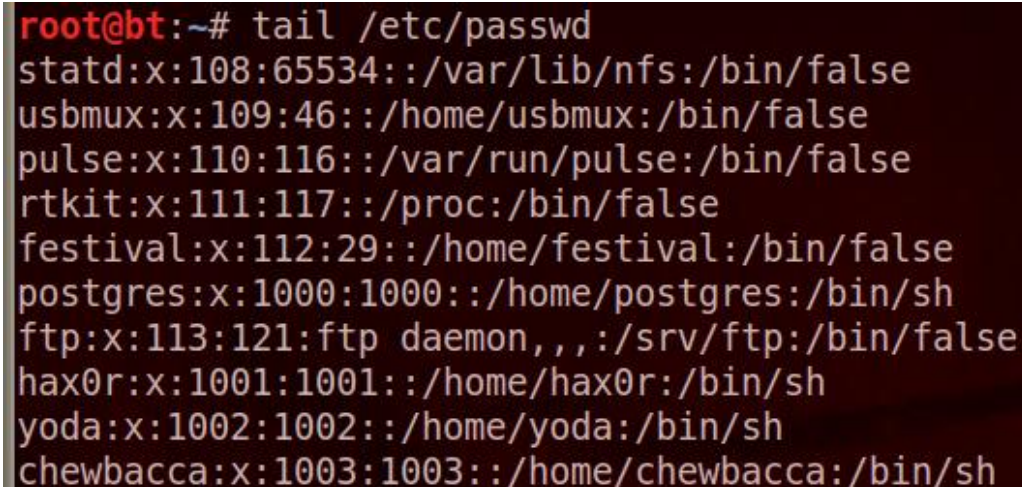
```
root@bt:~# useradd chewbacca
```



```
root@bt:~# useradd chewbacca
```

Figure 9: Adding an account

- Now, view the changes made to the passwd file by typing the following:



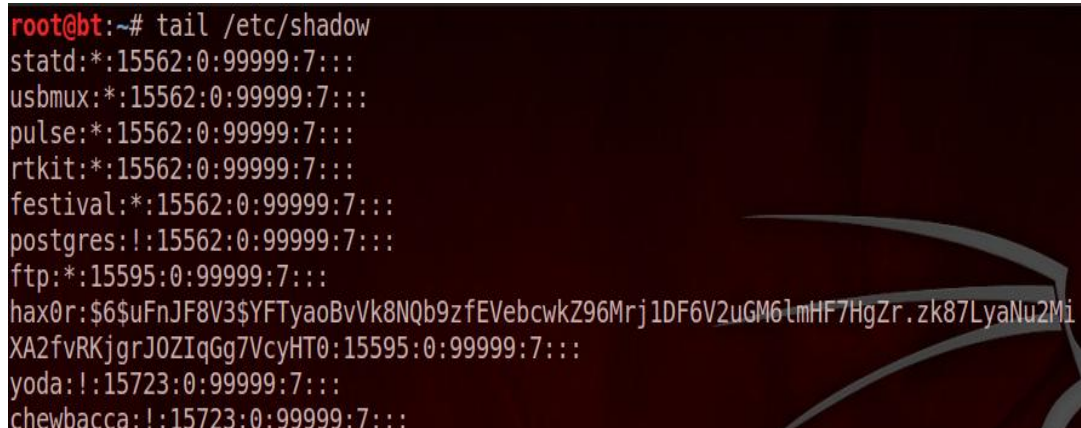
```
root@bt:~# tail /etc/passwd
statd:x:108:65534::/var/lib/nfs:/bin/false
usbmux:x:109:46::/home/usbmux:/bin/false
pulse:x:110:116::/var/run/pulse:/bin/false
rtkit:x:111:117::/proc:/bin/false
festival:x:112:29::/home/festival:/bin/false
postgres:x:1000:1000::/home/postgres:/bin/sh
ftp:x:113:121:ftp daemon,,:/srv/ftp:/bin/false
hax0r:x:1001:1001::/home/hax0r:/bin/sh
yoda:x:1002:1002::/home/yoda:/bin/sh
chewbacca:x:1003:1003::/home/chewbacca:/bin/sh
```

Figure 10: The passwd file

The tail command will display the last 10 lines of the file by default. When users are added to a Linux/UNIX system, the entries are added to the bottom of the file. On a typical Linux system, the first new user is given a User ID, or UID of 1001. In this case, an account called hax0r had already been created on the system. Yoda and Chewbacca were given the next available user IDs. The root account has a UID of zero. If another account were able to obtain an UID of 0, the account would also have root permissions.

- Next, examine the alterations to the shadow file by typing the following:

```
root@bt:~# tail /etc/shadow
```



```
root@bt:~# tail /etc/shadow
statd:!:15562:0:99999:7:::
usbmux:!:15562:0:99999:7:::
pulse:!:15562:0:99999:7:::
rtkit:!:15562:0:99999:7:::
festival:!:15562:0:99999:7:::
postgres:!:15562:0:99999:7:::
ftp:!:15595:0:99999:7:::
hax0r:$6$uFnJF8V3$YFTyaoBvVk8Nqb9zfEvebcwkZ96Mrj1DF6V2uGM6lmHF7HgZr.zk87LyaNu2Mi
XA2fvRKjgrJ0ZIqGg7VcyHT0:15595:0:99999:7:::
yoda:!:15723:0:99999:7:::
chewbacca:!:15723:0:99999:7:::
```

Figure 11: The shadow file

The “!” symbol (often called a bang) represents that fact the password has not been set.

9. Examine the entries in the auth.log related to account changes by typing:

```
root@bt:~# tail /var/log/auth.log
```

```
root@bt:~# tail /var/log/auth.log
Jan 18 11:26:28 bt login[1526]: pam_unix(login:session): session opened for use
r root by LOGIN(uid=0)
Jan 18 11:26:28 bt login[1623]: ROOT LOGIN on '/dev/tty1'
Jan 18 11:26:37 bt polkitd(authority=local): Registered Authentication Agent fo
r session /org/freedesktop/ConsoleKit/Session3 (system bus name :1.11 [/usr/lib
/policykit-1-gnome/polkit-gnome-authentication-agent-1], object path /org/gnome
/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Jan 18 11:39:01 bt CRON[1942]: pam_unix(cron:session): session opened for user
root by (uid=0)
Jan 18 11:39:01 bt CRON[1942]: pam_unix(cron:session): session closed for user
root
Jan 18 11:58:43 bt useradd[2033]: new group: name=yoda, GID=1002
Jan 18 11:58:43 bt useradd[2033]: new user: name=yoda, UID=1002, GID=1002, home
=/home/yoda, shell=/bin/sh
Jan 18 12:02:22 bt useradd[2054]: new group: name=chewbacca, GID=1003
Jan 18 12:02:22 bt useradd[2054]: new user: name=chewbacca, UID=1003, GID=1003,
home=/home/chewbacca, shell=/bin/sh
```

Figure 12: The auth.log file

Next, we will give each user a password. We will use simple passwords for the exercise, but that should never be done on a production system. Avoid dictionary words because attackers can use programs like John the Ripper to crack short passwords or passwords that are found in a dictionary. Stick to passwords with a minimum of eight characters, uppercase and lowercase letters, and special characters. Retype the password and it will be accepted.

For security reasons, the password will not be displayed.

10. Set yoda’s password to green by typing “green” twice after typing:

```
root@bt:~# passwd yoda
```

```
root@bt:~# passwd yoda
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Figure 13: Setting a Password for the User

11. Set chewbacca's password to green by typing "green" twice after typing:
root@bt:~# **passwd chewbacca**

```
root@bt:~# passwd chewbacca
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Figure 14: Setting a Password for the User

12. Next, examine the alterations to the shadow file by typing the following:
root@bt:~# **tail -n 2 /etc/shadow**

```
root@bt:~# tail -n 2 /etc/shadow
yoda:$6$S3IeZf2j$gWIfxoZkINVcvUjLxdI2YlOC1NS.d8hS.UCgrVns3MuDrD6GVYajdx8xr4zzdnk
YhdSD0rCGLCR9r/1mLph60.:15724:0:99999:7:::
chewbacca:$6$MeJxZyv4$IEg.BegR/elIQRYATU6051W15Yfk7yZFr6.3t4Cb0NMGMAqHsjQt3BMTH7
aKadjYK/zupr5oQKytEhB0ReZkP.:15724:0:99999:7:::
```

Figure 15: The shadow file

The password hashes are salted, which means if you give two users the same exact password, a different hash will be displayed. When salting is done, you will be unable to perform a rainbow table attack. Instead, you will need to perform a dictionary or brute force attack. You cannot use a rainbow table attack against a hash that has been salted. Both user's passwords were set to "green" but are different because they were salted. Changes to accounts, such as setting a password, will be logged in the auth.log.

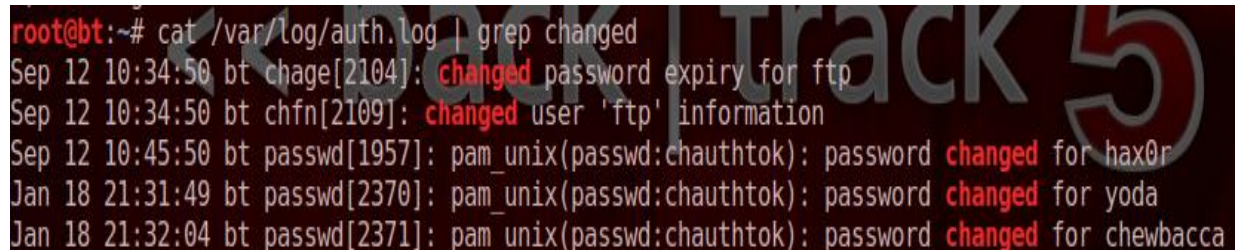
13. Examine the entries in the auth.log related to account changes by typing:
root@bt:~# **tail /var/log/auth.log**

```
root@bt:~# tail /var/log/auth.log
Jan 18 21:31:49 bt passwd[2370]: gkr-pam: couldn't update the login keyring password: no old password was entered
Jan 18 21:31:49 bt passwd[2370]: Error attempting to parse .ecryptfsrc file; rc = [-13]
Jan 18 21:31:49 bt passwd[2370]: Passphrase file wrapped
Jan 18 21:31:49 bt passwd[2370]: eCryptfs PAM passphrase change module retrieved at least one NULL passphrase; nothing to do
Jan 18 21:31:56 bt passwd[2371]: eCryptfs PAM passphrase change module retrieved a NULL passphrase; nothing to do
Jan 18 21:32:04 bt passwd[2371]: pam_unix(passwd:chauthtok): password changed for chewbacca
Jan 18 21:32:04 bt passwd[2371]: gkr-pam: couldn't update the login keyring password: no old password was entered
Jan 18 21:32:04 bt passwd[2371]: Error attempting to parse .ecryptfsrc file; rc = [-13]
Jan 18 21:32:04 bt passwd[2371]: Passphrase file wrapped
Jan 18 21:32:04 bt passwd[2371]: eCryptfs PAM passphrase change module retrieved at least one NULL passphrase; nothing to do
```

Figure 16: The auth.log file

Results will vary, but in some cases, only one or none of the password changes will show up in the log. They are in the log, but tail only provides the last ten entries of the file. Specific information within a file can be extracted by using the grep (global regular expression print) command. The grep command is native to most Linux distributions.

14. To look for specific information about password changes within auth.log, type:
`root@bt:~# cat /var/log/auth.log | grep changed`



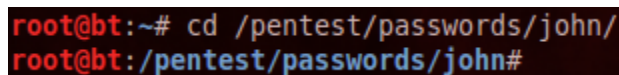
```

root@bt:~# cat /var/log/auth.log | grep changed
Sep 12 10:34:50 bt chage[2104]: changed password expiry for ftp
Sep 12 10:34:50 bt chfn[2109]: changed user 'ftp' information
Sep 12 10:45:50 bt passwd[1957]: pam_unix(passwd:chauthtok): password changed for hax0r
Jan 18 21:31:49 bt passwd[2370]: pam_unix(passwd:chauthtok): password changed for yoda
Jan 18 21:32:04 bt passwd[2371]: pam_unix(passwd:chauthtok): password changed for chewbacca

```

Figure 17: GREP for change

15. Switch to the **john** directory by typing the following command:
`root@bt:~# cd /pentest/passwords/john`



```

root@bt:~# cd /pentest/passwords/john/
root@bt:/pentest/passwords/john#

```

Figure 18: Switching to the john directory

16. Type the following command to see available switches for the john command:
`root@bt:~# ./john`



```

root@bt:/pentest/passwords/john# ./john
John the Ripper password cracker, version 1.7.6-jumbo-12
Copyright (c) 1996-2011 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--config=FILE           use FILE instead of john.conf or john.ini
--single[=SECTION]      "single crack" mode

```

Figure 19: The john command

17. Type the following command to attempt to crack the passwords with john:
 root@bt:~/pentest/passwords/john#./john /etc/shadow

```
root@bt:~/pentest/passwords/john# ./john /etc/shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 4 password hashes with 4 different salts (sha512crypt [32/32])
toor          (root)
green         (yoda)
green         (chewbacca)
hacker        (hax0r)
guesses: 4   time: 0:00:00:37 DONE (Fri Jan 18 21:53:36 2013) c/s: 252 trying: hacker
Use the "--show" option to display all of the cracked passwords reliably
```

Figure 20: Cracking the Passwords

Notice that even though there were only 3 different passwords in the list, the messages from john indicated that it *loaded 4 password hashes with 4 different salts*. If you need to view the password hashes and the corresponding revealed passwords at future time, you can always retrieve them from the john.pot file where they are stored.

18. To view the password hashes and corresponding passwords, type the following:
 root@bt:~/pentest/passwords/john#cat john.pot

```
root@bt:~/pentest/passwords/john# cat john.pot
$6$EVCJcsWz$gs9NsKYLw.gIRFQBfcMPho4xcxSo5AIPQYqNnd9gH4n2jokGqD7T5caWQUQ3ZZsr3wt.hvn1PbcIkshVa4LTa.:toor
$6$S3IeZf2j$glWIfxoZkINVcvUjLxdI2YlOC1NS.d8hS.UCgrVns3MuDrD6GVYajdx8xr4zzdnkYhdSD0rCGLCR9r/1mlph60.:green
$6$MeJxZyv4$IeG.BegR/e1IQRYATU6051W15Yfk7yZFr6.3t4CbONMGMAqHsjQt3BMTH7aKadjYK/zupr5oQKytEhB0ReZkP.:green
$6$uFnJF8V3$YFTyaoBvVk8NQb9zfEVeBcwKZ96Mrj1DF6V2uGM6lmHF7HgZr.zk87LyaNu2MiXA2fvRKjgrJOZIqGg7VcyHT0:hacker
```

Figure 21: The john.pot file

Unfortunately, if you attempt to crack the password again, you will not have success.

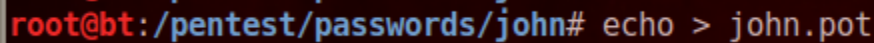
19. Type the following command to attempt to crack the passwords with john
 root@bt:~/pentest/passwords/john#./john /etc/shadow

```
root@bt:~/pentest/passwords/john# ./john /etc/shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 4 password hashes with 4 different salts (sha512crypt [32/32])
No password hashes left to crack (see FAQ)
```

Figure 22: No Passwords to Crack

This is because hashes and their corresponding passwords are stored within the john.pot file; john will not crack the password hash again. If you want the passwords to be cracked again, you will need to remove the information stored in the john.pot file.

20. Type the following command to attempt to crack the passwords with john
 root@bt:~/pentest/passwords/john# **./john /etc/shadow**



```
root@bt:~/pentest/passwords/john# echo > john.pot
```

Figure 23: Deleting john.pot

21. Type the following command to re-crack the passwords with john
 root@bt:~/pentest/passwords/john# **./john /etc/shadow**



```
root@bt:~/pentest/passwords/john# ./john /etc/shadow
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 4 password hashes with 4 different salts (sha512crypt [32/32])
toor          (root)
green         (yoda)
green         (chewbacca)
hacker        (hax0r)
guesses: 4   time: 0:00:00:34 DONE (Fri Jan 18 22:11:50 2013) c/s: 273 trying: hacker
Use the "--show" option to display all of the cracked passwords reliably
```

Figure 24: Re-cracking the Passwords

The first four passwords cracked were done via brute force. A brute force attack on a password hash usually takes the longest. If a password has a large number of characters and is very complex, the brute force attack can take a very long time. John also gives the user the ability to utilize a password file. It comes with a password file called password.1st, located in the john directory, with 3546 words in its list.

22. To view the first 20 lines of the file, type the following command in the terminal:
 root@bt:/pentest/passwords/john# **head -n 20 password.lst**

```

root@bt:/pentest/passwords/john# head -n 20 password.lst
#!/comment: This list has been compiled by Solar Designer of Openwall Project,
#!/comment: http://www.openwall.com/wordlists/
#!/comment:
#!/comment: This list is based on passwords most commonly seen on a set of Unix
#!/comment: systems in mid-1990's, sorted for decreasing number of occurrences
#!/comment: (that is, more common passwords are listed first). It has been
#!/comment: revised to also include common website passwords from public lists
#!/comment: of "top N passwords" from major community website compromises that
#!/comment: occurred in 2006 through 2010.
#!/comment:
#!/comment: Last update: 2011/11/20 (3546 entries)
123456
12345
password
password1
123456789
12345678
1234567890
abc123
computer

```

Figure 25: The passwd.lst file

If any account's passwords are changed, john will go through the cracking process again. We will set chewbacca's password to a word contained within the password.lst file.

23. Set chewbacca's password to computer by typing "**computer**" twice after typing:
 root@bt:/pentest/passwords/john# **passwd chewbacca**

```

root@bt:/pentest/passwords/john# passwd chewbacca
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

```

Figure 26: Setting a Password for the User

24. Type the following to run john again and the new password hash will be loaded.
 root@bt:/pentest/passwords/john# **./john /etc/shadow --wordlist=password.lst**

```

root@bt:/pentest/passwords/john# ./john /etc/shadow --wordlist=password.lst
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 4 password hashes with 4 different salts (sha512crypt [32/32])
Remaining 1 password hash
computer (chewbacca)
guesses: 1 time: 0:00:00:00 DONE (Fri Jan 18 23:16:11 2013) c/s: 112 trying: computer
Use the "--show" option to display all of the cracked passwords reliably

```

Figure 27: Using a Dictionary with John

Since that word was one of the first few in the dictionary, john was able to crack the password in less than one second. Now we will try one of the last passwords in the list.

25. To view the last 20 lines of the file, type the following command in the terminal:
 root@bt:/pentest/passwords/john# **tail password.lst**

```
root@bt:/pentest/passwords/john# tail password.lst
1701d
@#$%^&
Qwert
allo
dirk
go
newcourt
nite
notused
sss
```

Figure 28: The passwd.1st file

26. Set chewbacca's password to newcourt by typing "newcourt" twice after typing:
 root@bt:/pentest/passwords/john# **passwd chewbacca**

```
root@bt:/pentest/passwords/john# passwd chewbacca
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Figure 29: Setting a Password for the User

27. Type the following to run john again and the new password hash will be loaded:
 root@bt:/pentest/passwords/john# **./john/etc/shadow --wordlist=password.lst**

You can hit Enter during the cracking process to see which word is being tested.

```
root@bt:/pentest/passwords/john# ./john/etc/shadow --wordlist=password.lst
Warning: detected hash type "sha512crypt", but the string is also recognized as "crypt"
Use the "--format=crypt" option to force loading these as that type instead
Loaded 4 password hashes with 4 different salts (sha512crypt [32/32])
Remaining 1 password hash
guesses: 0 time: 0:00:00:01 14.59% (ETA: Fri Jan 18 23:21:12 2013) c/s: 272 trying: gambit
guesses: 0 time: 0:00:00:02 22.60% (ETA: Fri Jan 18 23:21:14 2013) c/s: 277 trying: cooking
guesses: 0 time: 0:00:00:07 62.33% (ETA: Fri Jan 18 23:21:17 2013) c/s: 277 trying: tatiana
newcourt (chewbacca)
guesses: 1 time: 0:00:00:12 DONE (Fri Jan 18 23:21:18 2013) c/s: 277 trying: newcourt
Use the "--show" option to display all of the cracked passwords reliably
```

Figure 30: The Cracked Passwords

For the final step, we will use John the Ripper to crack the passwords from a shadow file.

28. To view the shadow file from another system stored on your computer, type:
 root@bt:/pentest/passwords/john# **tail /root/lab8/shadow**

```
root@bt:/pentest/passwords/john# tail /root/lab8/shadow
gdm:!!:13940:0:99999:7:::
canna:!!:13940:0:99999:7:::
desktop:!!:13940:0:99999:7:::
administrator:$1$JhRI1KXh$yGxAM4Aorp7a/IoBH5vD70:13940:0:99999:7:::
jesse*:14774:::
bart:$1$hIzbbM6p$0YoxQHV1NuR9z1Ua/d2Rv0:15693:0:99999:7:::
lisa:$1$lyUBtRdH$zjJMpfeHE1eDu/sTSDXRp1:15693:0:99999:7:::
cartman:$1$95Fv9eqd$9.PImLhM.6J2PchjHFtmI/:15693:0:99999:7:::
stan:$1$VJdabstG$NfPALW9TfrXOV4t8GqGKx/:15693:0:99999:7:::
bigbird:$1$3KvlwD80$CUeDVogpl4zjtYBqga0L30:15693:0:99999:7:::
```

Figure 31: The passwd file from another system

29. To crack the passwords from the shadow file within you lab8 folder, type:
 root@bt:/pentest/passwords/john# **./john /root/lab8/shadow**

```
root@bt:/pentest/passwords/john# ./john /root/lab8/shadow
Loaded 7 password hashes with 7 different salts (FreeBSD MD5 [128/128 SSE2 intrinsics 4x])
password (administrator)
password (root)
yellow (bigbird)
boy (stan)
boy (bart)
eat (cartman)
girl (lisa)
guesses: 7 time: 0:00:02:41 DONE (Fri Jan 18 23:29:58 2013) c/s: 20070 trying: girl -
```

Figure 32: The Revealed Passwords

30. To view the password hashes and corresponding passwords, type the following:
 root@bt:~/pentest/passwords/john# **cat john.pot**

```
root@bt:/pentest/passwords/john# cat john.pot
$6$EVCJcswz$gs9NsKYLw.gIRFQBfcmPho4xcxSo5AIPQYqNnd9gH4n2jokGqD7T5caWQUQ3ZZsr3wt.hvn1PbcIkshVa4LTa.:toor
$6$S3IeZf2j$glWIfxoZkINVcvUjLxdI2YlOC1NS.d8hS.UCgrVns3MuDrD6GVYajdx8xr4zzdnkYhdSD0rCGLCR9r/1mlph60.:green
$6$MeJxZyv4$IEg.BegR/elIQRYATU6051W15Yfk7yZFr6.3t4CbONMGMAqHsjQt3BMTH7aKadjYK/zupr5oQKytEhB0ReZkP.:green
$6$uFnJF8V3$YFTyaoBvVvk8NQb9zfEVeBcwKZ96Mrj1DF6V2uGM6lmHF7HgZr.zk87LyaNu2MiXA2fvRKjgrJOZiQgG7VcyHT0:hacker
$6$wmX0Mw4y$WmILWrzW9vfkMw6FANH6NBfojC..LSeZPYWTG.A3HeJzPx2dEB.x1qW0Lm6nDwsduiI630de9hZKaMrr2iUM1:computer
$6$Y.v6x6aS0p.ZV3FWcR3s/W0NhJgbKT1YiKa0/HBYWgGpZCbEcjzWmbCPi5Jh.iqTDkAMJvFxcUrGZKmBitn0j94fn45Q0:newcourt
$1$JhRI1KXh$yGxAM4Aorp7a/IoBH5vD70:password
$1$Ntv5zRU$QIn9YyqKdbYtusWmwzB6v/:password
$1$3KvlwD80$CUeDVogpl4zjtYBqga0L30:yellow
$1$VJdabstG$NfPALW9TfrXOV4t8GqGKx/:boy
$1$hIzbbM6p$0YoxQHV1NuR9z1Ua/d2Rv0:boy
$1$95Fv9eqd$9.PImLhM.6J2PchjHFtmI/:eat
$1$lyUBtRdH$zjJMpfeHE1eDu/sTSDXRp1:girl
```

Figure 33: Viewing the Passwords

1.2 Conclusion

In Linux, the names of the user accounts are listed in the `/etc/passwd` file. The hashes for the user's passwords are stored in the shadow file. The password hashes are salted, which means if you give two users the same exact password, a different hash will be displayed for each. When salting is done, you will be unable to perform a rainbow table attack. Instead, you will need to perform a dictionary or brute force attack. John the Ripper is a password cracker that allows an attacker to use brute force or a dictionary file to try to find the password for the hash. All cracked passwords and their corresponding hashes will be stored in the `john.pot` file. Any account changes are recorded in the `auth.log` file.

1.3 Discussion Questions

1. What is stored within the shadow file?
2. When a user is created on a Linux Ubuntu system, which file has a record of it?
3. What is the command to change a user's password on a Linux system?
4. Where does John the Ripper store the passwords after they are cracked?

2 Creating an Additional Account with root Level Permissions

On Microsoft Windows operating systems, you can create multiple accounts with administrative rights. On a Linux system, there is usually only one root account. However, if another account is created with a UID of 0, that account will have root level permissions. We will modify the `/etc/passwd` and `/etc/shadow` to create an account that is 'equivalent' to root. In order to do this, you would need the password for the root account, physical access, or a local exploit that would elevate your permissions to root.

2.1 Creating another 'root'

If you are an attacker, and are able to get root level permissions, you can create another account with the same level of permissions by modifying the `passwd` and `shadow` files.

1. Type the following command to open the `passwd` file located in the `/etc` folder:
`root@bt:~# gedit /etc/passwd`

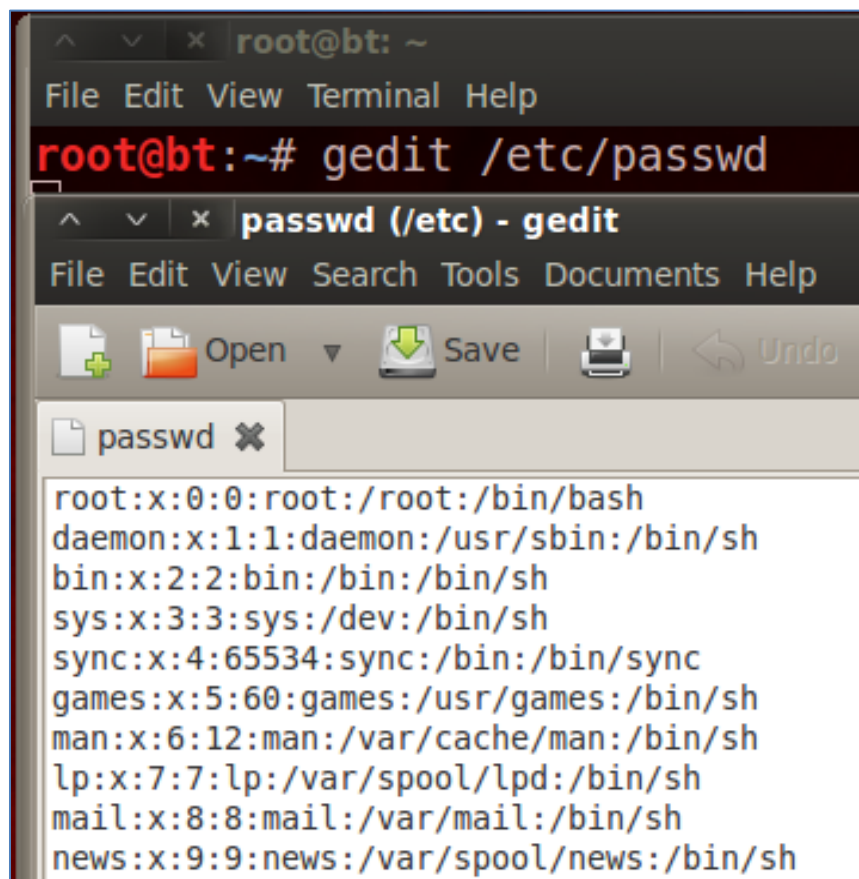


Figure 34: Editing the `passwd` file

- Copy the first line of the file. Go down in front of the "d" in daemon and hit enter. Go up one line to the blank line and paste the root account info into the 2nd line.

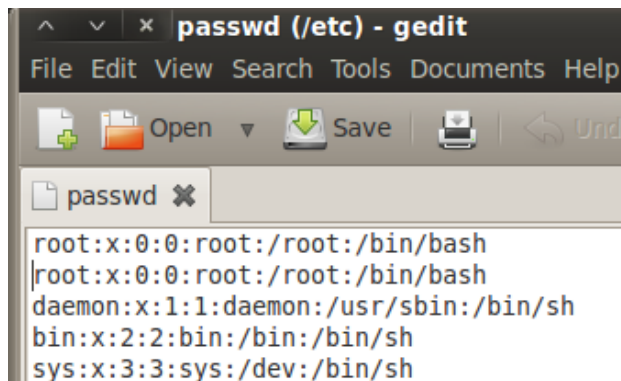


Figure 35: Editing the passwd file

- Change the name on the second line from root to vader. **Save** and close the file.

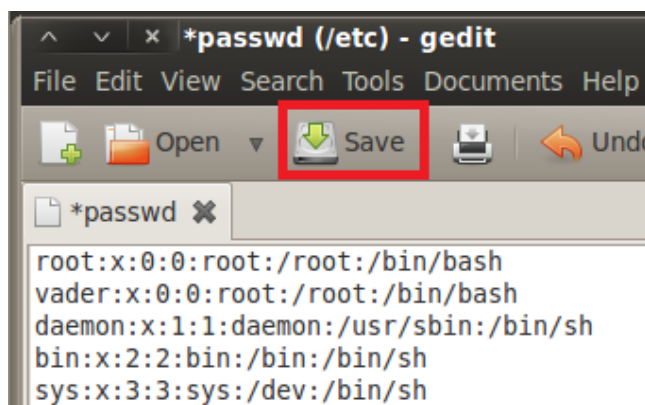


Figure 36: Saving the Edited File

- Type the following command to open the passwd file located in the /etc folder.
root@bt:~# **gedit /etc/shadow**

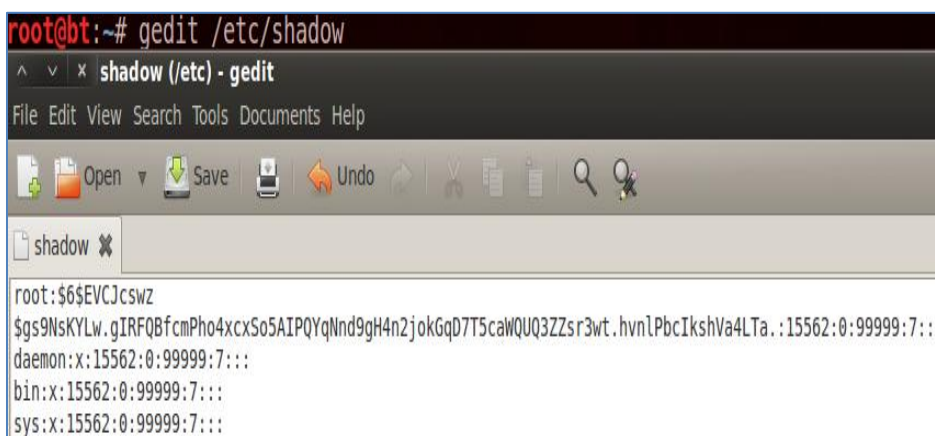


Figure 37: Editing the shadow file

- Put your cursor in front of the "d" in daemon. Right-click on the first two lines of the file and click **copy**. Go down in front of d in daemon, right-click and **paste**.

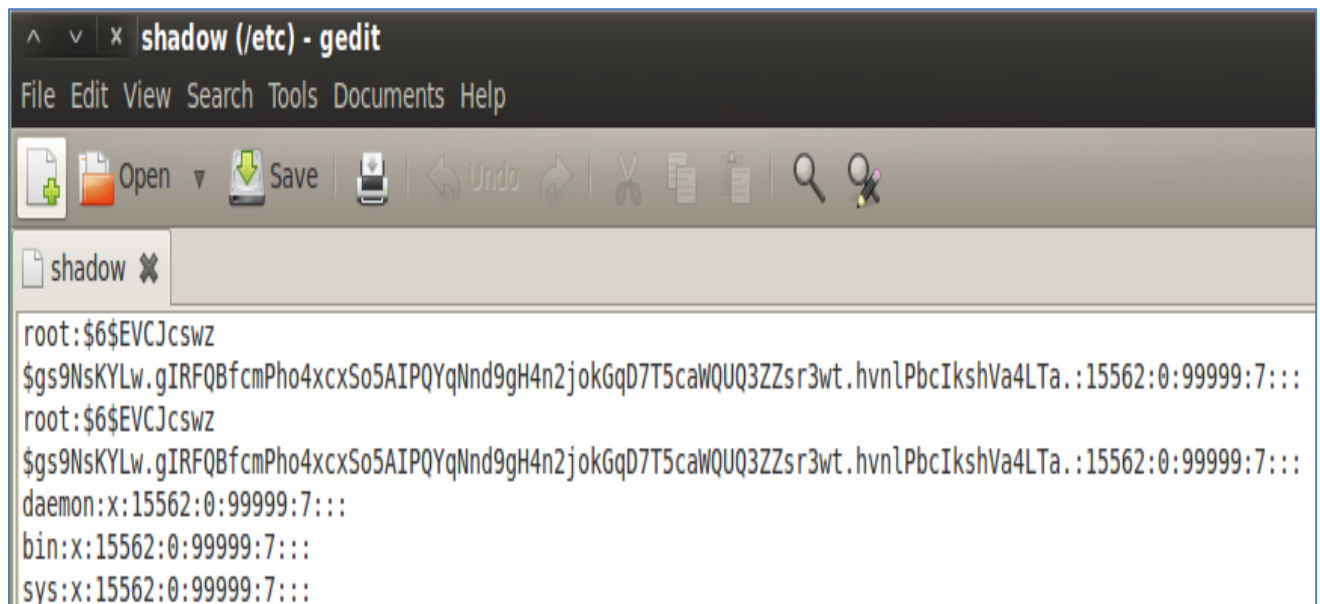


Figure 38: Editing the shadow file

- Change the name on the third line from root to vader. **Save** and close the file.

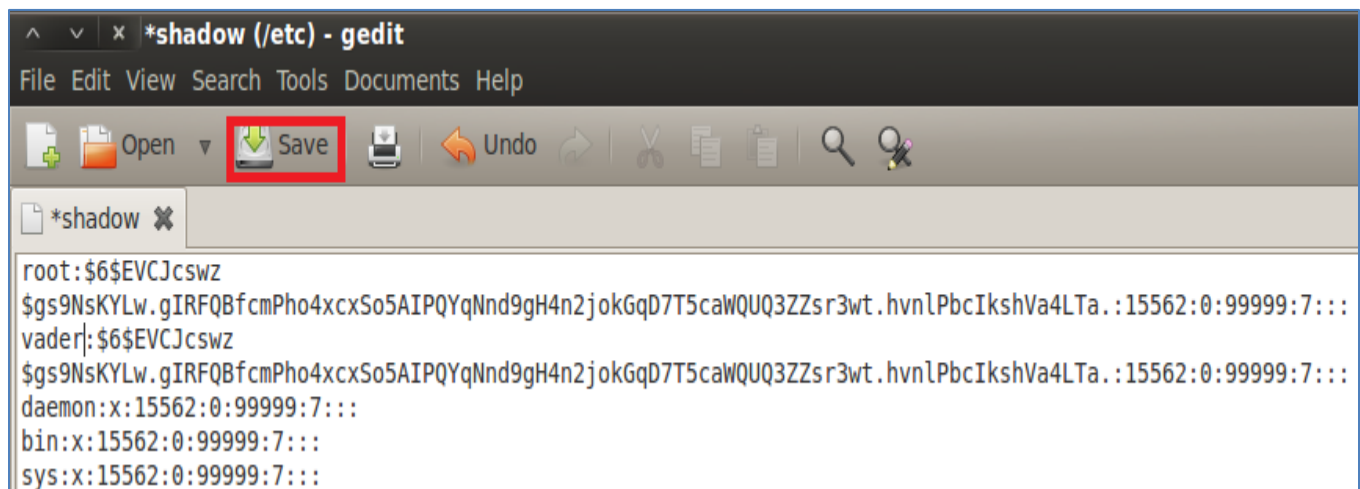


Figure 39: Saving the Shadow file

When we utilized the `useradd` and `passwd` commands during the first task, the commands triggered events in `auth.log`. In this task, however, a user named `vader` was created with the password of `toor`, by editing the `passwd` and `shadow` files.

7. Type the following to see if there is evidence of the vader account in auth.log.
root@bt:~# tail /var/log/auth.log

```
root@bt:~# tail /var/log/auth.log
Jan 19 12:55:08 bt login[1511]: pam_sm_authenticate: username = [root]
Jan 19 12:55:08 bt login[1511]: pam_unix(login:session): session opened for user root by LOGIN(uid=0)
Jan 19 12:55:08 bt login[1543]: ROOT LOGIN on '/dev/tty1'
Jan 19 12:55:26 bt polkitd(authority=local): Registered Authentication Agent for session /org/freedesktop/Con
soleKit/Session3 (system bus name :1.12 [/usr/lib/policykit-1-gnome/polkit-gnome-authentication-agent-1], obj
ect path /org/gnome/PolicyKit1/AuthenticationAgent, locale en_US.UTF-8)
Jan 19 13:09:02 bt CRON[1897]: pam_unix(cron:session): session opened for user root by (uid=0)
Jan 19 13:09:02 bt CRON[1897]: pam_unix(cron:session): session closed for user root
Jan 19 13:17:01 bt CRON[1962]: pam_unix(cron:session): session opened for user root by (uid=0)
Jan 19 13:17:01 bt CRON[1962]: pam_unix(cron:session): session closed for user root
Jan 19 13:39:01 bt CRON[2058]: pam_unix(cron:session): session opened for user root by (uid=0)
Jan 19 13:39:01 bt CRON[2058]: pam_unix(cron:session): session closed for user root
```

Figure 40: The auth.log file

8. To verify that there is no evidence of the vader account in auth.log, type:
root@bt:~# tail /var/log/auth.log | grep vader

```
root@bt:~# tail /var/log/auth.log | grep vader
root@bt:~#
```

Figure 41: Parsing for Vader in auth.log

The reason that there is no evidence of the vader account being created or given a password is because the /etc/passwd and /etc/shadow files were manually edited. Another trick we used was the placement of the account. When the yoda and chewbacca accounts were created, they were added to the bottom of the /etc/passwd and /etc/shadow files. When new accounts are added, that is the location they are placed.

9. Type the following command to display the last accounts created.
root@bt:~# tail /etc/passwd

```
root@bt:~# tail /etc/passwd
statd:x:108:65534:./var/lib/nfs:/bin/false
usbmux:x:109:46:./home/usbmux:/bin/false
pulse:x:110:116:./var/run/pulse:/bin/false
rtkit:x:111:117:./proc:/bin/false
festival:x:112:29:./home/festival:/bin/false
postgres:x:1000:1000:./home/postgres:/bin/sh
ftp:x:113:121:ftp daemon,,./srv/ftp:/bin/false
hax0r:x:1001:1001:./home/hax0r:/bin/sh
yoda:x:1002:1002:./home/yoda:/bin/sh
chewbacca:x:1003:1003:./home/chewbacca:/bin/sh
```

Figure 42: The last ten lines of the passwd file

By storing the account in the beginning of the file, we make it a lot less likely it will be discovered. For example, when you cat the file you will only see the bottom of it. Also, we wanted to avoid putting our account at the top of the list where root goes.

10. Type the following command to display the last accounts created.

root@bt:~# head /etc/passwd

```
root@bt:~# head /etc/passwd
root:x:0:0:root:/root:/bin/bash
vader:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

Figure 43: The last ten lines of the passwd file

To verify that the vader account works, we will ssh from Windows 7 to BackTrack.

11. To generate the keys that will be needed for an SSH connection, type:

root@bt:~# sshd-generate

```
root@bt:~# sshd-generate
Generating public/private rsa1 key pair.
Your identification has been saved in /etc/ssh/ssh_host_key.
Your public key has been saved in /etc/ssh/ssh_host_key.pub.
The key fingerprint is:
3c:12:ee:f7:6f:73:0b:1b:df:42:b4:ab:7b:9a:09:f6 root@bt
The key's randomart image is:
+--[RSA1 2048]-----+
|
|      .
|    . o
|   o S
|  . . .
| . . . o
|  . . o o .
|   o ooB=.
|  .oEBoo.
|
+-----+

```

Figure 44: SSH Key Generate

12. To start the SSH server on the **BackTrack 5** *external* machine, type:
`root@bt:~# /etc/init.d/ssh start`

```
root@bt:~# /etc/init.d/ssh start
Rather than invoking init scripts through /etc/init.d, use the service(8)
utility, e.g. service ssh start

Since the script you are attempting to invoke has been converted to an
Upstart job, you may also use the start(8) utility, e.g. start ssh
ssh start/running, process 2522
```

Figure 45: Starting SSH

13. To verify that the SSH server service is running on the machine, type:
`netstat -tan | grep 22`

```
root@bt:~# netstat -tan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:7337          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN
tcp6       0      0 :::7337                :::*                    LISTEN
tcp6       0      0 :::22                  :::*                    LISTEN
```

Figure 46: Verifying SSH is running

14. On the **Windows 7** system, login to the student account with the password of **password**. Double-click on the **putty.exe** file on the desktop.

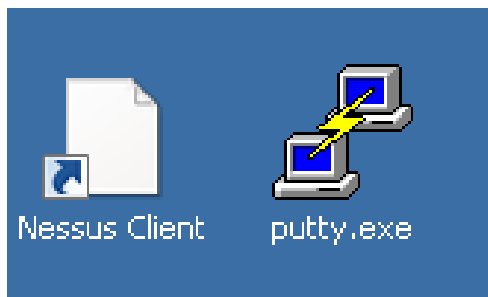


Figure 47: Putty File

15. In the Host Name Box, type the IP address of **216.6.1.100**. Click **Open**.

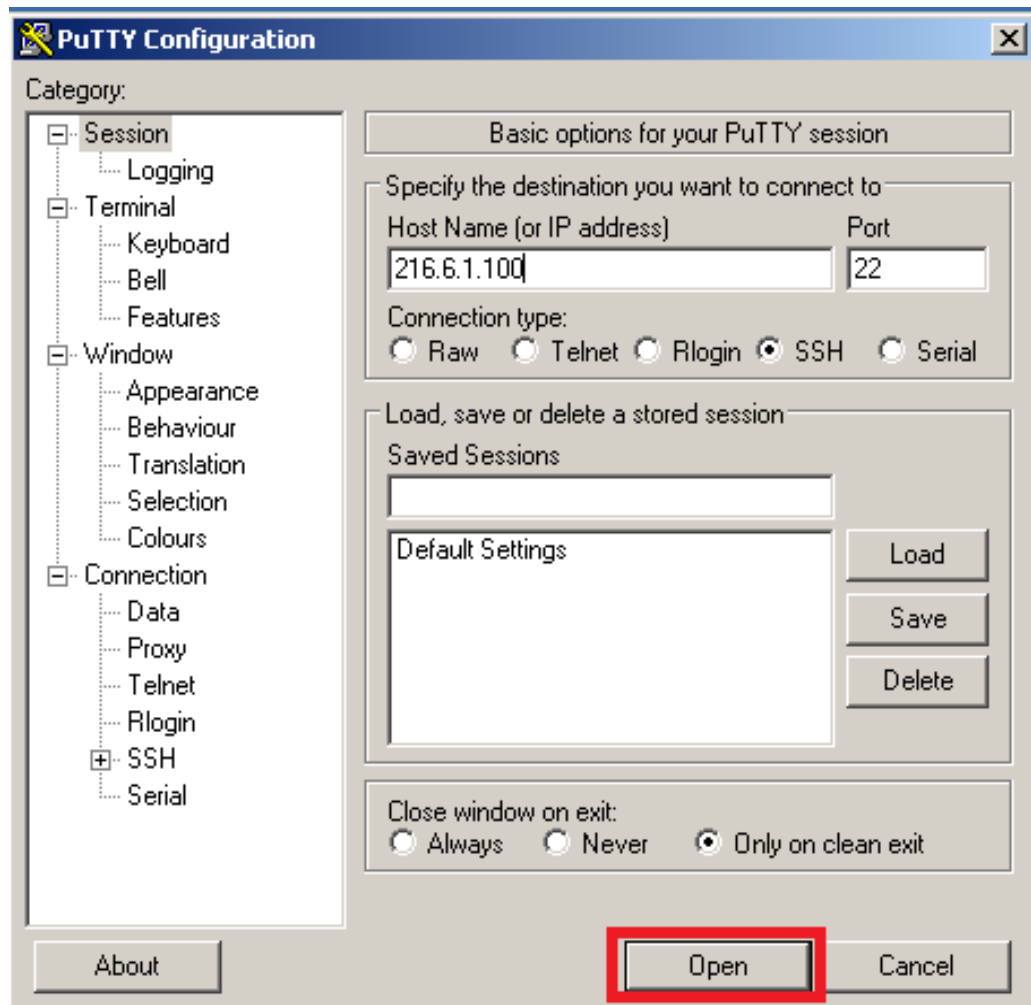


Figure 48: Connecting to the Server

When you first connect to a new SSH host, you will be warned about the connection.

16. Click **Yes** to the PuTTY security alert because you trust this host.

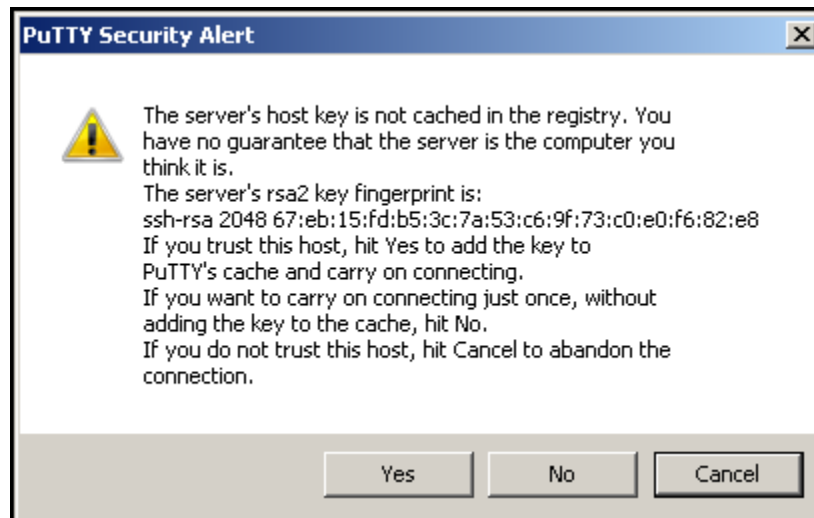


Figure 49: PuTTY Security Alert

Now, we will attempt to login with our vader account which has a UID of 0.

17. You should receive as **login as:** prompt. For the username, type **vader**.

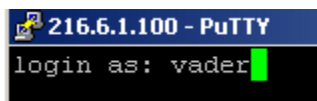


Figure 50: Login as vader

The vader account has the same password as the root account, because we copied the password hash, and other information, when we edited the passwd and the shadow file.

18. When you are asked for **vader@216.6.1.100's** password, type toor.

For security reasons, the password will not be displayed when you are typing it.

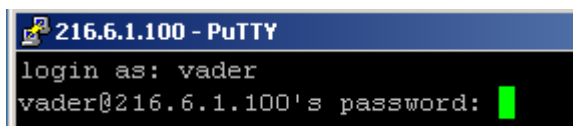


Figure 51: Providing the Password

You should be able to get in successfully and see a Last Login message.


```

216.6.1.100 - PuTTY
login as: vader
vader@216.6.1.100's password:
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux

System information as of Sat Jan 19 19:27:23 EST 2013

System load:  0.09          Processes:            111
Usage of /:   57.9% of 19.06GB Users logged in:       1
Memory usage: 11%          IP address for eth0: 216.6.1.100
Swap usage:   0%

=> There is 1 zombie process.

Graph this data and manage this system at https://landscape.canonical.com/
Last login: Sat Jan 19 19:23:42 2013
root@bt:~#

```

Figure 52: Logging in a Vader

We were able to log in as our account with root level permissions, but we need to verify that we can perform tasks that only the root account is allowed to do on the system.

19. Look at the permissions required to view the shadow file by typing:

```
root@bt:~# ls -l /etc/shadow
```

```

root@bt:~# ls -l /etc/shadow
-rw-r----- 1 root shadow 1298 2013-01-19 19:26 /etc/shadow

```

Figure 53: Listing the Shadow File Permissions

20. To prove that you actually do have root level access, type the following:

```
root@bt:~# head /etc/shadow
```

```

root@bt:~# head /etc/shadow
root:$6$EVCJcsWz$gs9NsKYLw.gIRFQBfcmPho4xcxSo5AIPQYqNnd9gH4n2jokGqD7T5caWQUQ3ZZs
r3wt.hvnlPbcIkshVa4LTa.:15562:0:99999:7:::
vader:$6$EVCJcsWz$gs9NsKYLw.gIRFQBfcmPho4xcxSo5AIPQYqNnd9gH4n2jokGqD7T5caWQUQ3ZZ
sr3wt.hvnlPbcIkshVa4LTa.:15562:0:99999:7:::
daemon:x:15562:0:99999:7:::
bin:x:15562:0:99999:7:::
sys:x:15562:0:99999:7:::
sync:x:15562:0:99999:7:::
games:x:15562:0:99999:7:::
man:x:15562:0:99999:7:::
lp:x:15562:0:99999:7:::
mail:x:15562:0:99999:7:::

```

Figure 54: Viewing the Shadow File

21. Even though there was no evidence of the vader account in auth.log when the account was created by editing the passwd and shadow files, there will now be evidence of the account because of the ssh login. To view the auth.log file, type:
root@bt:~# **tail /var/log/auth.log**



```
root@bt:~# tail /var/log/auth.log
Jan 19 19:24:59 bt useradd[1834]: new group: name=yoda, GID=1002
Jan 19 19:24:59 bt useradd[1834]: new user: name=yoda, UID=1002, GID=1002, home=/home/yoda, shell=/bin/sh
Jan 19 19:25:01 bt useradd[1839]: new group: name=chewbacca, GID=1003
Jan 19 19:25:01 bt useradd[1839]: new user: name=chewbacca, UID=1003, GID=1003, home=/home/chewbacca, shell=/bin/sh
Jan 19 19:27:05 bt sshd[1883]: Server listening on 0.0.0.0 port 22.
Jan 19 19:27:05 bt sshd[1883]: Server listening on :: port 22.
Jan 19 19:27:23 bt sshd[1885]: pam_sm_authenticate: Called
Jan 19 19:27:23 bt sshd[1885]: pam_sm_authenticate: username = [vader]
Jan 19 19:27:23 bt sshd[1885]: Accepted password for vader from 216.5.1.200 port 1061 ssh2
Jan 19 19:27:23 bt sshd[1885]: pam_unix(sshd:session): session opened for user vader by (uid=0)
```

Figure 55: Evidence of vader

2.2 Conclusion

Although the useradd command can be used to create users, it leaves evidence in the auth.log file. When a user is created by manually editing the passwd and shadow files, a log entry is not created in auth.log. If a user created is given a User ID, or UID of zero, then that user account will have root level permissions on the system.

2.3 Discussion Questions

1. Which account has a User ID (UID) of zero?
2. When the useradd command is utilized, which file has a record of the event?
3. When the passwd command is utilized, which file has a record of the event?
4. Which file has a record of incoming SSH connections?

3 Using the SSH Keys to Break into Linux

On the *External BackTrack 5* system, you generated SSH keys and started the SSH server. If we provide a user with the SSH keys, they will be able to log in remotely without a username or password. This will be the case even if the user changes their root password. In this scenario, the *external Backtrack 5* machine will act as the victim machine.

3.1 SSH Keys

1. On the *internal BackTrack 5* machine, type the following to generate an SSH key:
root@bt:~# ssh-keygen

Hit enter **3** times to accept the defaults for all of the questions asked.

```

root@bt:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
e6:ca:de:3d:48:ab:b9:e0:9b:ac:b4:ae:e1:08:ee:8c root@bt
The key's randomart image is:
+--[ RSA 2048 ]-----+
  << back | track
    S
    0.
  o . . . .0
  Bo + + +0..
  EB+.=oBo. .
+-----+
root@bt:~#

```

Figure 56: Generating the Key

Use SCP (secure copy) to copy the public key to the `authorized_keys` file on 216.6.1.100:

2. Type the following to copy the key to the `authorized_keys` file on 216.6.1.100:
`root@bt:~# scp /root/.ssh/id_rsa.pub vader@216.6.1.100:/root/.ssh/authorized_keys`

Type yes to the question “Are you sure you want to continue connecting?”

```
root@bt:~# scp /root/.ssh/id_rsa.pub vader@216.6.1.100:/root/.ssh/authorized_keys
The authenticity of host '216.6.1.100 (216.6.1.100)' can't be established.
RSA key fingerprint is 7d:95:6a:62:d9:6e:f1:a1:24:43:83:46:0e:41:fb:18.
Are you sure you want to continue connecting (yes/no)? yes
```

Figure 57: SCP to the Remote Host

3. When you are asked for vader@216.6.1.100's password, type **toor**. For security reasons, the password will not be displayed when you are typing it.

```
Warning: Permanently added '216.6.1.100' (RSA) to the list of known hosts.
vader@216.6.1.100's password:
id_rsa.pub 100% 389 0.4KB/s 00:00
root@bt:~#
```

Figure 58: 100% of the File Copied

Now, you can ssh to the 216.6.1.100 machine without a username or password:

4. Type the following to SSH to the remote Linux machine without authentication.
`root@bt:~# ssh 216.6.1.100`

```
root@bt:~# ssh 216.6.1.100
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux

System information as of Sat Jan 19 23:49:43 EST 2013

System load: 0.0          Processes:              111
Usage of /:  57.9% of 19.06GB Users logged in:           1
Memory usage: 14%         IP address for eth0: 216.6.1.100
Swap usage:  0%

=> There is 1 zombie process.

Graph this data and manage this system at https://landscape.canonical.com/
Last login: Sat Jan 19 23:34:38 2013 from 216.1.1.1
root@bt:~#
```

Figure 59: SSH Without Authentication

5. Type the following command to leave the SSH session to 216.6.1.100.
root@bt:~# **exit**

```
root@bt:~# exit
logout
Connection to 216.6.1.100 closed.
root@bt:~#
```

Figure 60: Exiting

6. Change the root password on the *External BackTrack 5* machine by typing:
root@bt:~# **passwd root**

```
root@bt:~# passwd root
Enter new UNIX password:
```

Figure 61: Changing the root password on 216.6.1.100

For the password, type "123" twice. For security reasons, it will not be displayed.

7. Type the following to SSH to the remote Linux machine without authentication.
root@bt:~# **ssh 216.6.1.100**

```
Connection to 216.6.1.100 closed.
root@bt:~# ssh 216.6.1.100
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux

System information as of Sat Jan 19 23:56:02 EST 2013

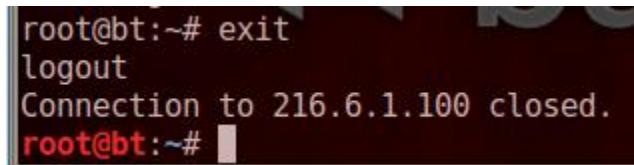
System load: 0.01          Processes:          111
Usage of /:  57.9% of 19.06GB Users logged in:      1
Memory usage: 14%          IP address for eth0: 216.6.1.100
Swap usage:  0%

=> There is 1 zombie process.

Graph this data and manage this system at https://landscape.canonical.com/
Last login: Sat Jan 19 23:49:43 2013 from 216.1.1.1
root@bt:~#
```

Figure 60: SSH Without Authentication Again

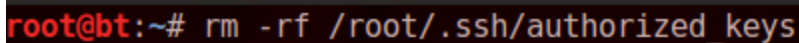
8. Type the following command to leave the SSH session to 216.6.1.100.
root@bt:~# **exit**



```
root@bt:~# exit
logout
Connection to 216.6.1.100 closed.
root@bt:~#
```

Figure 63: Exiting

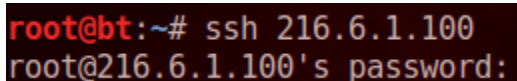
9. On the *External BackTrack 5* machine, type the following command:
root@bt:~# **rm -rf /root/.ssh/authorized_keys**



```
root@bt:~# rm -rf /root/.ssh/authorized_keys
```

Figure 64: Removing the Authorized Keys

10. Type the following to SSH to the remote Linux machine. The `authorized_key` file was deleted from the 216.6.1.100 server, so you will be asked for a password.
root@bt:~# **ssh 216.6.1.100**



```
root@bt:~# ssh 216.6.1.100
root@216.6.1.100's password:
```

Figure 65: SSH to 216.6.1.100

3.2 Conclusion

If a user stores their public key on in the `authorized_keys` file on a remote SSH server, they will be able to connect to the system without providing authentication. Even if the root password is changed on the SSH server, the user will still be able to connect to the SSH server as long as their key resides in the `authorized_keys` file.

3.3 Discussion Questions

- 1 In what directory is the `authorized_keys` file stored?
- 2 How does a user generate SSH keys in Linux?
- 3 What effect will changing the root password have on an authorized key?
- 4 How do you keep someone out who has an authorized key?

References

1. John the Ripper Password Cracker
<http://www.openwall.com/john/>
2. Understanding /etc/shadow file
<http://www.cyberciti.biz/faq/understanding-etcshadow-file/>
3. Authorized Keys File
<http://www.dzone.com/snippets/quickly-add-your-public-key>
4. SSH/OpenSSH/Configuring
<https://help.ubuntu.com/community/SSH/OpenSSH/Configuring>
5. HOWTO: set up ssh keys
<http://paulkeck.com/ssh/>