Updated: Oct 1, 2022

# Lab 6.1 Cracking Linux Passwords with JtR and Hashcat

> 💡Ok, we've exhausted our password guessing tricks and now we need to see what we can do to derive passwords from hashes that we have recovered from an exploited system. There are several tools we can use. This lab will continue our adventures against bios.shire.local (10.0.5.21).

Figure out how to download the seclists package to kali, this may take some time. Investigate /usr/share/seclists and spend some time exploring the Passwords/Common-Credentials sub-directory. Trying some shorter wordlists is probably a good idea before passing in the entire rockyou.txt file to a tool.

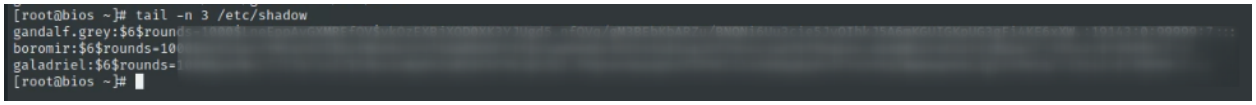
Now, one of the password guessing victims from last week has a bit more privilege on bios.shire.local than the other users. If you've not cracked that user, do so (you may ask a neighbor what wordlist they used). Leverage that account to become root on 10.0.5.21. When you login as a user, it is a good idea to see what groups they are in. Please don't perform any destructive actions on bios or change any passwords.

> 💡You can do this with the `id` command. On RHEL, member's of the wheel group are a good bet while on debian, members of the sudo group have elevated privileges.

dump the last 3 entries in /etc/passwd file to the terminal using tail

```
[root@bios ~]# tail -n 3 /etc/passwd
gandalf.grey:x:1005:1005::/home/gandalf.grey:/bin/bash
boromir:x:1006:1006::/home/boromir:/bin/bash
galadriel:x:1007:1007::/home/galadriel:/bin/bash
[root@bios ~]#
```

As root, dump the last 3 entries in the /etc/shadow file to the terminal using tail.




`Deliverable 1. Provide screenshots similar to the ones above showing the last 3 entries in /etc/passwd and /etc/shadow.`

`Deliverable 2. Research what hashing algorithm is being used on this server, one of the fields in /etc/shadow points to the format. Explain this.`

Copy the two excerpts to a week6 directory on kali and name them **etc_passwd.txt** and **etc_shadow.txt** respectively.

Updated: Oct 1, 2022

Deliverable 3. Examine user Galadriel's shadow entry.

- What is the salt?
- What is the hashed salt+password?

Provide a screenshot that shows each explicitly labeled. Note, you may see a different format between password hashes. Some explicitly indicate the number of "rounds".

Deliverable 4. Figure out how to use the unshadow utility to create a file usable by John the Ripper(JtR) and then crack the unshadowed files hashes using JtR. Provide a screenshot showing your results.

Cracking Passwords in a Virtual Machine can be an exercise in frustration. Generally, the performance of john and hashcat are abysmal. The following link describes how to leverage humpty.cyber.local to run your cracks on a decent physical workstation.

```
┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$ john --wordlist=small.txt unshadow.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 1000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 3 candidates left, minimum 4 needed for performance.
(gandalf.grey)
(boromir)
(galadriel)
3g 0:00:00:00 DONE (2022-10-01 09:33) 300.0g/s 300.0p/s 900.0c/s 900.0C/s gandalfrockyou..BoRomir2000Z
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$
```

💡There is also a way to use just the shadow file with JtR, though you will need to specify the format on the command line. Feel free to research and do this instead of using the unshadow file.

Deliverable 5. Let's see if you can reverse engineer the shadow file using python. We will use peregrin.took's bios hash.
The grayed out area has the plaintext password for gandalf. Provide a screenshot similar to the one below. Use Boromir or Galadriel's shadow entry.

Updated: Oct 1, 2022

```
[root@bios ~]# cat /etc/shadow | grep peregrin
peregrin.took:$6$qIWlkSUTixsoFc/H$kKz7zMyQm6CAtVi9HN0/a5rYY7YtdQsQPR2UUMsm63BmitXGa1f4UaH3tQBLDeQzbmRcWnEW9S59/n5W8FT9z0:19229:0:99999:7:::
[root@bios ~]#
```

```
┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$ cat peregrin.shadow.txt
peregrin.took:$6$qIWlkSUTixsoFc/H$kKz7zMyQm6CAtVi9HN0/a5rYY7YtdQsQPR2UUMsm63BmitXGa1f4UaH3tQBLDeQzbmRcWnEW9S59/n5W8FT9z0:19229:0:99999:7:::

┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$ python3 -c "from passlib.hash import sha512_crypt
print(sha512_crypt.hash('           ',salt='qIWlkSUTixsoFc/H'))"

$6$rounds=656000$qIWlkSUTixsoFc/H$G7R2zlZomGJLp6eUOaH2B5c1gpORkADfzjgtdUTDQZNNW3opiKZieEZULacU1qy9BEgDkJYiQxm9OIWJnMYH7/

┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$
```

To reverse engineer one of the hashes where the rounds are not default, you can try this syntax.

```
┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$ python3 -c "from passlib.hash import sha512_crypt
print(sha512_crypt.hash('           ',rounds=1000,salt='LneEppAvGXMREfOV'))"

$6$rounds=1000$LneEppAvGXMREfOV$vkOzEXBjXOD0XK3YJUgd5.nfQVq/gM3BEbKbARZu/BNQNi6Uu3cie5JvOIhkJ5A6mKGUIGKpUG3gFi4KE6xXW.

┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$ cat unshadow.txt| grep gandalf
gandalf.grey:$6$rounds=1000$LneEppAvGXMREfOV$vkOzEXBjXOD0XK3YJUgd5.nfQVq/gM3BEbKbARZu/BNQNi6Uu3cie5JvOIhkJ5A6mKGUIGKpUG3gFi4KE6xXW.:1005:1005::/home/gandalf.grey:/bin/bash

┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$
```

Hashcat as a tool is arguably better than JtR. The following syntax is suggested

```
File  Actions  Edit  View  Help
┌──(champuser㉿kali)-[~/sec335/tech-journal/SEC335/week6]
└─$ hashcat -m 1800 -a 0 -o cracked.txt unshadowed.txt small.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.
```

cracked hashes will show up in cracked.txt. the -m indicates the method or algorithm that produced the hash. Run hashcat -h, grepping for 1800 and you will see how that works. You will be using hashcat against other types of credentials in future.

`Deliverable 6. crack at least one of the hashes using hashcat and show the result in a screenshot similar to the one below (hit s for status). Again, consider leveraging humpty or your own physical system for this crack.`

```
┌──(champuser㉿kali)-[~/sec335/targets/bios]
└─$ cat cracked.txt
$6$rounds=1000$LneEppAvGXMREfOV$vkOzEXBjXOD0XK3YJUgd5.nfQVq/gM3BEbKbARZu/BNQNi6Uu3cie5JvOIhkJ5A6mKGUIGKpUG3gFi4KE6xXW.:
$6$rounds=1000$poPWvLT/CfA/sxS/$lHbu1oMqRV2aM18fkFPbJw25U2.POqhonSmaUpbzPIPVKl2IxS86Qq8q9v3fYu5Y6qlWwbmqekbL3g1vtPmlQ/:
$6$rounds=1000$UvKLGar/VWtqFGCE$DcfWOzRolV4T6GABOUOFFXfg4lpmD4mKriKX1n5sN3ugJSY3nnicjuGfbT9hgEeo.b6dpWSitnK3z3jjBQ2w//:
```

Updated: Oct 1, 2022

Deliverable 7. Start a text or csv or markdown file similar to the one below. Include your successful guesses from Week 5 as well as the cracks from this week. We will need this data in our future adventures. a listing or screenshot of all your acquired passwords. This type of material is normally called "loot" in hacker parlance. Documenting uncracked hashes is also a great idea. You may have better luck cracking them as you learn more about your target or decide to crack on a real workstation instead of a kali vm.

| user | password | service |
|---|---|---|
| samwise | [redacted] | httpd |
| samwise.gamgee | [redacted] | ssh |
| bilbo | [redacted] | httpd |
| bilbo.baggins | [redacted] | ssh |
| pippin | [redacted] | httpd |
| peregrin.took (wheel user) | [redacted] | ssh |
| frodo | [redacted] | httpd |
| frodo.baggins | [redacted] | ssh |
| NOTE, THESE ARE FOR USE AFTER First Crack | | |
| gandalf.grey | [redacted] | ssh |
| boromir | [redacted] | ssh |
| galadriel | [redacted] | ssh |

Updated: Oct 1, 2022

Deliverable 8. Develop your own password cracking content page within your tech-journal or extend the password guessing content already created. You can include both tools in this page or create a page per tool. Provide associated links.
Document the following:

- How to grab password hashes, Can you extend the example to grab only those shadow accounts that have a hash? Some lines don't even have a hash.
- The format of the shadow file, with emphasis on username, algorithm, salt and hash.
- the use of unshadow
- cracking with john
- cracking with hashcat
- Make sure to understand how the algorithm within the shadow file relates with the flags you may need to pass to the program.Find a good reference that relates the code in the shadow file ($6$ or other) to the algorithm. Link to that.

Deliverable 9. As always, reflect on this lab and any challenges or areas you are not clear on. What do you think about password generators and managers now?

For a challenge, consider adding a test user on kali (with an easy password and in a small password list) and use kali to try to crack the $y$ format. I had no luck with hashcat but was able to do so with JtR (you will want to remove this user before ever enabling ssh on your kali system).