Crack Shadow Hashes After Getting Root

I use system basic 2 on retake exam belt to do this lab

Passwords are stored in a one-way encryption called hashes. There are multiple ways of obtaining these hashes, such as .dll injection in Windows systems or capturing the hash in transit, such as in WPA2 wireless cracking.

Once we can grab the hash, the next step becomes one of finding an effective and efficient way of cracking it. There are numerous tools, some of which I have highlighted in other articles here, but hashcat is unique in its design and versatility, so let's take a look at how it works. By Crack Shadow Hashes After Getting Root on a Linux System

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.

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:

```
kay:x:1000:1000:Kay,,,:/home/kay:/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:

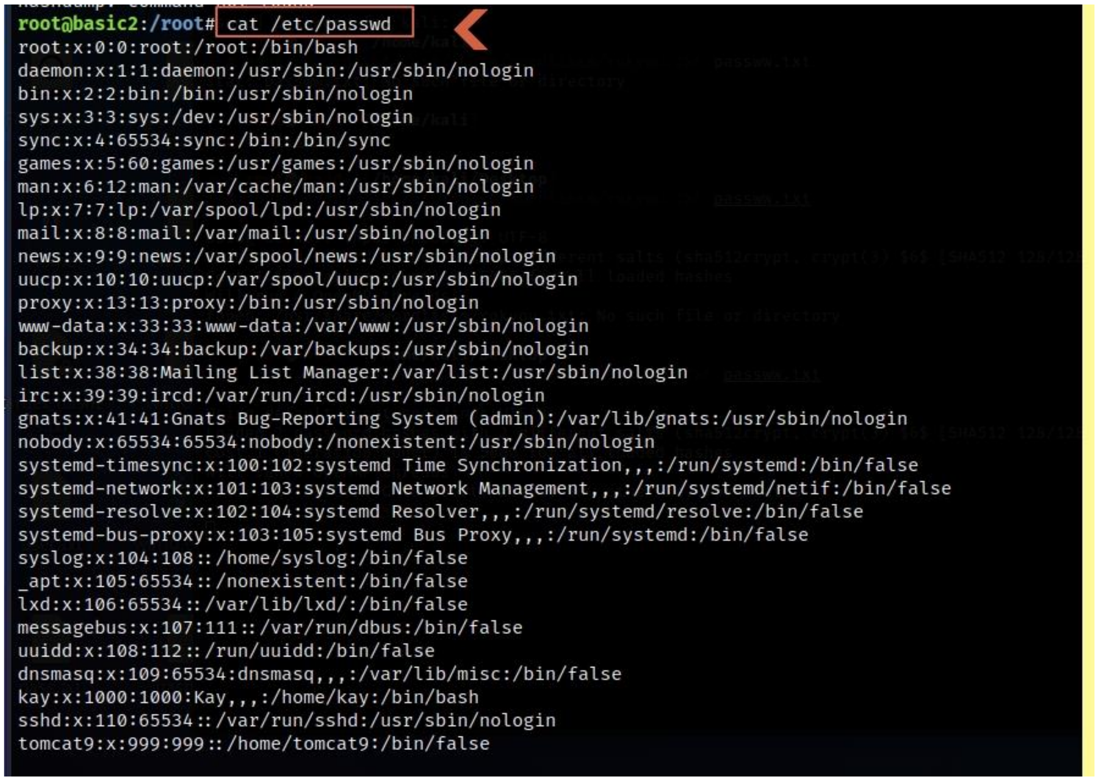
```
kay:$6$ON8Wi9Ow$Puwzhgbc2chaNEqWFO/UVH2yJ5zVb3WirwtCxQ5ssr2OEMAuYCrHscUNe.KPUhH6N
D4CYx9WWu449W3mrzVtk/:17644:0:99999:7:::
```

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.

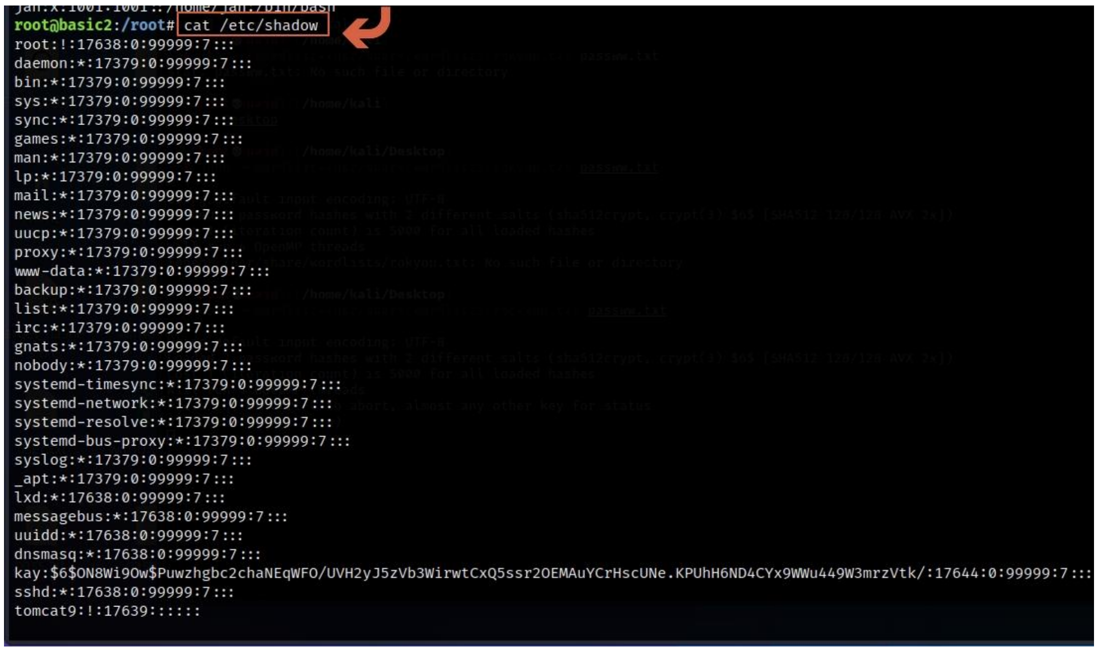
Lab 1 Linux System

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.



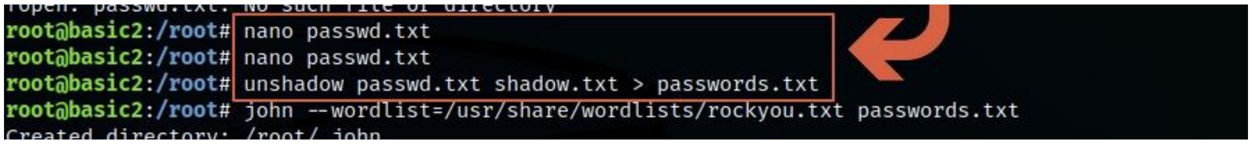
```
root@basic2:/root# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
lxd:x:106:65534::/var/lib/lxd/:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
uuidd:x:108:112::/run/uuidd:/bin/false
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
kay:x:1000:1000:Kay,,,:/home/kay:/bin/bash
sshd:x:110:65534::/var/run/sshd:/usr/sbin/nologin
tomcat9:x:999:999::/home/tomcat9:/bin/false
```



```
jan:x:1001:1001::/home/jan:/bin/bash
root@basic2:/root# cat /etc/shadow
root:!:17638:0:99999:7:::
daemon:*:17379:0:99999:7:::
bin:*:17379:0:99999:7:::
sys:*:17379:0:99999:7:::
sync:*:17379:0:99999:7:::
games:*:17379:0:99999:7:::
man:*:17379:0:99999:7:::
lp:*:17379:0:99999:7:::
mail:*:17379:0:99999:7:::
news:*:17379:0:99999:7:::
uucp:*:17379:0:99999:7:::
proxy:*:17379:0:99999:7:::
www-data:*:17379:0:99999:7:::
backup:*:17379:0:99999:7:::
list:*:17379:0:99999:7:::
irc:*:17379:0:99999:7:::
gnats:*:17379:0:99999:7:::
nobody:*:17379:0:99999:7:::
systemd-timesync:*:17379:0:99999:7:::
systemd-network:*:17379:0:99999:7:::
systemd-resolve:*:17379:0:99999:7:::
systemd-bus-proxy:*:17379:0:99999:7:::
syslog:*:17379:0:99999:7:::
_apt:*:17379:0:99999:7:::
lxd:*:17638:0:99999:7:::
messagebus:*:17638:0:99999:7:::
uuidd:*:17638:0:99999:7:::
dnsmasq:*:17638:0:99999:7:::
kay:$6$ON8Wi9Ow$Puwzhgbc2chaNEqWFO/UVH2yJ5zVb3WirwtCxQ5ssr2OEMAuYCrHscUNe.KPUhH6ND4CYx9WWu449W3mrzVtk/:17644:0:99999:7:::
sshd:*:17638:0:99999:7:::
tomcat9:!:17639::::::
```

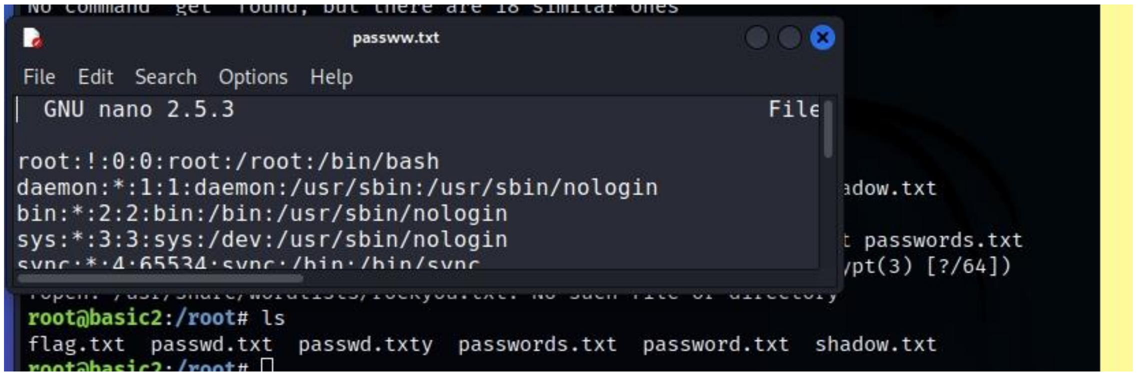
John the Ripper is a popular password cracking tool that supports many common hash types as well as a useful autodetect feature. It has been around for a while now, and as such, it continues to be one of the strongest and easiest to use crackers available.

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.



```
root@basic2:/root# nano passwd.txt
root@basic2:/root# nano passwd.txt
root@basic2:/root# unshadow passwd.txt shadow.txt > passwords.txt
root@basic2:/root# john --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
```

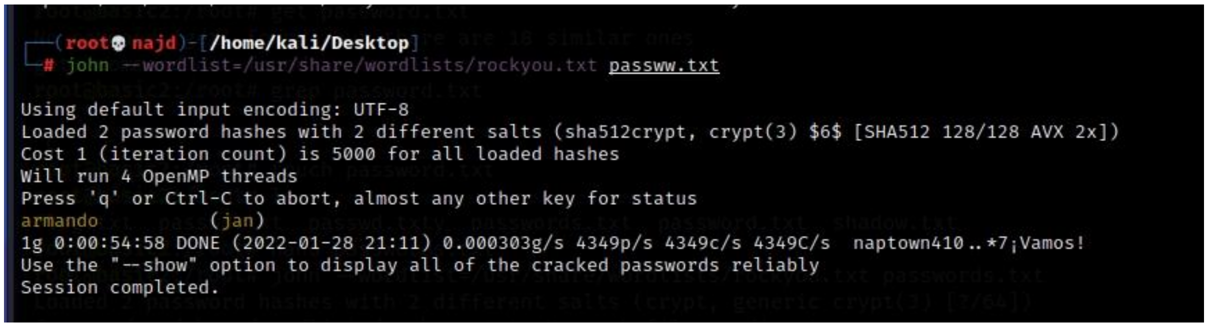
unshadow passwd.txt shadow.txt > passww.txt



```
GNU nano 2.5.3                                        File

root:!:0:0:root:/root:/bin/bash
daemon:*:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:*:2:2:bin:/bin:/usr/sbin/nologin
sys:*:3:3:sys:/dev:/usr/sbin/nologin
```

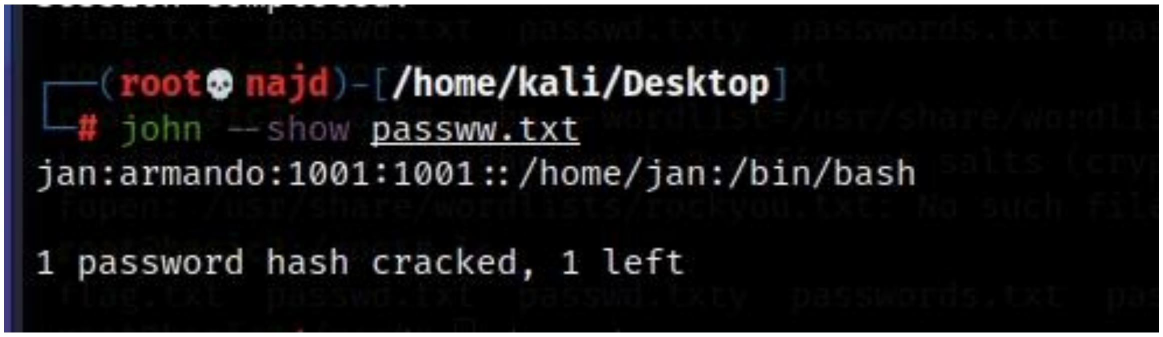
```
root@basic2:/root# ls
flag.txt  passwd.txt  passwd.txty  passwords.txt  password.txt  shadow.txt
```

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 rockyou.txt since it is quite a nice list. Use the --wordlist flag to specify the list to use and pass in our input file:



```
┌──(root💀najd)-[/home/kali/Desktop]
└─# john --wordlist=/usr/share/wordlists/rockyou.txt passww.txt

Using default input encoding: UTF-8
Loaded 2 password hashes with 2 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
armando          (jan)
1g 0:00:54:58 DONE (2022-01-28 21:11) 0.000303g/s 4349p/s 4349c/s 4349C/s  naptown410..*7¡Vamos!
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

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:



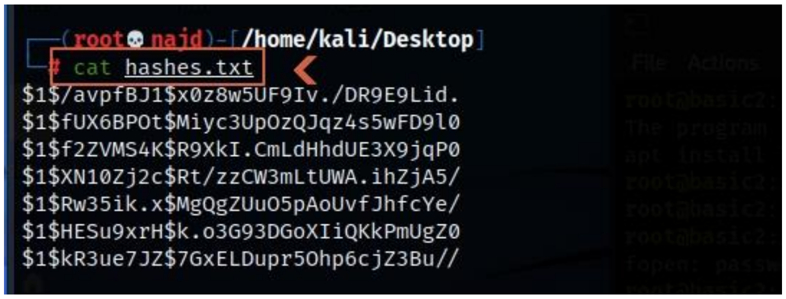
```
┌──(root💀 najd)-[/home/kali/Desktop]
└─# john --show passww.txt
jan:armando:1001:1001::/home/jan:/bin/bash

1 password hash 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.

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**:

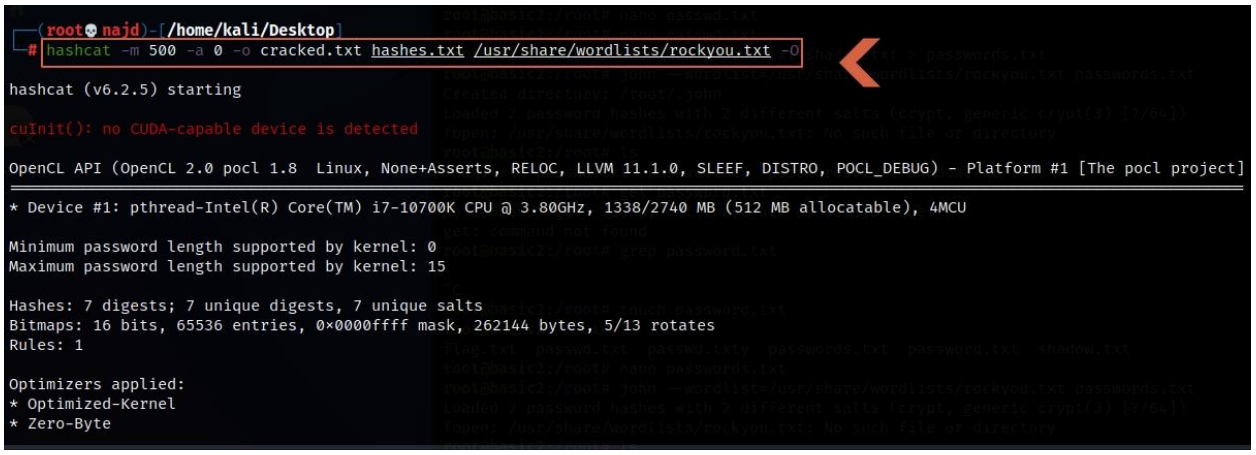


```
┌──(root💀 najd)-[/home/kali/Desktop]
└─# cat hashes.txt
$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.
$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0
$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0
$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/
$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/
$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0
$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//
```

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:

Run the following command to start cracking.

```
hashcat -m 500 -a 0 -o cracked.txt hashes.txt /usr/share/wordlists/rockyou.txt -O
```


```
┌──(root💀najd)-[/home/kali/Desktop]
└─# hashcat -m 500 -a 0 -o cracked.txt hashes.txt /usr/share/wordlists/rockyou.txt -O

hashcat (v6.2.5) starting

cuInit(): no CUDA-capable device is detected

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==============================================================================================================================
* Device #1: pthread-Intel(R) Core(TM) i7-10700K CPU @ 3.80GHz, 1338/2740 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 15

Hashes: 7 digests; 7 unique digests, 7 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Optimized-Kernel
* Zero-Byte
```

Let's break this down.

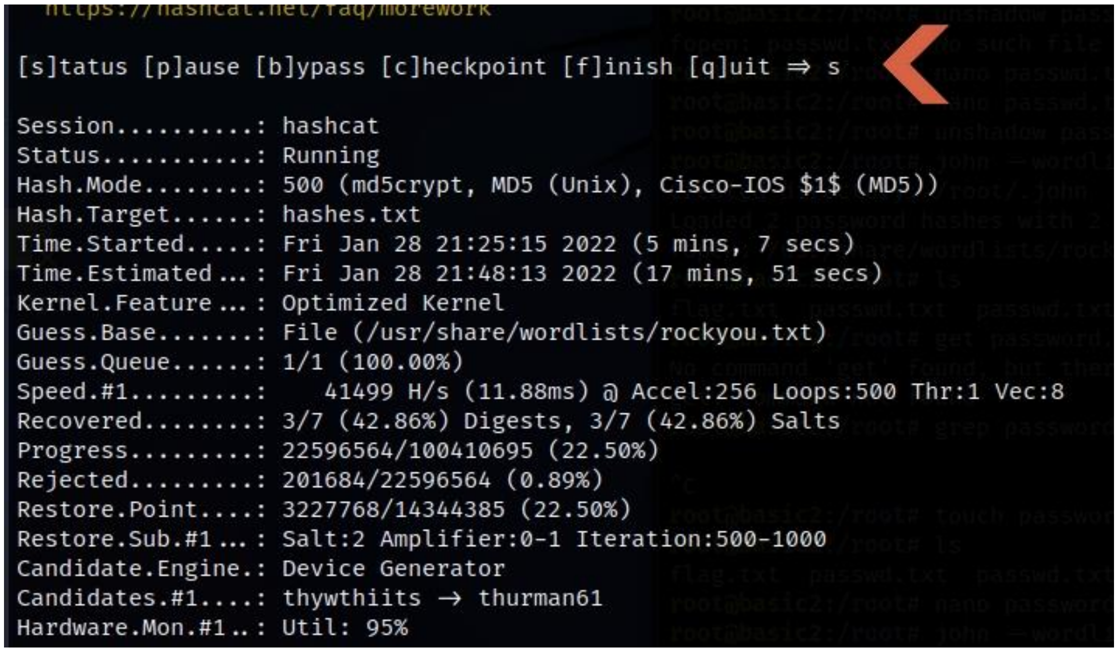
The -m flag specifies the mode we want to use.

The -a flag determines the attack type, in this case, 0 as the default straight mode.

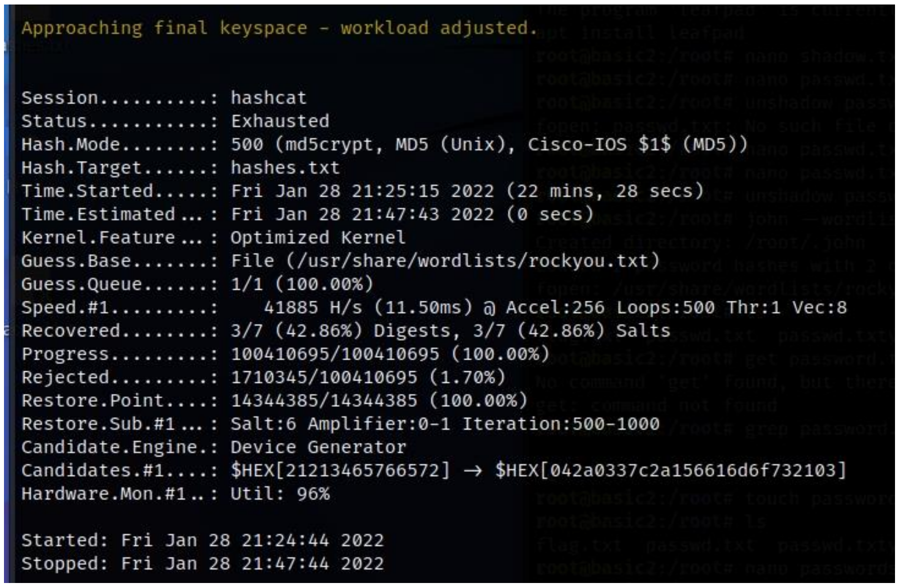
Then we specify the output file as cracked.txt with the -o flag and pass in hashes.txt as our input file that contains the hashes. We can also use a wordlist just like we did before with John.

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:

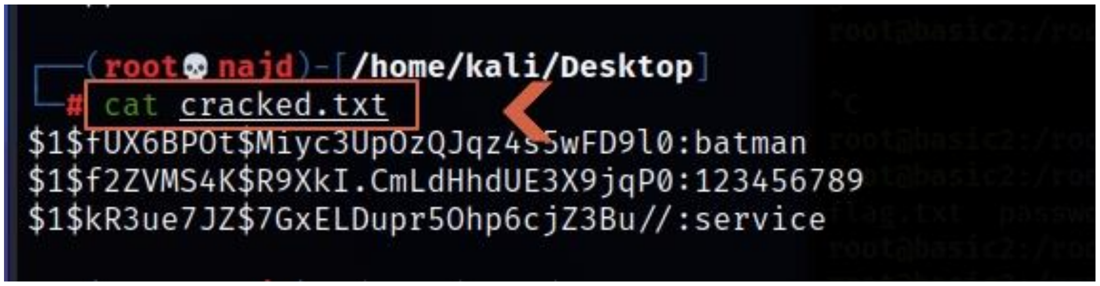

```
  https://hashcat.net/faq/morework

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit ⇒ s

Session..........: hashcat
Status...........: Running
Hash.Mode........: 500 (md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5))
Hash.Target......: hashes.txt
Time.Started.....: Fri Jan 28 21:25:15 2022 (5 mins, 7 secs)
Time.Estimated...: Fri Jan 28 21:48:13 2022 (17 mins, 51 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    41499 H/s (11.88ms) @ Accel:256 Loops:500 Thr:1 Vec:8
Recovered........: 3/7 (42.86%) Digests, 3/7 (42.86%) Salts
Progress.........: 22596564/100410695 (22.50%)
Rejected.........: 201684/22596564 (0.89%)
Restore.Point....: 3227768/14344385 (22.50%)
Restore.Sub.#1...: Salt:2 Amplifier:0-1 Iteration:500-1000
Candidate.Engine.: Device Generator
Candidates.#1....: thywthiits → thurman61
Hardware.Mon.#1..: Util: 95%
```

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.Mode........: 500 (md5crypt, MD5 (Unix), Cisco-IOS $1$ (MD5))
Hash.Target......: hashes.txt
Time.Started.....: Fri Jan 28 21:25:15 2022 (22 mins, 28 secs)
Time.Estimated...: Fri Jan 28 21:47:43 2022 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:    41885 H/s (11.50ms) @ Accel:256 Loops:500 Thr:1 Vec:8
Recovered........: 3/7 (42.86%) Digests, 3/7 (42.86%) Salts
Progress.........: 100410695/100410695 (100.00%)
Rejected.........: 1710345/100410695 (1.70%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:6 Amplifier:0-1 Iteration:500-1000
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[21213465766572] -> $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1..: Util: 96%

Started: Fri Jan 28 21:24:44 2022
Stopped: Fri Jan 28 21:47:44 2022
```

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



```
┌──(root💀najd)-[/home/kali/Desktop]
└─# cat cracked.txt
$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:batman
$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:123456789
$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:service
```

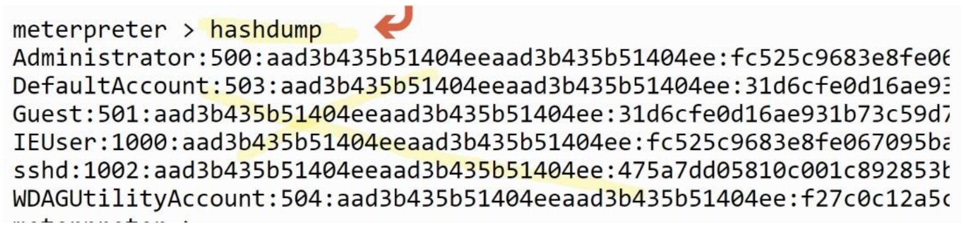
Online Hash Cracking

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 Crack Station, 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.

Lab 2 Windows System

I USE HASH FROM WINDOWS 10 SYSTEM IN VMWARE#



```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe06
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae93
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba
sshd:1002:aad3b435b51404eeaad3b435b51404ee:475a7dd05810c001c892853b
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:f27c0c12a5c
```

Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::

DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

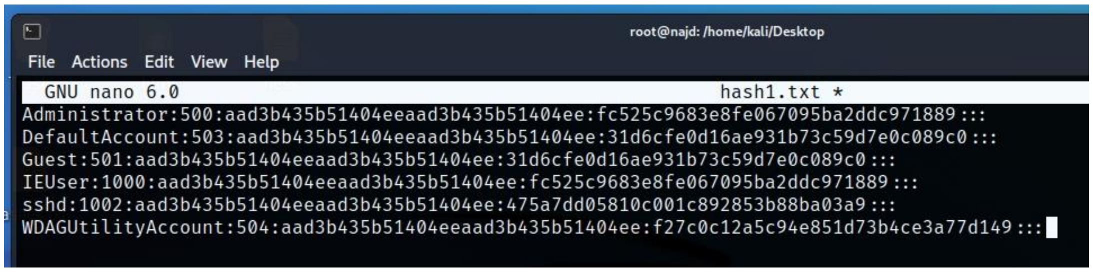
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::

sshd:1002:aad3b435b51404eeaad3b435b51404ee:475a7dd05810c001c892853b88ba03a9:::

WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:f27c0c12a5c94e851d73b4ce3a77d149:::

copy and paste these and save them to a txt file on our system for cracking.

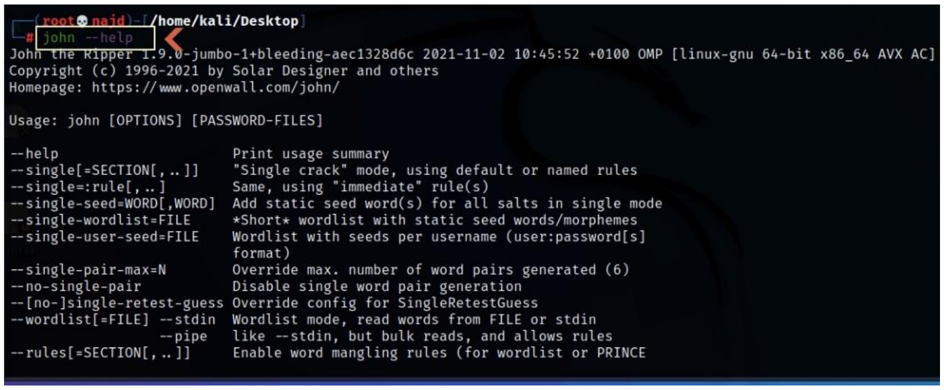
Now we have a text file on our desktop of the passwords but they are in an unreadable format. This is where a tool on kali known as Hashcat and John the Ripper comes in handy. In this tutorial we will looking at how we can crack the windows 10 password we collected in the hashdump using this tool.



```
  GNU nano 6.0                                          hash1.txt *
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
sshd:1002:aad3b435b51404eeaad3b435b51404ee:475a7dd05810c001c892853b88ba03a9:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:f27c0c12a5c94e851d73b4ce3a77d149:::
```

We have saved this .txt file on our desktop as hash.txt.

Now open a new terminal window and enter john

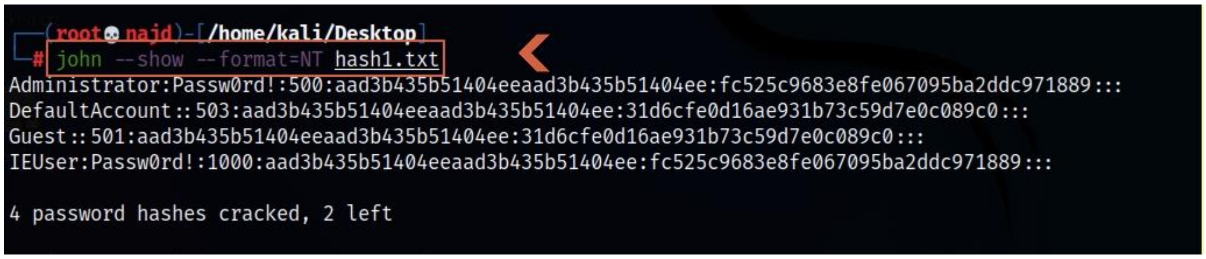


```
┌──(root💀najd)-[/home/kali/Desktop]
└─# john --help
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP [linux-gnu 64-bit x86_64 AVX AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

--help                    Print usage summary
--single[=SECTION[,..]]   "Single crack" mode, using default or named rules
--single=:rule[,..]       Same, using "immediate" rule(s)
--single-seed=WORD[,WORD] Add static seed word(s) for all salts in single mode
--single-wordlist=FILE    *Short* wordlist with static seed words/morphemes
--single-user-seed=FILE   Wordlist with seeds per username (user:password[s]
                          format)
--single-pair-max=N       Override max. number of word pairs generated (6)
--no-single-pair          Disable single word pair generation
--[no-]single-retest-guess Override config for SingleRetestGuess
--wordlist[=FILE] --stdin Wordlist mode, read words from FILE or stdin
                  --pipe  like --stdin, but bulk reads, and allows rules
--rules[=SECTION[,..]]    Enable word mangling rules (for wordlist or PRINCE
```

You will get a lot of options! Here we are going to do something really simple so we can ignore many of those options. We will explore them in future posts. For now, we are going to do something straight forward.

RUN the command john --show --format=NT hash.txt

This will tell John the Ripper to crack the hashed passwords contained in our hash.txt file and display the results. So go ahead and hit enter, let us see what we get.



```
┌──(root💀najd)-[/home/kali/Desktop]
└─# john --show --format=NT hash1.txt
Administrator:Passw0rd!:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
DefaultAccount::503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest::501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
IEUser:Passw0rd!:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::

4 password hashes cracked, 2 left
```

We were successful in cracking the password for the IE User and the Administrator. Both of these are the same Passw0rd! which is the default password used on the virtual box Windows 10 machine. It is cracked very quickly just a few seconds.