Fireside Fridays

Authentication and Password Cracking

Thanks to our sponsors!

ACTIVE | COUNTERMEASURES

BLACK HILLS
Information Security
• 2008 •

WILD WEST
HACKIN' FEST

Antisyphon Training

Lab requirements for this section

- Docker based password cracking lab
- Built on Ubuntu but should work on most OSes

```
sudo apt -y install docker.io
wget https://random-class.s3.us-east-1.amazonaws.com/ff-jtr-lab.tar.gz
sudo docker load -i ./ff-jtr-lab.tar.gz
sudo docker images
sudo docker run --rm -it ff-jtr-lab
cd root
ls
```

You should see the files:

```
password-list.txt  shadow
```

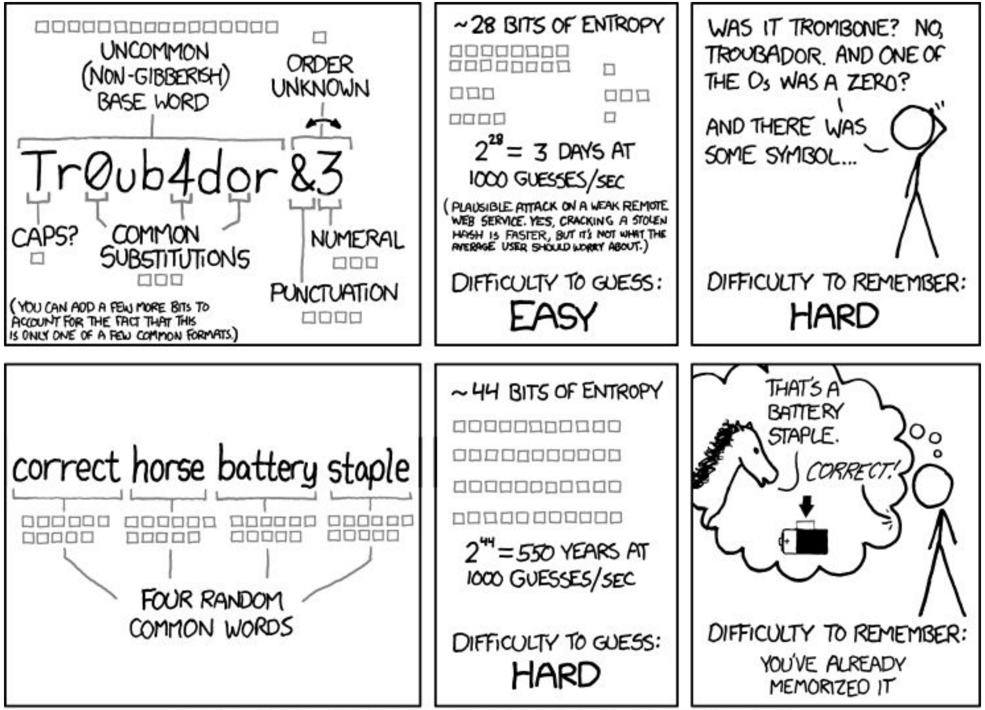
Authentication

- Verizon's annual breach report shows about 60% of compromises involve weak or stolen creds
- We pretty much drive users to create weak creds
  - 8-15 characters
  - Don't write it down
  - Change it frequently
  - Make it something you can remember
  - "Spring2025"

Password policies

- Similar to "What the market will bear" pricing
  - High complexity requirements
  - Frequent password changes
- You could be driving users to break the rules
  - Save them in Notepad
  - Easily derived so easy to remember
  - Reuse one password everywhere (personal and work)

xkcd is most awesome


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT

THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

https://xkcd.com/936

Methods of breaking passwords

- Implementation weaknesses
  - Dependent on the protocol in use
- Over the wire
  - Very slow
  - Generates lots of network traffic
- Offline
  - Much, much faster
  - Various techniques to speed up cracking

Implementation weakness

- Focuses on weaknesses in the security protocol
- Hashes with no salt
  - Vulnerable to Rainbow Table attacks
- Hashes that use a small output string
- Encryption used on predictable data
  - Like IP headers
- Windows still vulnerable to Pass the Hash??!!??

Over the wire

- Tool attempts to login, same as a user
- Looks for success to identify password match
- Attempts are usually rule based (more later)
- Cycles accounts to avoid lockout
- Common tools
    - Hydra (multiple variations)
    - Medusa
    - Brutus

Offline cracking

- Dictionary - Try wordlist as passwords
  - Works with random seed
- Rainbow tables - pre-hashed wordlist
  - Fastest but challenging with random seed
- Brute force - Let's try everything
  - Will always work…eventually
- Rule based - Mod dictionary to match user tricks
  - Usually the best balance to crack pa55w0rd5

Random seed

- Makes identical passwords look different
  - My password is "money" and so is yours
  - The resulting hash will be identical
  - Break mine and the attacker knows they have yours too
- It's just random data added to the password
- Stored with the password
- But now identical passwords generate different hashes

Offline cracking tools

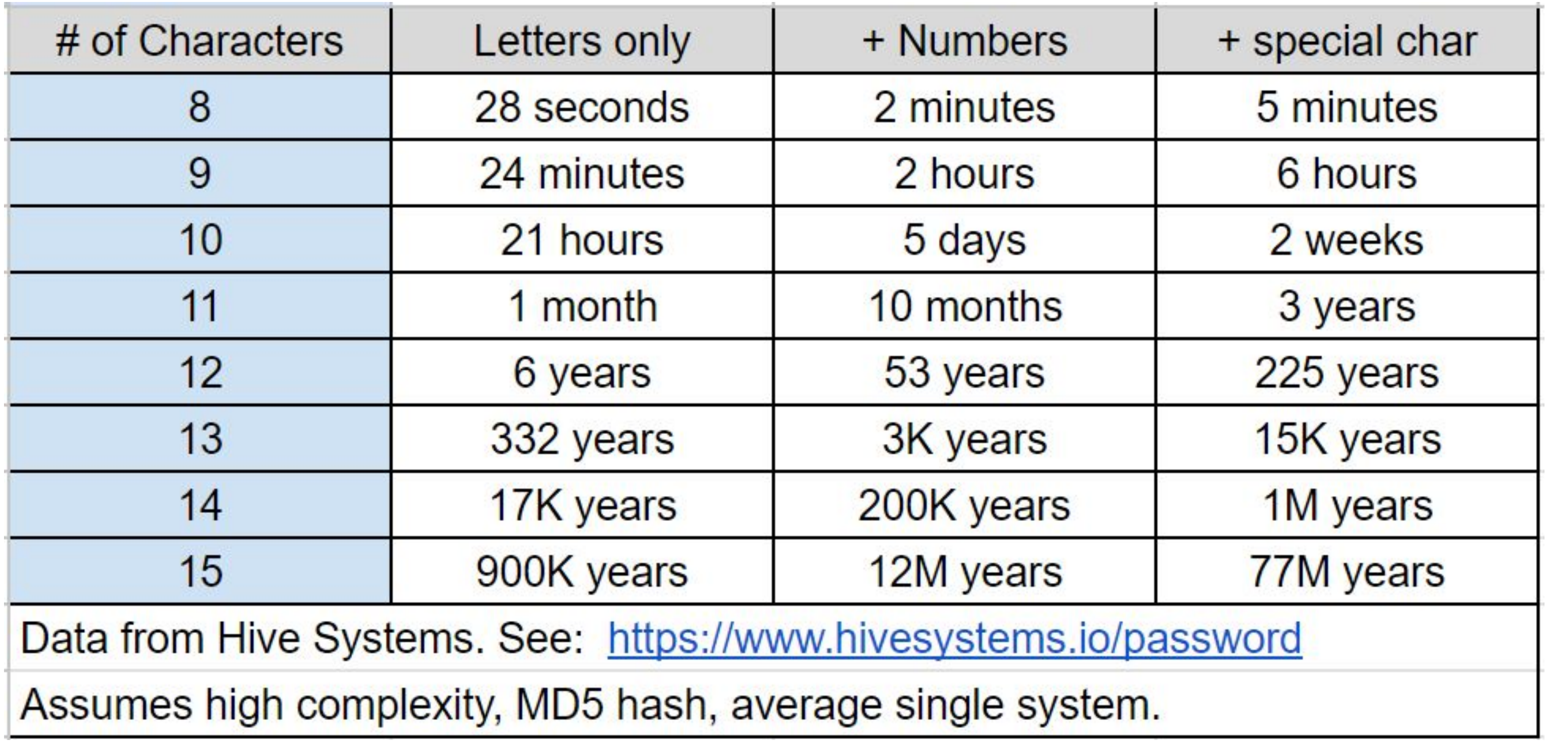
- Rainbow cracker
- John the Ripper
  - Parallel and distributed support
- Hashcat
- Crackstation
- Commercial options available
  - Not necessarily better than open source options
- Online options available
  - But you are sharing hashes with 3rd party

Extracting password hashes

- Varies by OS and applications
- Windows - Mimikatz is your friend/enemy
- Linux/UNIX - /etc/shadow file
- Databases - varies by application
    - Sometimes without random seeds
    - Sometimes in plaintext!

https://github.com/ParrotSec/mimikatz

How long to brute force passwords?

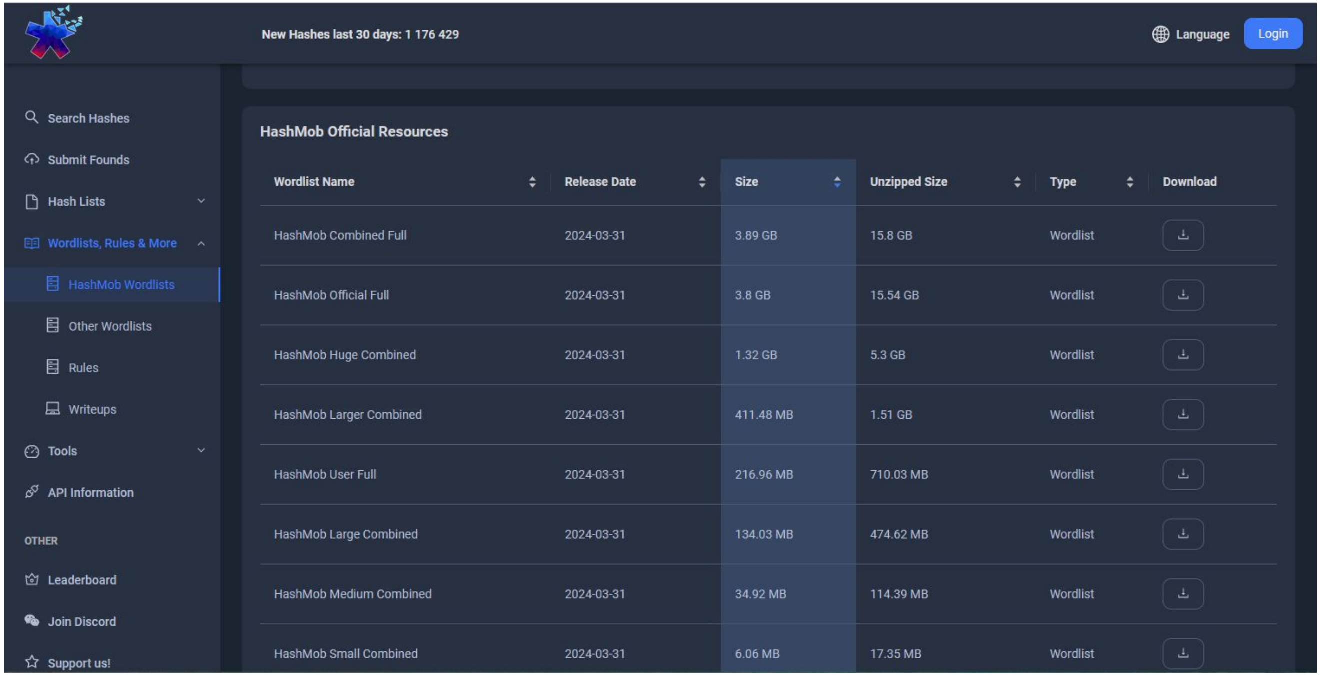


# of Characters	Letters only	+ Numbers	+ special char
8	28 seconds	2 minutes	5 minutes
9	24 minutes	2 hours	6 hours
10	21 hours	5 days	2 weeks
11	1 month	10 months	3 years
12	6 years	53 years	225 years
13	332 years	3K years	15K years
14	17K years	200K years	1M years
15	900K years	12M years	77M years

Data from Hive Systems. See: https://www.hivesystems.io/password

Assumes high complexity, MD5 hash, average single system.

Where to get good wordlists?



HashMob Official Resources

Wordlist Name	Release Date	Size	Unzipped Size	Type	Download
HashMob Combined Full	2024-03-31	3.89 GB	15.8 GB	Wordlist	
HashMob Official Full	2024-03-31	3.8 GB	15.54 GB	Wordlist	
HashMob Huge Combined	2024-03-31	1.32 GB	5.3 GB	Wordlist	
HashMob Larger Combined	2024-03-31	411.48 MB	1.51 GB	Wordlist	
HashMob User Full	2024-03-31	216.96 MB	710.03 MB	Wordlist	
HashMob Large Combined	2024-03-31	134.03 MB	474.62 MB	Wordlist	
HashMob Medium Combined	2024-03-31	34.92 MB	114.39 MB	Wordlist	
HashMob Small Combined	2024-03-31	6.06 MB	17.35 MB	Wordlist	

https://hashmob.net/resources/hashmob

Password cracking Hands-on

- We'll do a hands-on walkthrough with John the Ripper
- Very effective software
- We'll run John in a Docker container and crack some passwords

Starting the Container with a terminal

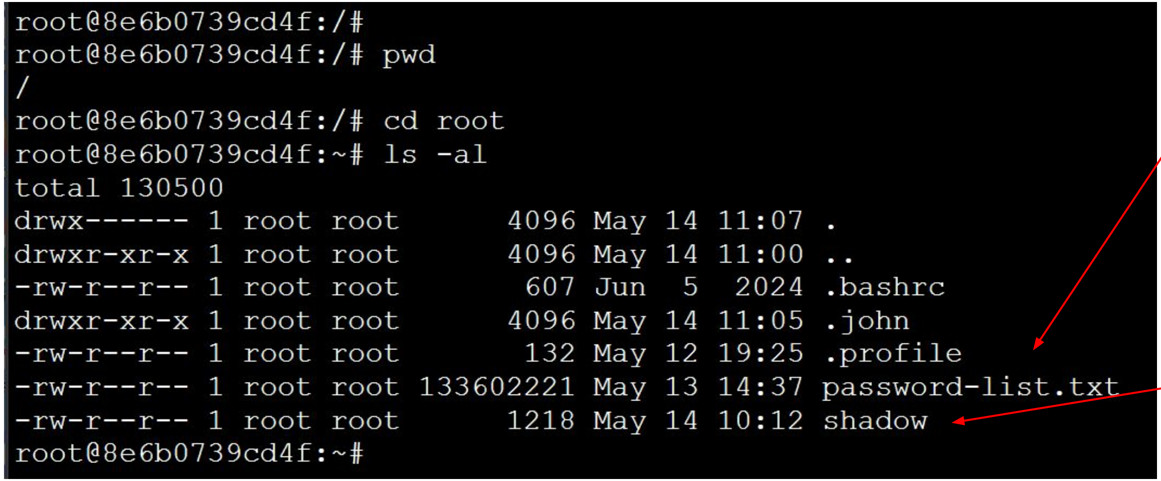
Follow the install steps. Then do a:

```
sudo docker run --rm -it ff-jtr-lab
cd root
ls
```

You should see the files:

```
password-list.txt  shadow
```

Navigate to root's home dir



```
root@8e6b0739cd4f:/#
root@8e6b0739cd4f:/# pwd
/
root@8e6b0739cd4f:/# cd root
root@8e6b0739cd4f:~# ls -al
total 130500
drwx------ 1 root root      4096 May 14 11:07 .
drwxr-xr-x 1 root root      4096 May 14 11:00 ..
-rw-r--r-- 1 root root       607 Jun  5  2024 .bashrc
drwxr-xr-x 1 root root      4096 May 14 11:05 .john
-rw-r--r-- 1 root root       132 May 12 19:25 .profile
-rw-r--r-- 1 root root 133602221 May 13 14:37 password-list.txt
-rw-r--r-- 1 root root      1218 May 14 10:12 shadow
root@8e6b0739cd4f:~#
```

Dictionary

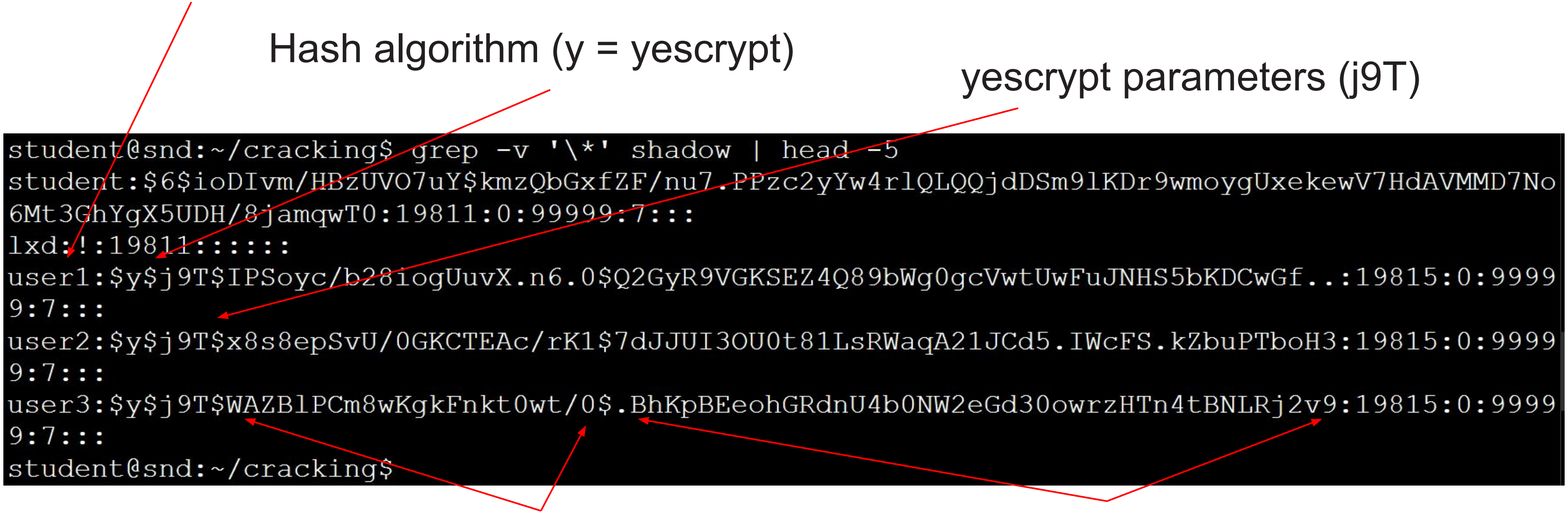
Passwords to crack

Password file format

Account name

Hash algorithm (y = yescrypt)

yescrypt parameters (j9T)

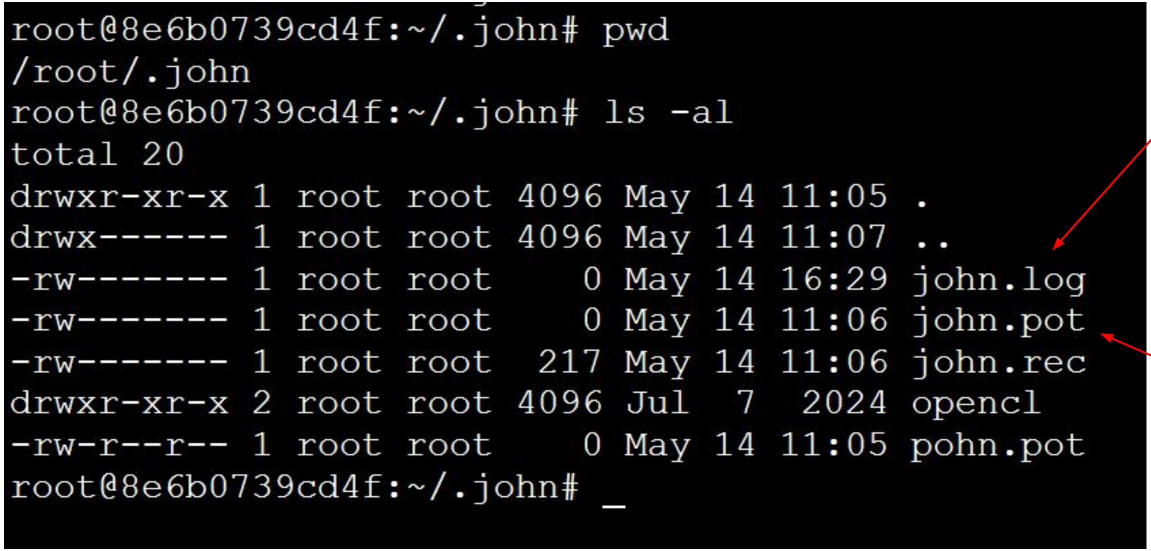


```
student@snd:~/cracking$ grep -v '\*' shadow | head -5
student:$6$ioDIvm/HBzUVO7uY$kmzQbGxfZF/nu7.PPzc2yYw4rlQLQQjdDSm9lKDr9wmoygUxekewV7HdAVMMD7No
6Mt3GhYgX5UDH/8jamqwT0:19811:0:99999:7:::
lxd:!:19811::::::
user1:$y$j9T$IPSoyc/b28iogUuvX.n6.0$Q2GyR9VGKSEZ4Q89bWg0gcVwtUwFuJNHS5bKDCwGf..:19815:0:9999
9:7:::
user2:$y$j9T$x8s8epSvU/0GKCTEAc/rK1$7dJJUI3OU0t81LsRWaqA21JCd5.IWcFS.kZbuPTboH3:19815:0:9999
9:7:::
user3:$y$j9T$WAZBlPCm8wKgkFnkt0wt/0$.BhKpBEeohGRdnU4b0NW2eGd30owrzHTn4tBNLRj2v9:19815:0:9999
9:7:::
student@snd:~/cracking$
```

Random seed

Hashed password

John's files

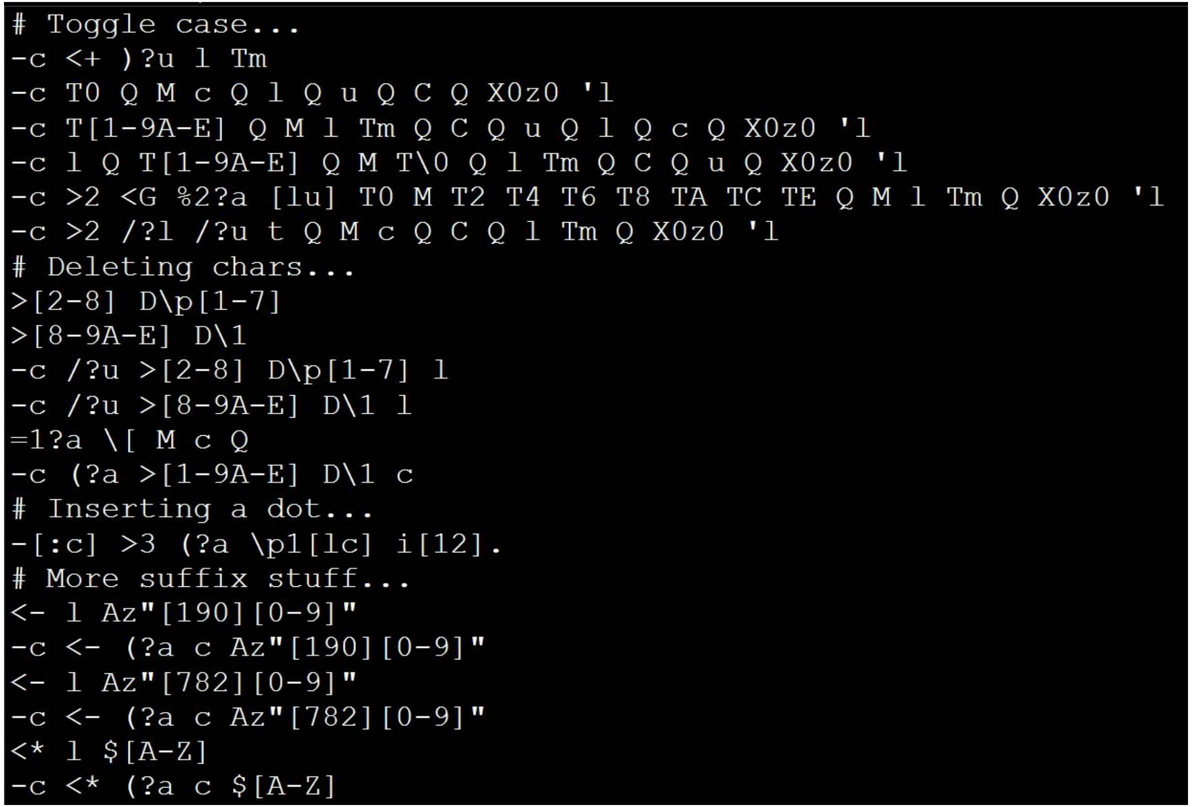


```
root@8e6b0739cd4f:~/.john# pwd
/root/.john
root@8e6b0739cd4f:~/.john# ls -al
total 20
drwxr-xr-x 1 root root 4096 May 14 11:05 .
drwx------ 1 root root 4096 May 14 11:07 ..
-rw------- 1 root root    0 May 14 16:29 john.log
-rw------- 1 root root    0 May 14 11:06 john.pot
-rw------- 1 root root  217 May 14 11:06 john.rec
drwxr-xr-x 2 root root 4096 Jul  7  2024 opencl
-rw-r--r-- 1 root root    0 May 14 11:05 pohn.pot
root@8e6b0739cd4f:~/.john# _
```

Activity log

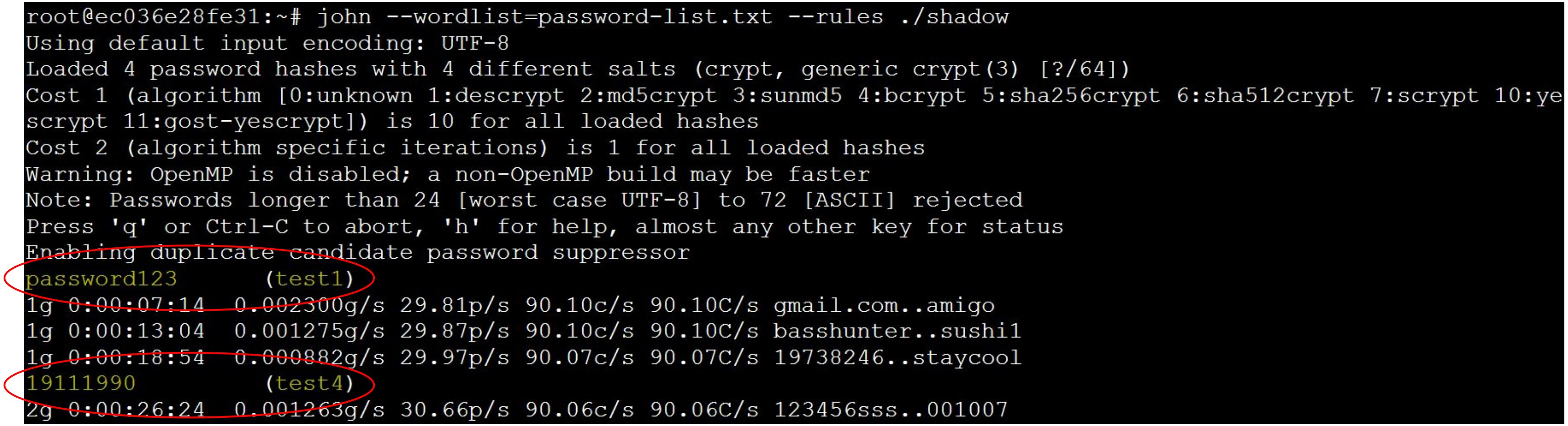
Previously cracked accounts

Rules in /etc/john/john.conf



```
# Toggle case...
-c <+ )?u l Tm
-c T0 Q M c Q l Q u Q C Q X0z0 'l
-c T[1-9A-E] Q M l Tm Q C Q u Q l Q c Q X0z0 'l
-c l Q T[1-9A-E] Q M T\0 Q l Tm Q C Q u Q X0z0 'l
-c >2 <G %2?a [lu] T0 M T2 T4 T6 T8 TA TC TE Q M l Tm Q X0z0 'l
-c >2 /?l /?u t Q M c Q C Q l Tm Q X0z0 'l
# Deleting chars...
>[2-8] D\p[1-7]
>[8-9A-E] D\1
-c /?u >[2-8] D\p[1-7] l
-c /?u >[8-9A-E] D\1 l
=1?a \[ M c Q
-c (?a >[1-9A-E] D\1 c
# Inserting a dot...
-[:c] >3 (?a \p1[lc] i[12].
# More suffix stuff...
<- l Az"[190][0-9]"
-c <- (?a c Az"[190][0-9]"
<- l Az"[782][0-9]"
-c <- (?a c Az"[782][0-9]"
<* l $[A-Z]
-c <* (?a c $[A-Z]
```

Running John the Ripper

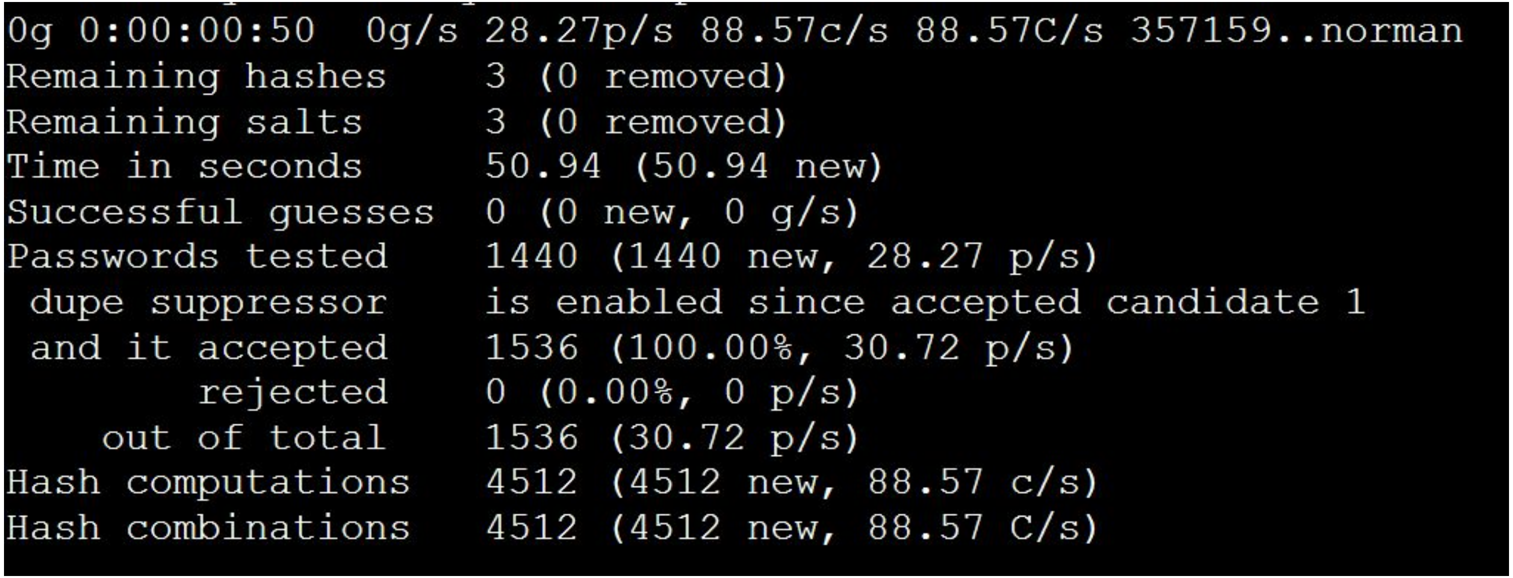


```
root@ec036e28fe31:~# john --wordlist=password-list.txt --rules ./shadow
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (crypt, generic crypt(3) [?/64])
Cost 1 (algorithm [0:unknown 1:descrypt 2:md5crypt 3:sunmd5 4:bcrypt 5:sha256crypt 6:sha512crypt 7:scrypt 10:ye
scrypt 11:gost-yescrypt]) is 10 for all loaded hashes
Cost 2 (algorithm specific iterations) is 1 for all loaded hashes
Warning: OpenMP is disabled; a non-OpenMP build may be faster
Note: Passwords longer than 24 [worst case UTF-8] to 72 [ASCII] rejected
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
Enabling duplicate candidate password suppressor
password123      (test1)
1g 0:00:07:14  0.002300g/s 29.81p/s 90.10c/s 90.10C/s gmail.com..amigo
1g 0:00:13:04  0.001275g/s 29.87p/s 90.10c/s 90.10C/s basshunter..sushi1
1g 0:00:18:54  0.000882g/s 29.97p/s 90.07c/s 90.07C/s 19738246..staycool
19111990         (test4)
2g 0:00:26:24  0.001263g/s 30.66p/s 90.06c/s 90.06C/s 123456sss..001007
```

Hit <spacebar> to see current statistics

```
john --wordlist=password-list.txt --rules ./shadow
```

Detailed status



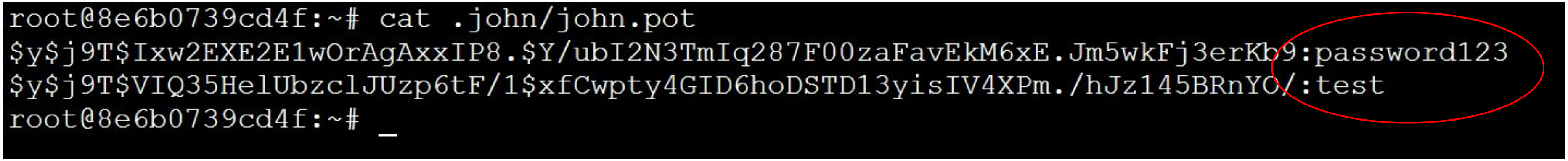
```
0g 0:00:00:50  0g/s 28.27p/s 88.57c/s 88.57C/s 357159..norman
Remaining hashes    3 (0 removed)
Remaining salts     3 (0 removed)
Time in seconds     50.94 (50.94 new)
Successful guesses  0 (0 new, 0 g/s)
Passwords tested    1440 (1440 new, 28.27 p/s)
 dupe suppressor    is enabled since accepted candidate 1
 and it accepted    1536 (100.00%, 30.72 p/s)
        rejected    0 (0.00%, 0 p/s)
    out of total    1536 (30.72 p/s)
Hash computations   4512 (4512 new, 88.57 c/s)
Hash combinations   4512 (4512 new, 88.57 C/s)
```

Press "s" for detailed status while running

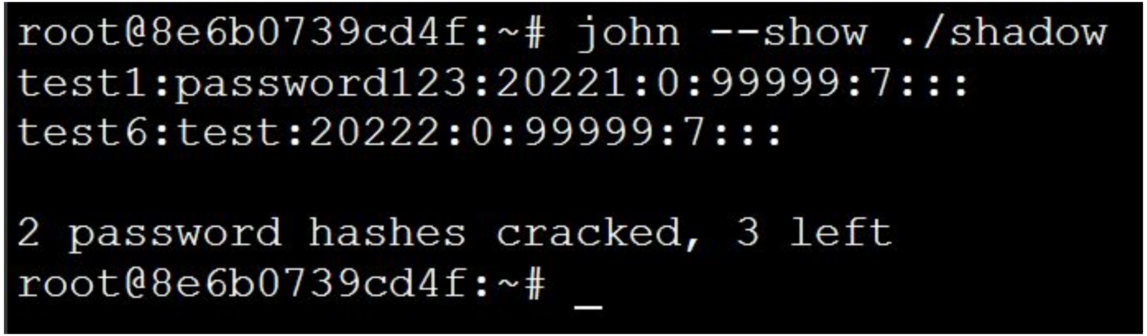
Why is the computation rate so slow?

- "c/s" identifies the brute force speed
- Number of guesses taking place
- Numbers in my slides are pretty slow
    - It's a Docker instance
    - Running in a VM
    - On a Proxmox system
    - With other VMs running
- Your speed should be faster

Cracking results



```
root@8e6b0739cd4f:~# cat .john/john.pot
$y$j9T$Ixw2EXE2E1wOrAgAxxIP8.$Y/ubI2N3TmIq287F00zaFavEkM6xE.Jm5wkFj3erKb9:password123
$y$j9T$VIQ35HelUbzclJUzp6tF/1$xfCwpty4GID6hoDSTD13yisIV4XPm./hJz145BRnYO/:test
root@8e6b0739cd4f:~# _
```



```
root@8e6b0739cd4f:~# john --show ./shadow
test1:password123:20221:0:99999:7:::
test6:test:20222:0:99999:7:::

2 password hashes cracked, 3 left
root@8e6b0739cd4f:~# _
```

Improving cracking performance

- Enable OpenMP support (parallelization)
- Enable MPI support
- Add additional GPUs
    - Then enable OpenCL
- Run distributed (DJohn)
- Most of above require compiling from source

Password managers

- In an ideal world, no more passwords
- We are a ways off from that world
- Password managers have been shown to produce the greatest improvements in password integrity
- Options to share with teams when needed
  - Critical when a user leaves the organization
- But they are a single point of password failure

2-Factor authentication

- Includes two of the following:
  - Something you know (password, PIN)
  - Something you have (private key, Auth app)
  - Biometric (fingerprint, iris scan)
- Provides better than 2X security improvement
- Easiest protection against phishing
- Improvement but we have seen vulnerabilities

This is the last Fireside Friday

- At least for a while
- Too much to keep up with with so much going on
- This is class #17
- In case you missed any of the previous ones

https://www.activecountermeasures.com/fireside-fridays/

But…

- I'm working on an update to "Intro to Threat Hunting"
- Same 6 hour format
- We'll be running that in a month or so
- Subscribe to get notified when we firm up a date

Wrap up

- Thank you for attending!
- Certs & video will go out by Monday
- If you have any lingering questions, the Discord channel will remain active
  - Also a good chance to socialize with others in the class
  - Have other tips and tricks? Please share with others!
- **Thank you** for sharing your time with us!