Password Cracking With Your GPU

And Why Password Complexity Matters

For Wolverhampton Linux User Group

By Adam Sweet

20/03/21

Who Am I?

- Adam Sweet, I run Transitiv Technologies Ltd
- Open Source Consultant
- Linux user for 20 years (Mandrake Linux 7.1, mid-2000)
- 15 years Linux systems administration
- OSes, networks, databases, mail, DNS, virtualisation, network monitoring etc
- Linux podcast presenter, conference organiser, 21 year UK Linux community participant
- A Linux geek who likes to play with stuff

Password Security

- Anywhere you use a username and password to sign in should be hashing your password to protect it from prying eyes
  - Websites
  - Work applications
  - Corporate VPNs
- Otherwise anyone with access to the user database can log in as you
  - Do you use the same username and password in lots of places?
- Anything that emails your password to you isn't hashing your password
  - The UK LUG mailing list server can send password reminders
  - You paid attention to the warning when you signed up, right?

Break-Ins

- What if they get broken into? It happens a lot
  - TalkTalk (British ISP)
  - British Airways
  - Adult FriendFinder
  - Ashley Madison
  - Dropbox
  - Experian
  - MyFitnessPal
  - Many more - https://haveibeenpwned.com/PwnedWebsites
- The attacker will aim to steal the user database

Password Hashing

- Hashing is a mathematical process which turns a human readable information into an unintelligible string of characters
- A password hash might look like:
  - `c57aeddaffce62fead6be61022eb1340`
- The idea with password hashing is that it only works one way
  - You can hash a password easily with a known method and by repeating the hashing process with the same method you get exactly the same hash each time
  - Once hashed it's almost impossible to reverse it to get the original password
  - When you log in somewhere, the server receives the plaintext password from the login box (over TLS), hashes it using the same hash method that was used to hash the stored password, compares it with the stored hash
  - If they match you're logged in but the OS or website never stores the plaintext password

Cracking Password Hashes

- Why would you want to?
  - You forgot your own password and can't log in to a system you own
  - To test the strength of your own passwords
  - As a fun technical exercise
  - You're a sysadmin and you want to check for weak passwords amongst your users
    - Check with your boss before doing so, you could get fired
    - Set password complexity requirements rather than closing the gate after the horse has bolted
  - If you're cracking somebody else's password to check what they're up to on their phone/email/social media etc, you've taken a wrong turn in life
- Since it's a one way process, it's not going to be easy but there are ways to increase your chances of success

Human v Computer

- Manually typing one password after another into a login box has a very low chance of success and it will take many lifetimes
  - More than likely you will lose track of what you already tried
- Repeating tasks very quickly is what computers were designed for
- There are programs that can guess for you, e.g.
  - John the Ripper
  - Ophcrack (for Windows passwords)
  - Hashcat – we will look at this one

Hashcat

- Hashcat is a command line password recovery tool
- https://hashcat.net/hashcat/
- It runs on Linux, macOS and Windows
- It can run on your CPU or GPU (your graphics card)

Why a GPU?

- Guessing passwords is a job that scales very well across many processing cores
- CPU cores are faster than GPU cores but CPUs might have 2-32 cores while GPUs typically have hundreds if not thousands
- My CPU
  - AMD Ryzen 7 3700X
  - 8 cores (16 threads)
  - Base clock: 2.2 GHz
  - Boost clock: 3.6 GHz
- My GPU
  - Nvidia GeForce RTX 2060
  - CUDA Cores: 1920
  - Base clock: 1365 MHz
  - Boost clock: 1680 Mhz

Getting a Password Hash

- On Linux the file `/etc/passwd` contains user entries like:
  - `testing:x:1001:1001:testing,,,:/home/testing:/bin/bash`
- Traditionally this included the password hash in the second field, these days the x just tells you a password was assigned
- On its own that doesn't tell you the user's password hash, you need access `/etc/shadow`, which contains entries like (forgive the multi-line formatting)
- `testing:`
  `$6$pDt77oWp7MJyc6zT$dy0v3rIfMoCELrWbgJ2TE5az2IHPfiiMjjf4icSxfSXR5fkOYKC66t`
  `eXowVboZ91V2UZ8NJNx.5PXGW8mp1Ov.:18506:0:99999:7:::`
- John the Ripper provides a tool called unshadow to extract the password hashes from a Linux shadow file
  - `sudo unshadow /etc/passwd /etc/shadow > unshadow.txt`
- Or you can just rip the second field from between the colons in `/etc/shadow`
  - `$6$pDt77oWp7MJyc6zT$dy0v3rIfMoCELrWbgJ2TE5az2IHPfiiMjjf4icSxfSXR5fkOYKC66teXowVboZ`
    `91V2UZ8NJNx.5PXGW8mp1Ov.`
- Either way you'll need root/sudo access to `/etc/shadow`

Hash Types

- Now we have a password hash we need to know what type of hashing was used to create it
- Hashcat supports a large number of hash types
- The number after the first $ signs tells us what hash type it is:
  - `$6`
  - https://hashcat.net/wiki/doku.php?id=example_hashes
  - This is a SHA512crypt hash, hashcat hash type 1800 on the wiki

Setting Up Hashcat

- For Ubuntu based distros
  - `apt install hashcat`
- If you're using an Nvidia GPU it's simple
  - You need `nvidia-cuda-toolkit`
  - And the latest Nvidia proprietary driver
    - That's `nvidia-driver-450` at the time of writing
- Everyone else – I'm sorry, I don't use your distro
  - I imagine it's well documented for Fedora and Arch

Setting Up OpenCL

- If you're not using an Nvidia GPU it gets messy
- You need an OpenCL implementation
  - There's a proprietary Intel CPU OpenCL implementation behind a sign-up wall
  - https://software.intel.com/content/www/us/en/develop/tools/opencl-cpu-runtime.html
- AMD abandoned their CPU OpenCL implementation
  - You have to use the Intel one or POCL, a third-party hardware independent one
  - On Ubuntu, POCL is in the repos - `pocl-opencl-icd`
- There's `intel-opencl-icd` which supports Intel GPUs
  - Interesting to see what happens with Intel X^e GPUs
- I don't have any AMD GPUs so I didn't try, but on Linux at least, it looks muddy
- Nvidia won the GPU compute war, if you're serious use Nvidia hardware

Running Hashcat

- `hashcat -m 1800 -a 3 -o ./found-passwords.txt -O -w 4 --session=firstattempt --status --status-timer=30 test-password.txt`
- `-m` – specify hashcat's hash type (1800 is sha512crypt)
- `-a` – attack mode (3 is brute force)
- `-o` – output file for recovered passwords
- `-O` – Optimises performance but limits the supported password length (16 chars for sha512crypt)
- `-w` – workload profile (1-4, 4 is 'nightmare' – fastest/biggest impact on system responsiveness/'insane' power consumption)
- `--session` – allows you to quit and resume later with `--restore --session=`*`sessionname`*
- `--status` and `--status-timer=30` – output status every 30 seconds
- `test-password.txt` – the file containing the hash

Specifying Devices

- You can see which devices hashcat can use with
  - `hashcat -I`
- By default it will just use the first listed device
- You can specify a device from the output of the above with `-d`
  - `hashcat -m 1800 -d 2 -a 3 -o ./found-passwords.txt -O -w 4 --session=firstattempt --status --status-timer=30 test-password.txt`
  - `-d` parameter can be comma separated to pass multiple devices
  - You can specify an OpenCL platform type with –`opencl-platforms=`*`num`*
    - E.g. use all CUDA devices
- You can further specify which device types to use with *`-D`*
- You might need to use *`--force`* to get it to stop complaining and run – but it won't be fast

My Hardware

- My CPU
  - AMD Ryzen 7 3700X 8-Core Processor
  - 8 cores (16 threads)
  - Base clock: 2.2 GHz
  - Boost clock: 3.6 GHz
- My GPU
  - Nvidia GeForce RTX 2060
  - CUDA Cores: 1920
  - Base clock: 1365 MHz
  - Boost clock: 1680 Mhz

Password Cracking Speed

- For Linux sha512crypt password hashes

Device	Hashes Per Second
Nvidia Geforce RTX 2060	160,000
Nvidia Geforce GTX 1050 TI GPU	48,000
Nvidia Geforce RTX 2060 without -O	34,200
Intel i7-7700HQ CPU	2,600
Intel HD 630 GPU	2,300
AMD Ryzen 7 3700X CPU	1,800

- The Ryzen CPU performance surprised me but I haven't investigated yet

Time to Crack

- For sha512crypt brute force at 160,000 h/s

Characters in Password	Time, Default Charset	Time, All Chars
All 1, 2 and 3 char passwords	Under 2 secs each, 5 total	2, 2 and 7 seconds
4	18 seconds	8 mins
5	10 mins	13 hours
6	6 hours 30 mins	51 days
7	9 ½ days	13 ½ years
8	13 months	1311 years
9	43 years	120,000 years

Password Complexity Matters

- A serious attacker will have faster GPUs and more of them
- A state-backed attacker might have supercomputing resources
- Using the default charset
  - A 6 char password is trivial for me
  - 7 chars is do-able if I was serious
  - 8 chars is a long-game without serious resources
  - 9 chars is unrealistic without serious compute resources (state-backed attacker)
- With all chars I'd have to tap out after 5 chars, a pro wouldn't
- You don't want to be the low-hanging fruit

Benchmarking Your Hardware

- You can see how your hardware will perform using hashcat's benchmarking mode
  - `hashcat -b`
- Uses `-O` implictly to optimise performance
- You can specify devices to compare performance as with a regular hashcat run
  - `hashcat -b -d 1,2`
  - `hashcat -b --opencl-platforms=2`
- There's a great variation in the speed your hardware can guess different hash types

Not All Hashing Methods Are Equal

- Using Nvidia GeForce RTX 2060

Hash Type	Hashes Per Second
NTLM (Windows password hashes)	43 billion
md5crypt	11 million
sha512crypt	170K
macOS v10.8+	14,500
bcrypt	12,500

Character Sets

- There are 4 basic character types
  - l = lower case
  - u = upper case
  - d = digits
  - s = special chars
  - a = all alphanumeric and special chars
  - There are also some non-English charsets under `/usr/share/hashcat/charsets/`
- You can group these character types into charsets to optimise for certain patterns in the password
- Hashcat accepts up to 4 charsets at a time
- You use a character mask to specify where to expect these charsets in a password
- https://hashcat.net/wiki/doku.php?id=mask_attack

Using Charsets

- By default hashcat uses a best effort pattern to guess the characters
- `Guess.Mask.....: ?1?2?2?2?2?2?2?3 [8]`
- `Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined`
- Anything that follows a `?` symbol is a variable
  - Otherwise it's an explicit character definition
- `-1 ?l?d?u` – characters in set 1 can be lower case, a digit or upper case
- `-2 ?l?d` – characters in set 2 can be lower or upper case
- `-3 ?l?d*!$@_` - characters in set 3 may be lower case, a digit or one of the explicitly defined special characters

Using Charsets and Masks

- `Guess.Mask.....: ?1?2?2?2?2?2?3 [8]`
- `Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined`
- When you put the charset groups together with the mask, the password guesses will:
  - Start with upper or lower case, or a digit
  - The following 6 chars are upper or lower case
  - The last char can be lower case, a digit or one of the given special chars
- A mask of `?1?1?1?1?1?1`, means
  - The first 4 chars from the first charset
  - The fifth char is the number 1 (not preceded by `?`)
  - The last 2 chars are from the first charset again
- The following charset/mask would try all possible characters in any position, starting from 1 char up to 9:
- `-1 ?a ?1?1?1?1?1?1?1?1?1 --increment`
- https://hashcat.net/wiki/doku.php?id=mask_attack

Optimising Crack Time

- Anything you know about the password you're trying to crack helps reduce the time it will take
- If you know the length you have a fixed number of hashes to try
- If you know the character types you can reduce the runtime considerably
- If you know any individual characters it will dramatically reduce the runtime, especially if you know where they appear
- A 7 char Apache htpasswd crack was expected to take 72 days using all possible characters
- Knowing it only used lower case and digits brought that down to 2 hours
- It was actually cracked in 11 minutes
- `hashcat -m 1800 -a 3 -o ./found-passwords.txt -O -w 4 -1 ?l?d --session=secondattempt --status --status-timer=30 test-password.txt ?1?1?1?1?1?1?1`

Using Password Lists

- Unless you're attacking a specific hash, it's likely you have a set of hashes and you want to crack as many as possible with the least time/resources
- If you're looking for low hanging fruit then using a password list will give you a good chance of cracking the weakest passwords
- There are many password lists out there
- https://en.wikipedia.org/wiki/List_of_the_most_common_passwords
- https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-10000.txt
- `-a 0` – attack mode uses a password list
- `hashcat -m 1800 -a 0 -o found-passwords.txt --status --status-timer=30 test-password3.txt passwordlist.txt`

Other Attack Types

- Combinator Attack – use two dictionaries to combine common password lists
  - https://hashcat.net/wiki/doku.php?id=combinator_attack
- Hybrid Attack – dictionary with brute-force chars appended or prepended
  - https://hashcat.net/wiki/doku.php?id=hybrid_attack
- Rule-Based Attack - Use rules to generate password guesses, e.g.
  - Substitute letters a, e, i and o for @, 3, 1, 0
  - Swap last letter o for 0
  - Append 1
  - https://hashcat.net/wiki/doku.php?id=rule_based_attack

Cracking Windows Passwords

- You can copy the following files with Windows offline
  - `C:/Windows/System32/config/SAM`
  - `C:/Windows/System32/config/SYSTEM`
- Or export the registry files with Windows running:
  - `reg save HKLM\SYSTEM SYSTEM.hiv`
  - `reg save HKLM\SAM SAM.hiv`
- You can use a Windows tool called mimikatz to extract the hashes
  - Look for lines that start with `User`
  - The hash will be on the following line that starts `Hash NTLM`
- Then use hashcat with hashtype 1000 (NTLM)

Weak Passwords

- Never use variations of the following
  - The same as your username (especially root!)
  - password
  - letmein
  - abc123
  - qwerty
  - abcdef
  - 1234567890
  - Yours or another persons name
  - Dates or calendar months
  - Curse words/sexual slang

Creating Complex Passwords

- Firefox and Chrome will suggest complex passwords when you are filling in an account creation form and they'll save the resulting password
  - You can't control the length or complexity
- Most password managers will also offer to create passwords
- On the Linux CLI, `pwgen` or `makepasswd` will create random passwords for you, you can pass complexity options
  - `pwgen -sync 17`
- Don't expect to remember them

Don’t Make It Too Complicated

- The hardest passwords to crack will be more than 16 characters long with upper and lower case letters, numbers and special characters
  - But you won’t be able to remember them
- If you enforce a strict password policy which requires a new password every 3 months, people will use variations or write them down
  - I’ve faced policies where with my understanding of password complexity issues, I could not create a valid one

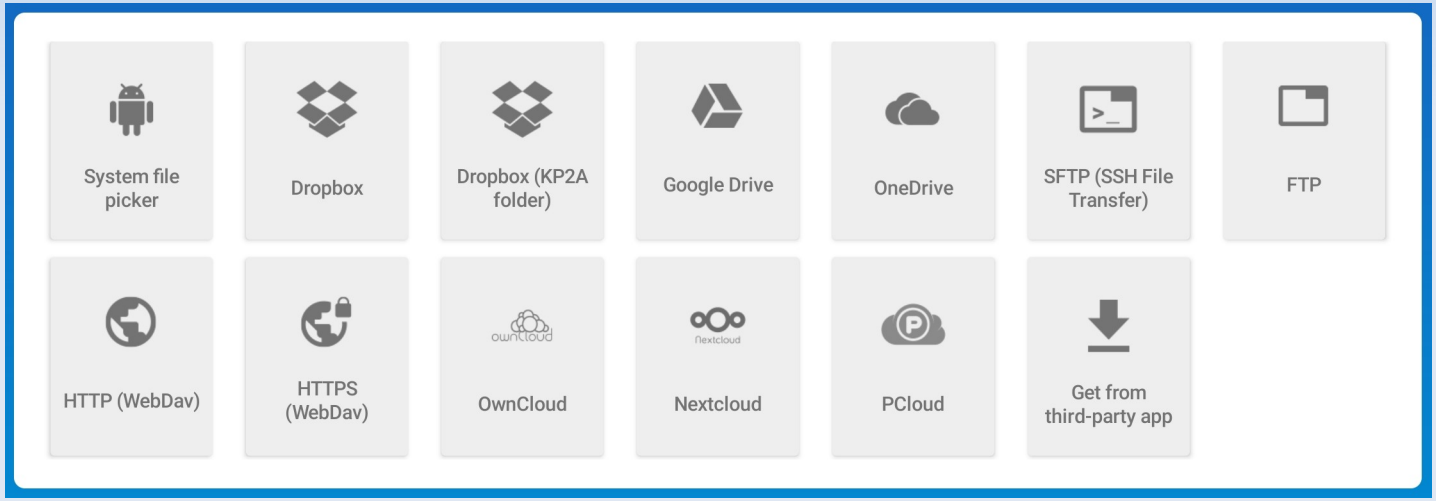
correct horse battery staple

- https://xkcd.com/936/
- Using dictionary words, replacing some letter with numbers works, but can be easily mixed up
  - `Tr0ub4or&3`
- Perhaps just using a couple of easily memorable words one after another is complicated enough to make it take too long to crack
  - `correct horse battery staple`
  - Do not use this password, it's a well known example

Password Managers

- Pretty much every browser has a password manager these days
  - Anyone who has access to your desktop has access to all your passwords
  - Firefox allows you to set a master password
  - On Windows, Chrome uses your Windows password
- Web services
  - Last Pass – recently limited free tier
  - Bitwarden
- Desktop/Mobile – Sync with Nextcloud, GDrive, OneDrive, Dropbox etc
  - Desktop – KeepassXC (not KeePass or KeePassX)
  - iOS – Strongbox, Keepassium
  - Android – Keepass2Android, KeepassDX
  - Back it up!

KeePass2Android Storage Support



Password Re-Use

- We've all used the same username, email address and password for everything
- A break-in at one site means if your password hash is stolen and cracked (or it was stored in plaintext) the attacker knows it might be used on other sites
- If an attacker has your email address and a password, what could they access?
- Use you LinkedIn or Dropbox password for your email/Facebook/Twitter/dating?
- Chrome/Firefox now tell you if an account for a site has been found in a leak
  - If they do, go change your password
- Use a different password for every site, your browser can suggest and store
- Change any passwords where you use the same username/email address and password as those found in a breach
- Passwords can still get stolen/cracked
- Use 2 factor auth everywhere you can

Have You Been Pwned?

- haveibeenpwned.com is a great resource
- It collects data on website breaches where the user credential database was stolen
- You can search for your email addresses to see if they have appeared in any breaches
- Change your password on any websites where your account showed up in a breach
  - Or close your account
- And any site where you use the same username/email address and password combination
- Register your email addresses on haveibeenpwned.com to be notified when they appear in newly discovered breaches

Summary

- Password cracking is relatively trivial with affordable hardware
- Don’t be the easy pickings in a stolen password database
- The more complex your password the better
  - 10 or more upper and lower case, digits and special chars
- Use an encrypted password manager
  - KeepassXC or Bitwarden
- Let your password manager or browser create and store unique passwords for each website/application
- Register your email addresses on haveibeenpwned.com
- Change your password whenever you are notified your credentials have been found in a breach