Practical Password Cracking

Hardware, Tools, Methods... and AI

Sean T Smith

April 2024

whoami

Name	Sean T Smith
Hacker Handle	smidiculous
Occupation	Smart Factory IIoT Consultant
Another Occupation	Cyber Team Lead
Yet Another Occupation	Adjunct Professor of Cybersecurity
Family	Wife, 2 Kiddos
Education	Undergrad, Grad, Google, ChatGPT
Hobbies	Tech, Hardware, Building Stuff, 3D Printing, Pretending to Code
My Social Media Links	**www.seantsmith.me**

Agenda

- Introduction
- Password Cracking Factors & Considerations
- Overview of Password Attack Types
- Recommended Cracking Rigs
- The Methodology & Experiment
- Tools & Resources for Password Attacks
- Walkthrough & Results
- **Demo**
- Final Thoughts

What are we learning today?

Password cracking is an exercise in maximizing guesses per unit of time.

- This presentation will focus on learning a **methodology and mindset** which can be applied using many different tools across all password types.
- Expensive GPU(s) and Cloud Infrastructure is not required (although it helps); **most passwords can be cracked with a laptop.**
- Bottom line, password cracking is not difficult if you understand the process, I have spent the last 5 years refining my approach during **real-world assessments.**
- Time to **share what I've learned!**

ChatGPT Hardware

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	Instantly	Instantly	Instantly
8	Instantly	Instantly	Instantly	Instantly	1 secs
9	Instantly	Instantly	4 secs	21 secs	1 mins
10	Instantly	Instantly	4 mins	22 mins	1 hours
11	Instantly	6 secs	3 hours	22 hours	4 days
12	Instantly	2 mins	7 days	2 months	8 months
13	Instantly	1 hours	12 months	10 years	47 years
14	Instantly	1 days	52 years	608 years	3k years
15	2 secs	4 weeks	2k years	37k years	232k years
16	15 secs	2 years	140k years	2m years	16m years
17	3 mins	56 years	7m years	144m years	1bn years
18	26 mins	1k years	378m years	8bn years	79bn years

HIVE SYSTEMS 2023 › Learn how we made this table at hivesystems.io/password

https://www.hivesystems.com/blog/are-your-passwords-in-the-green

Key Hive Systems 2023 Chart Assumptions

1. *Offline Cracking of MD5 Hash*
2. *Randomly Generated: a-z, A-Z, 0-9, ^*%$!&@#*
3. *Black Box, Password Not Previously Breached*

How is this useful?

Nothing makes a bigger impact during a cyber assessment than SHOWING the results of poor security practices; password cracking is an easy and useful tool for this.

- Penetration testing requires **initial access and privilege escalation**, frequently this is enabled by cracking captured password hashes.
- Successfully **cracking password hashes can make the difference** between a successful or failed assessment.
- Auditing an organizations NTDS.dit file passwords can **confirm technical controls** of the password policy, or lack thereof!
- The skill of password cracking in a time/resource constrained setting is **critical for any cyber operator**.

How does this help the good guys?

A compelling readout can successfully advocate on behalf of struggling corporate cybersecurity.

- **Presenting compelling engagement results** to the customer's leadership is intensely effective.
- This often leads to swift organizational changes that **improve cybersecurity focus and resourcing.**
- **Non-technical leaders can easily relate** to exploiting passwords and understand the real-world impact.
- It is for this reason that honing your password cracking craft is a **critical skill** for any Cyber Operator.

The Password Cracking Mindset

A pragmatic approach to maximize your cracking probability.

Password Cracking is a Planned Mission

- **Embrace Complexity**
  Password cracking involves understanding and navigating complex systems. Recognize that each password and security system presents a unique challenge, requiring a thoughtful tailored approach.
- **Start by Analyzing the Variables**
  These include the type of hash, password strength, available hardware, time constraints, etc. Successful solvers methodically assess these factors before starting.
- **Develop a Plan**
  After understanding the variables, develop a plan based on the data. Decide the types of attacks and their order ahead of time so that you quickly "pick the lowest hanging fruit" first.

Work Smarter, Not Harder

- **Leverage Tools & Techniques**
  Familiarize yourself with the plethora of tools and techniques to understand which are best suited for your scenario. Efficiency comes more from using the right tools for the job, not just the most powerful cracking rig.
- **Understand the Target**
  Understand the target system or password's nature. For example, knowing that a password follows a certain format or that the system has rate limiting can significantly influence your approach and increase efficiency.
- **Prioritize Targets**
  Prioritize your efforts based on the potential gain and the likelihood of success. This may mean focusing on weaker passwords first or targeting those that grant access to more valuable data.

Patience & Persistence

- **Trust Your Planning**
  Follow the plan, even if you don't see success immediately; knowing the time to pivot comes with experience.
- **Learn Continuously**
  Successful crackers make a habit of learning from each attempt, whether successful or not, and incorporating that information into the future attempts.

"If I had a nickel for every password guess made by my cracking rigs during this research, I would have 18,222,411,116,041,000 nickels... and if that was converted into a stack of $100 bills, it would stretch from the Earth to the Moon and back!"

-Sean Smith

…assisted by ChatGPT

Password Cracking Difficulty Factors

Sean's Equation for 'Crackability'

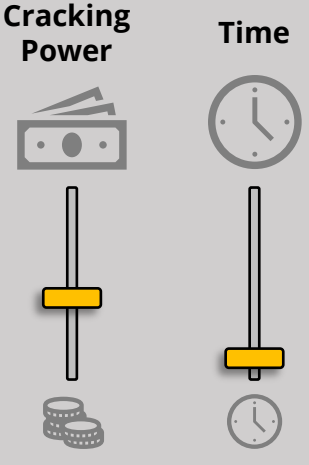
$$Crackability = \frac{Amplifiers \times Attack\ Levers}{Password\ Composition \div Human\ Factors}$$

PATENT-PENDING

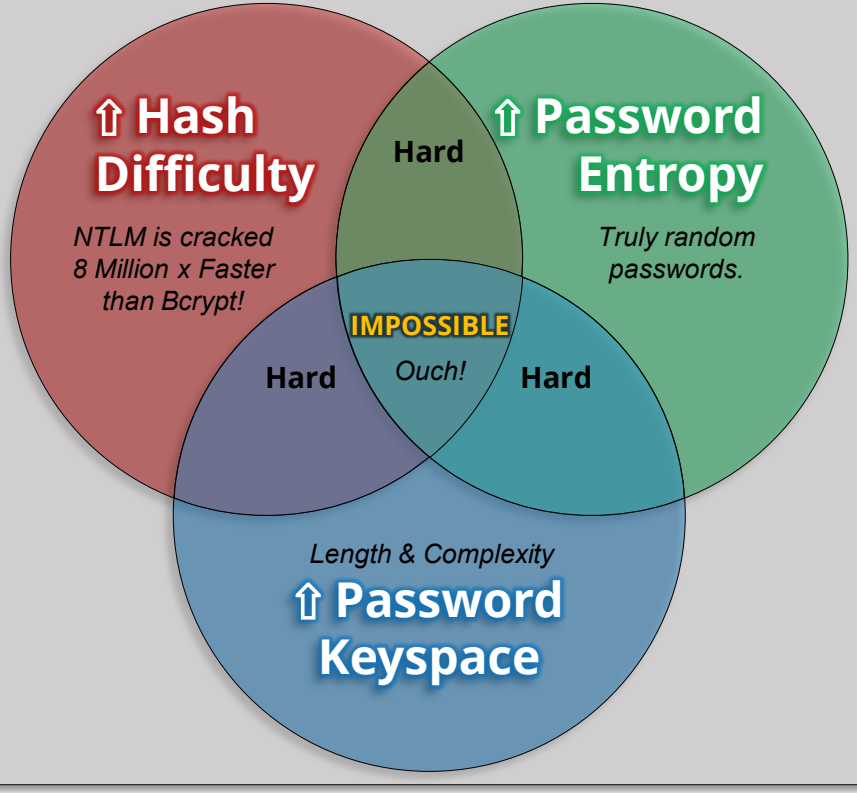
Amplifiers

- ☑ Tools & Process
- ☑ Tailored Wordlists
- ☑ Masks & Rules
- ☑ AI Models

Attack Levers



Password Composition

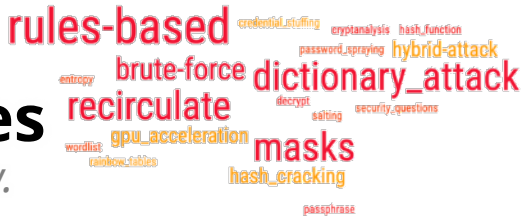


Human Factors

- ☑ Default Password
- ☑ Previously Breached
- ☑ Password Reuse
- ☑ Common Password
- ☑ Personal Information
- ☑ Pattern Following
- ☑ Dictionary Words
- ☑ Cultural References
- ☑ Keyboard Walks
- ☑ Simple Substitutions

Overview of Our Password Attack Types

The Hashcat attacks which underpin the research are listed below.



Base Hashcat Command

{BaseCommand} = hashcat.exe -d 1 -m 1000 -w 4 --username --backend-ignore-hip --bitmap-max 28 --session MySession -o "C:\Users\Sean\Output\output.txt"

Hashcat Binary | GPU Device | Hash Type | Work Factor | Ignore Username | Ignore AMD Error | Password Dictionary | Increase Memory | Attack Mode | Session Name | Output File | Hash List

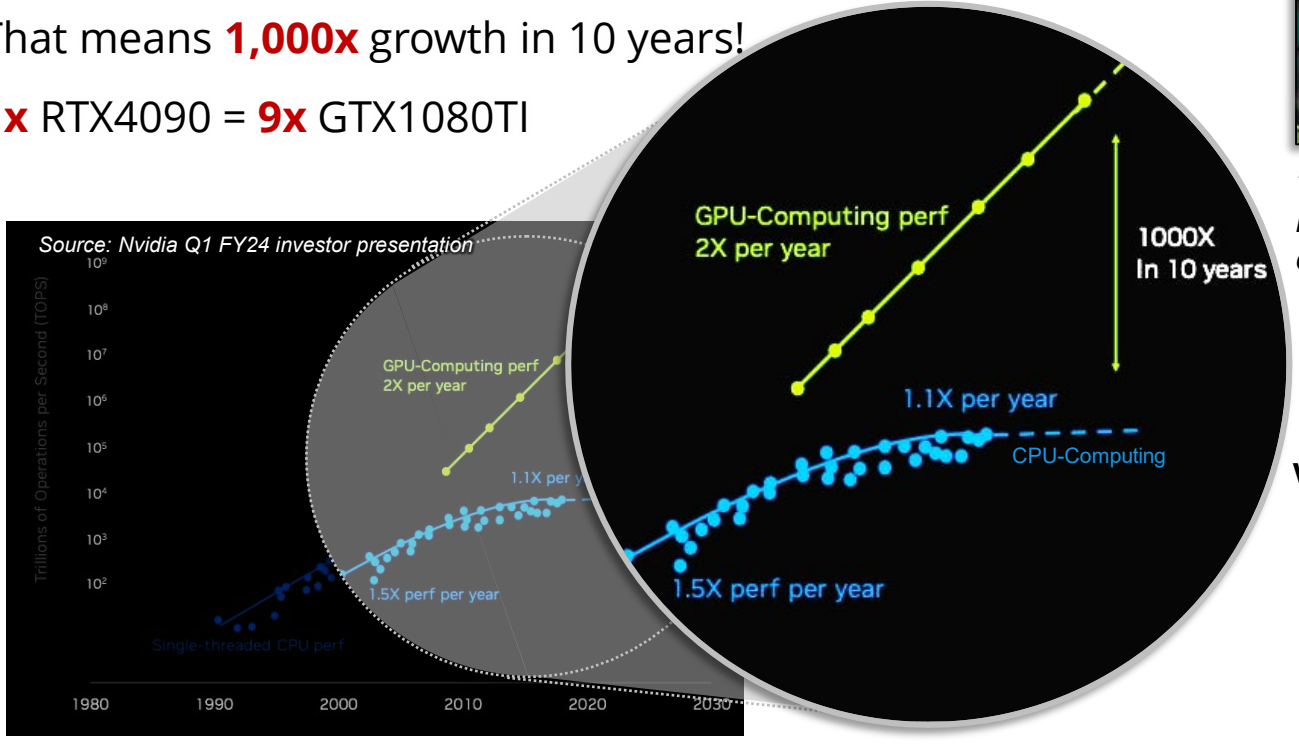
Attack	Description	Example
Dictionary	Read from a text file (aka 'dictionary' or 'wordlist') and try each line as a password candidate.	{BaseCommand} -a 0 "C:\Users\Sean\Hashes\password_hashes.txt" "C:\Users\Sean\Dictionaries\dictionary.txt"
Brute-Force	Tries all combinations from a given keyspace (1-8 characters).	{BaseCommand} -a 3 --increment --increment-min=1 "C:\Users\Sean\Hashes\password_hashes.txt" ?a?a?a?a?a?a?a?a → Single Mask
Masks	Try all combinations from a given keyspace, similar to a Brute-Force attack, but more specific.	{BaseCommand} -a 3 "C:\Users\Sean\Hashes\password_hashes.txt" "C:\Users\Sean\Masks\masks.hcmask" → Mask List
Rules	The rule-based attack is a scripting language which has functions to modify, cut or extend words to generate additional password candidates.	{BaseCommand} -a 0 "C:\Users\Sean\Hashes\password_hashes.txt" "C:\Users\Sean\Dictionaries\dictionary.txt" -r "C:\Users\Sean\Rules\rules.rule" → Rule Script
Recirculate	Apply rules to previously cracked passwords to generate new password candidates.	{BaseCommand} -a 0 "C:\Users\Sean\Hashes\password_hashes.txt" "C:\Users\Sean\Cracked\cracked.txt" -r "C:\Users\Sean\Rules\rules.rule"
LLM-Dictionary	Generate a dictionary of password candidates using a trained LLM model.	{BaseCommand} -a 0 "C:\Users\Sean\Hashes\password_hashes.txt" "C:\Users\Sean\LLM_Model\llm_dictionary.txt"

Note: *Do not use the [-O \| --optimized-kernel-enable] switch if you need maximum coverage, instead use the default Pure Kernel.*

Sourcing GPUs: Which to buy?

Buying a single new latest generation graphics card is far better than gambling on used-and-abused older generation cards; > Moore's Law.

- GPU Performance has **doubled each year!**
- That means **1,000x** growth in 10 years!
- **1x** RTX4090 = **9x** GTX1080TI



"But how do I get one for retail, they are all overpriced or sold out?"

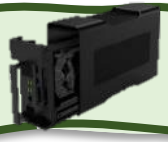
-Everyone

www.nowinstock.net
+
Patience

Choosing a Cracking Rig

This is my favorite part, but it is actually the least important aspect.

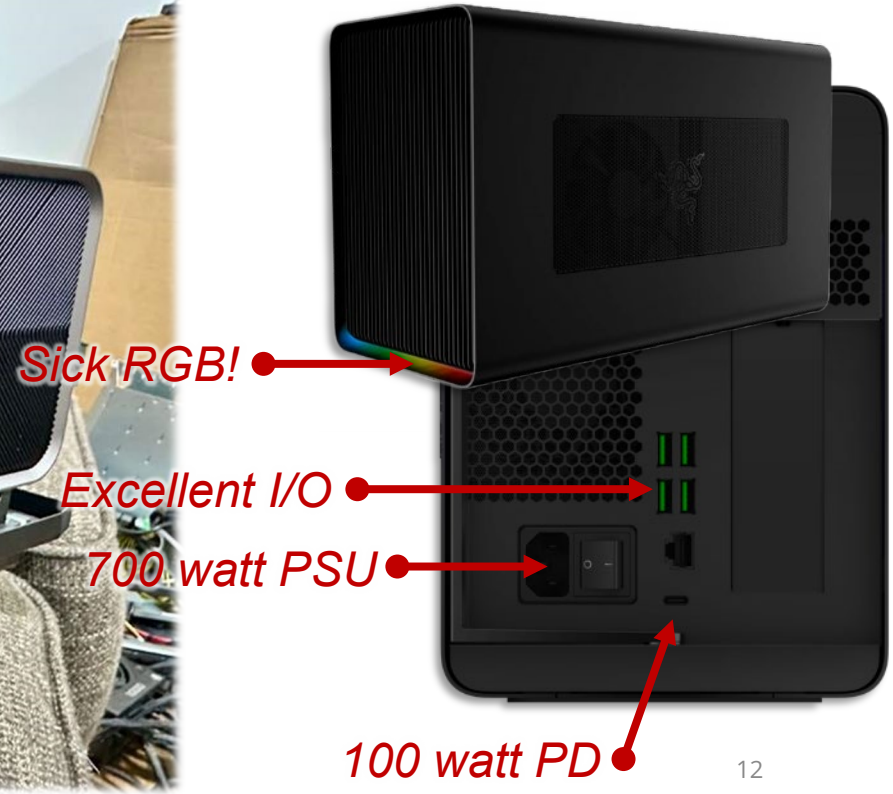
My Pick

Potential Cracking Rigs	Pros	Cons
 **External Graphics Cards (eGPUs)**	+ Multi-Purpose + Dead Simple to Setup + I Had Them Already	– Supports fewer total GPUs – eGPU enclosures are $ – Thunderbolt can be unstable
 **Repurposed Mining Rig**	+ Compact Form Factor + Supports Many GPUs + Built for Stress	– Not Intended for Cracking – Weak Compute – Minimal RAM & Storage
 **Use a Gaming Desktop**	+ Built for Graphics Stability + Solid Storage & Compute + Cool RGB Colors	– **I don't have one!** – My friend wouldn't stop gaming on his so I could use it.
 **Cloud Infrastructure**	+ Highly Scalable + Easy to Provision Resources	– **Expensive** for my use case. (more on that later)
 **Online Password Cracking Service**	+ Zero Setup	– **Expensive and does not support my use case.**

External Graphics Cards (eGPU)

I've had a lot of success with eGPUs and I value their balance between simplicity, power, flexibility, and portability; connect it to your laptop when you need it! It's even (barely) compatible with the RTX4090 … after some slight modifications!

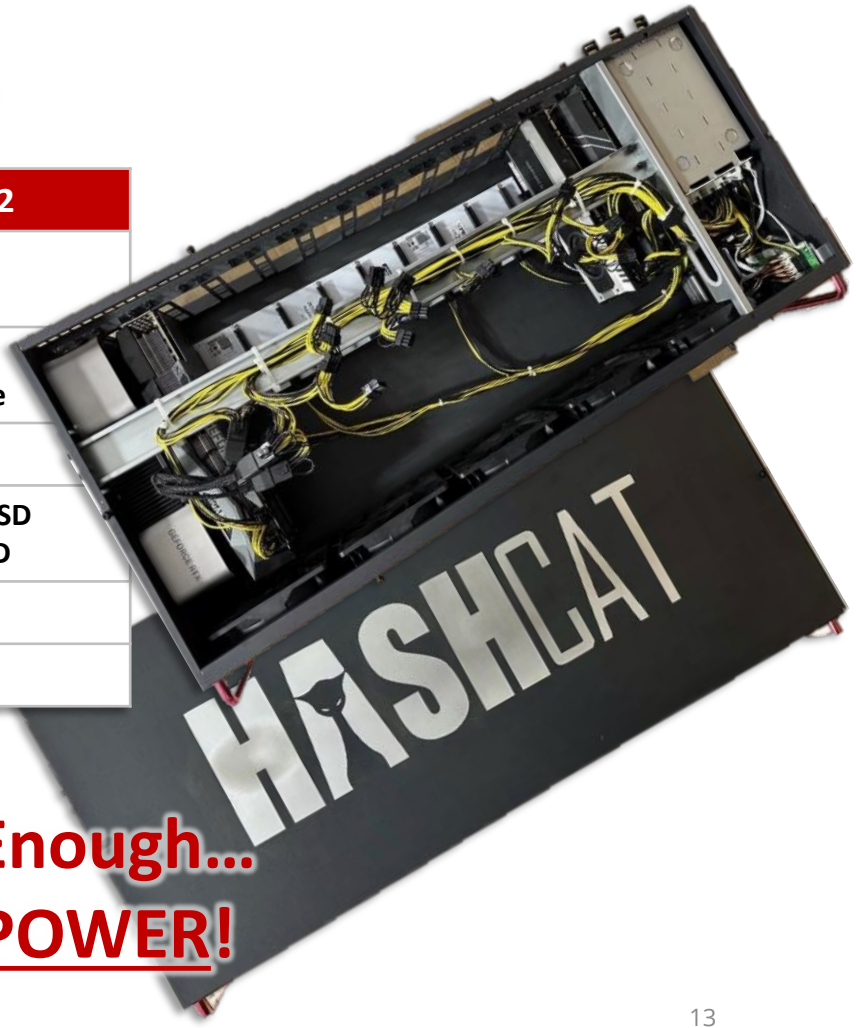
Razer Core X Chroma



Upgraded Mining Rig

MinerDude X12 XTREME = Octominer X12 ULTRA

Specs	Default X12	Upgraded X12
Motherboard	Default w/ PCIe Riser Extension	
Processor	Intel G1840 Celeron 2.8Ghz 2-Core	**Intel i7-7700T** **2.9Ghz, 4-Core**
Memory	8GB DDR3	**32GB DDR3**
Storage	60GB SATA SSD	**250GB MSATA SSD** **+ 4TB SATA SSD**
Power Supply	3600W@120v | 4200W@240v	
Cooling	5x 140x38mm Fans	



Still Not Good Enough…
NEEDS MORE POWER!

Read this, and don't ignore it like me… https://hashcat.net/forum/thread-11041.html

FUTURE UPGRADE: Transformed Mining Rig

Replace X12 Motherboard with Biostar TZ590-BTC DUO

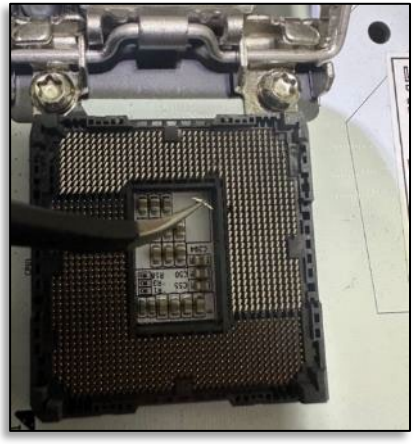
Specs	Default X12	Upgraded X12	X12 → TZ590-BTC
Motherboard	Default w/ PCIe Riser Extension		**Biostar TZ590-BTC DUO w/ External PCIe Risers**
Processor	Intel G1840 Celeron 2.8Ghz 2-Core	**Intel i7-7700T 2.9Ghz, 4-Core**	**Intel i9-11900T 4.8Ghz, 8-Core, 16-Thread**
Memory	8GB DDR3	**32GB DDR3**	**128GB DDR4**
Storage	60GB SATA SSD	**250GB SATA SSD + 4TB SATA SSD**	**512GB NVME + 4TB NVME**
Power Supply	3600W@120v | 4200W@240v		
Cooling	5x 140x38mm Fans		

=== Key Parts List ===

- Biostar TZ590-BTC DUO
- 12x External PCIe Risers
- Intel i9-11900T
- Noctua NH-L9i Processor Fan
- 4x G.SKILL TridentZ DDR4-3600 32GB
- Samsung 990 Pro 4TB
- Silicon Power A55 4TB

Embrace the Hard(ware)ship.

Mistakes were made, things go wrong, the home office gets hot. You can learn from every experience, especially bad ones… and wish you just did this in the Cloud.



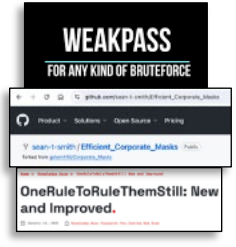
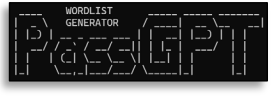
Dropped Power Supply on Exposed Socket: **Busted Pin**

Purchased shady used GPUs on eBay/Facebook: **Dirty, Abused & Glitchy**

Cracking in the Home Office 24/7: **Sweaty Zoom Meetings**

The Experiment

Setting the stage to conduct the research.

Goal	**To improve upon previous methods to crack unknown 'hard' passwords.**
Hypothesis	If a Large Language Model (LLM) is trained on an existing set of breached plain text passwords, it will then generate a wordlist of novel but highly probable passwords capable of cracking the difficult and uncracked 9-16 character passwords.
High-Level Approach	1. Identify a test hash dataset. ▼ • Have I Been Pwned Downloader  2. Select and gather tools. ▼ • Hashcat • Cracking Rigs  3. Use traditional approach for cracking 'easy' passwords. ▼ • Brute-Force • Wordlists • Masks • Rules  4. Leverage LLM to crack 'hard' passwords. ▼ • LLM Dictionary  5. Analyze results and present findings. ▼ • Python • MS Excel • PowerPoint 

Identify a test hash dataset.

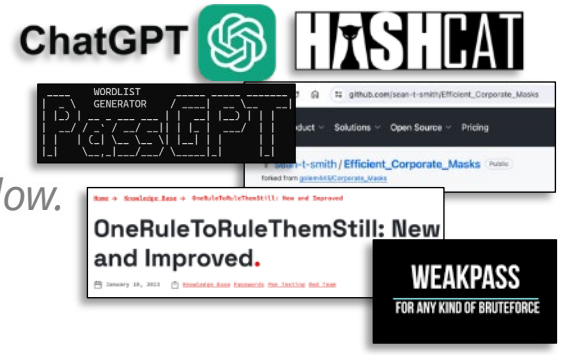
HaveIBeenPwned.com provides the ideal test dataset because it is clean, contains real passwords as NTLM hashes, and is extremely quick and reliable to download.

	Description
Name	Have I Been Pwned (HIBP)
Source	HIBP dataset rolling dataset of breached password hashes. This dataset is the most comprehensive list of public / non-public data breaches from various sources including the FBI. Included along with the hashes are the associated occurrences across breaches which can be used to create interesting outputs.
Data Location	https://haveibeenpwned.com/Passwords
Date D/L	1/21/2024
File Size	31.3GB (uncompressed)
Total Hashes	931,855,073
Hash Type	NTLM
Format	<NTLM_HASH>:<#_OCCURRANCE>

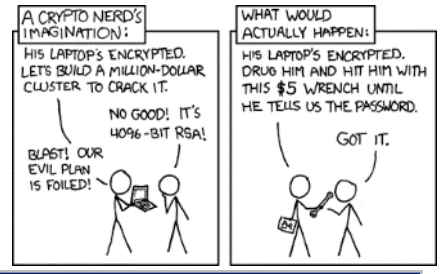
HIBP Sample Data

Select and gather [software] tools.

The tools & resources which underpin the research are listed below.



Tool / Resources	Link	Description
Hashcat	• https://hashcat.net/hashcat • https://hashcat.net/wiki • https://hashcat.net/wiki/doku.php?id=frequently_asked_questions	The best resource for Hashcat related information is the Hashcat Wiki and FAQ. I recommend reading the both from beginning to end. Hashcat also has verbose errors, if you see an error don't ignore it, resolve it!
Rules	• https://in.security/2023/01/10/oneruletorulethemstill-new-and-improved • https://github.com/stealthsploit/OneRuleToRuleThemStill	An improved version of the very popular "OneRuleToRuleThemAll" with improved performance and without any reduction in efficiency.
Masks	• https://github.com/sean-t-smith/Efficient_Corporate_Masks/raw/master/hcmask_Generator_9000.xlsx	A personal project which contains a sortable filterable list of prioritized Hashcat masks. Use this to generate .hcmask files to perform targeted mask attacks.
Wordlists	• https://weakpass.com • https://weakpass.com/all-in-one	This site is the best repository for high-quality wordlists. The wordlists used in my research are: Top304, brockyou, hk_hlm_founds, hashkiller-dict, HashesOrg, weakpass_3, All-in-One
LLM Model	• https://arxiv.org/abs/2306.01545 • https://huggingface.co/javirandor/passgpt-10characters • https://huggingface.co/javirandor/passgpt-16characters	PassGPT is a causal language model trained on password leaks. It was first introduced in this paper (https://arxiv.org/abs/2306.01545). This version of the model was trained on passwords from the RockYou leak, after filtering those that were at most 16 characters long.
ChatGPT	• https://chat.openai.com	Manipulating large datasets requires specialized knowledge of specific tools: PowerShell, Python, sed/awk… but as you know ChatGPT can help with pretty much anything.



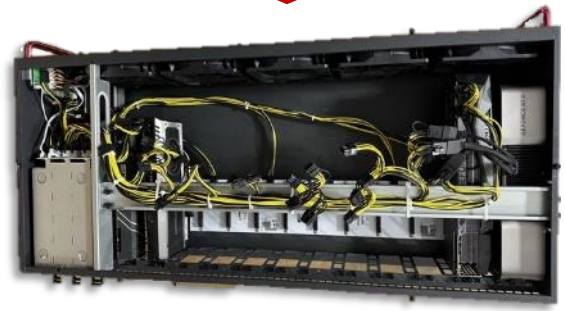
Select and gather [hardware] tools.

The tools & resources which underpin the research are listed below.

Framework 13 (AMD)	
Laptop	**Description**
Model	Framework Laptop 13
Processor	Ryzen 7840U, 8-Core
RAM	64GB DDR5
SSD	4TB NVME
eGPU Case	Razer Core X Chroma
GPU 1	RTX 4090

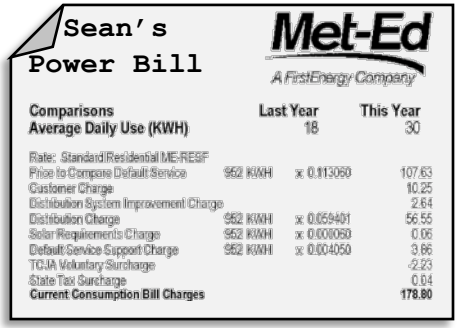
NUC Hades Canyon	
Laptop	**Description**
Model	NUC8i7HVK
Processor	Intel i7-8809G 4-Core
RAM	64GB DDR4
SSD	512GB NVME
eGPU Case	Razer Core X
GPU 1	RTX 3080TI

Octominer x12 Ultra	
Laptop	**Description**
Model	MinerDude X12 XTREME
Processor	Intel i7-7700T, 4-Core
RAM	32GB DDR3
SSD	4TB SATA
GPU 1	RTX 3080
GPU 2,3,4	3x GTX 1080



Password Cracking Walkthrough

Sweating in the office sauna for 52 days!



0 **Download**
HIBP Hash List

1 **Dictionary**
Breach Compilation + COMB

2 **Dictionary**
hashes.org HIBP Plains List v1-v7

3 **Brute-Force**
1-7 Characters

4 **Brute-Force**
8 Characters

5 **Masks**
Targeted
9 Characters

6 **Masks**
Targeted
10 Characters

7 **Masks**
Targeted
11-12 Characters

8 **Brute-Force**
13-14 Digits

9 **Rules**
Common Wordlists + OneRule
hashkiller-dict
HashesOrg
weakpass_3

10 **Recirculate**
+ OneRule
3x
Top304
brockyou
hk_hlm_founds

Easy

Hard

11 **Dictionary**
Weakpass All-In-One

12 **Rules**
Weakpass All-In-One + OneRule

13 **Recirculate**
+ OneRule
4x

14 **LLM-Dictionary**
PassGPT LLM

15 **Dictionary**
PassGPT LLM + OneRule

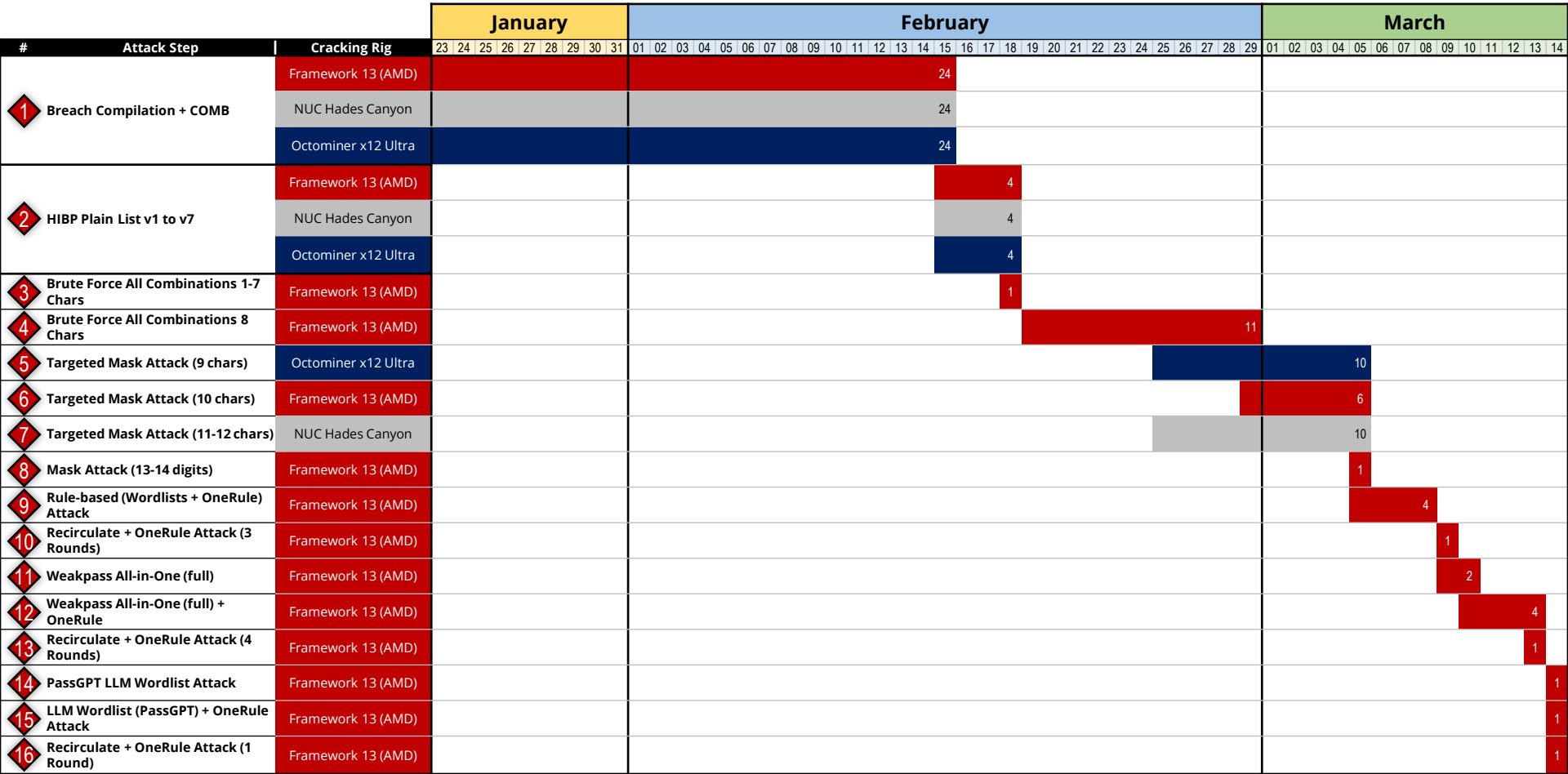
16 **Recirculate**
+ OneRule
1x

Password Cracking Timeline

The rigs were used for 130 total compute days which would have cost this much with AWS!

AWS Cloud EC2 Equivalency — *As of 3/20/24*

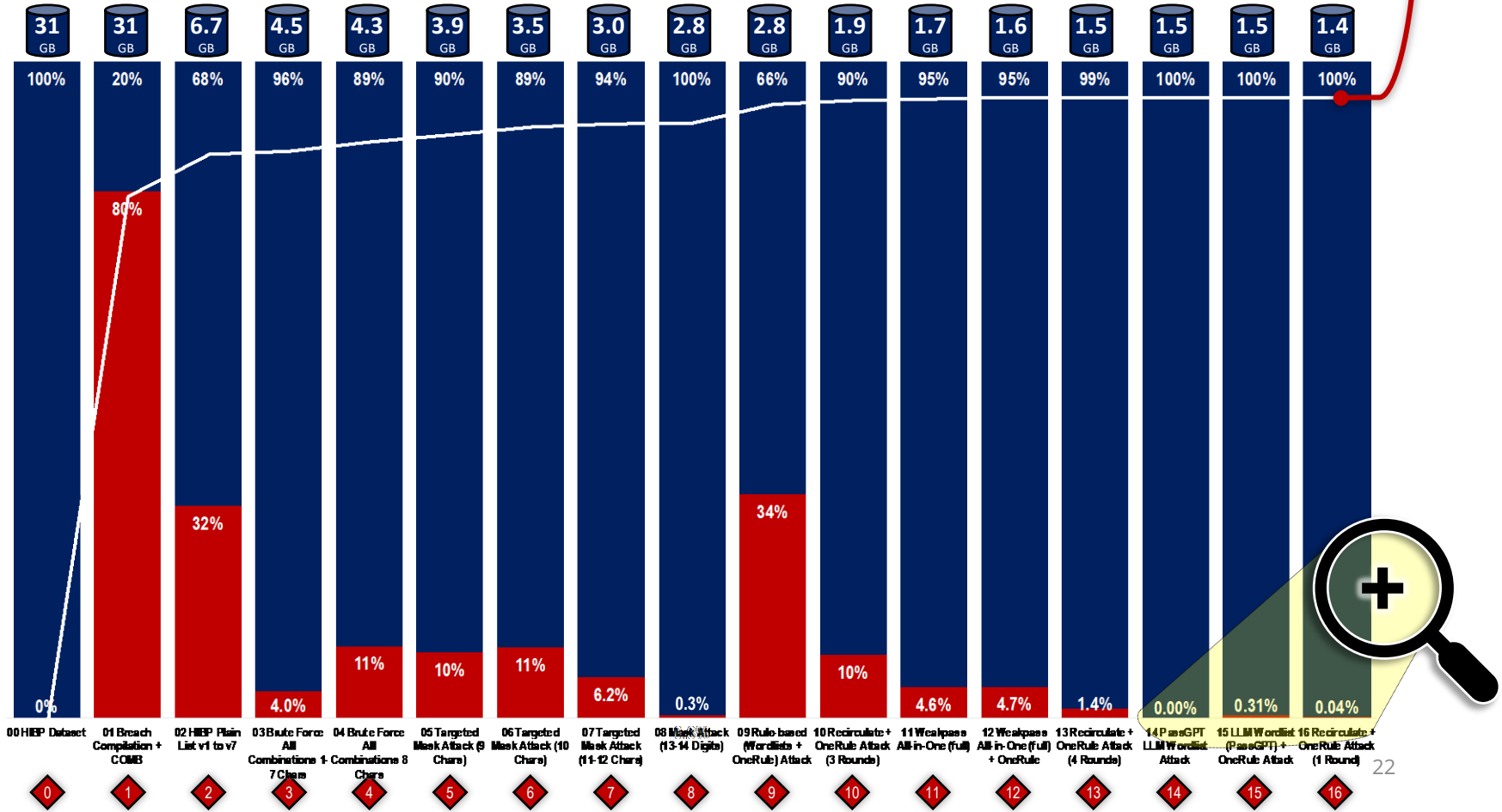
EC2 Type	$/Day	Days	Total Cost
g5.4xlarge	$38.98	56	$2,182.66
g5.4xlarge	$38.98	37	$1,442.11
g5.2xlarge	$29.09	37	$1,076.26
https://aws.amazon.com/ec2/instance-types/g5			**$4,701.02**



Summarized Results

The methodology cracked over 95% of the Have I Been Pwned hash dataset!

$$\frac{888{,}921{,}267\ \textit{Cracked Hashes}}{931{,}855{,}073\ \textit{Total Hashes}} = 95.39\%$$



Was the LLM-Dictionary attack successful?

The LLM-Dictionary attack cracked more passwords per total guesses than the Weakpass All-in-One attack, even being last in the attack order!

#	Phase	Attack Keyspace	Uncracked	%	Size	Cracked	%	Size
0	HIBP Dataset	-	931,855,073	100.0%	31.3GB	-	0.0%	0.0GB
1	Breach Compilation + COMB (102GB)	8,775,988,939	184,698,787	19.8%	6.7GB	747,156,286	80.2%	33.2GB
2	HIBP Plain List v1 to v7 (8GB)	624,156,006	124,991,382	13.4%	4.5GB	59,707,405	6.4%	36.2GB
3	Brute Force All Combinations 1-7 Chars	70,576,641,626,495	120,052,757	12.9%	4.3GB	4,938,625	0.5%	36.4GB
4	Brute Force All Combinations 8 Chars	6,634,204,312,890,620	107,004,666	11.5%	3.9GB	13,048,091	1.4%	37.0GB
5	Targeted Mask Attack (9 chars)	2,663,175,327,420,910	96,407,321	10.3%	3.5GB	10,597,345	1.1%	37.6GB
6	Targeted Mask Attack (10 chars)	3,047,124,553,245,060	86,112,207	9.2%	3.0GB	10,295,114	1.1%	38.3GB
7	Targeted Mask Attack (11-12 chars)	4,039,055,493,990,400	80,776,718	8.7%	2.8GB	5,335,489	0.6%	38.5GB
8	Mask Attack (13-14 digits)	110,000,000,000,000	80,531,094	8.6%	2.8GB	245,624	0.0%	38.5GB
9	Rule-based (Wordlists + OneRule) Attack	161,864,881,597,555	53,178,940	5.7%	1.9GB	27,352,154	2.9%	39.7GB
10	Recirculate + OneRule Attack (3 Rounds)	85,267,558,724,478	48,080,063	5.2%	1.7GB	5,098,877	0.5%	40.0GB
11	Weakpass All-in-One (full)	28,330,556,550	45,877,005	4.9%	1.6GB	2,203,058	0.2%	40.1GB
12	Weakpass All-in-One (full) + OneRule	1,371,595,564,811,700	43,718,518	4.7%	1.5GB	2,158,487	0.2%	40.3GB
13	Recirculate + OneRule Attack (4 Rounds)	238,916,699,352	43,087,510	4.6%	1.5GB	631,008	0.1%	40.3GB
14	PassGPT LLM Wordlist Attack (8.64GB)	810,980,079	43,087,251	4.6%	1.5GB	259	0.0%	40.3GB
15	LLM Wordlist (PassGPT) + OneRule Attack	39,262,789,544,706	42,952,553	4.6%	1.4GB	134,698	0.0%	40.4GB
16	Recirculate + OneRule Attack (1 Round)	6,533,808,198	42,933,806	4.6%	1.4GB	18,747	0.0%	40.4GB
	OVERALL RESULTS:	**18,222,411,116,041,000**	**42,933,806**	**4.61%**	**1.44GB**	**888,921,267**	**95.39%**	**40.41GB**

Weakpass All-in-One *Steps 11 + 12 + 13*	
Dictionary Size	329GB
Cracked	4,992,553
Keyspace	1,371,862,812,067,600
Guess Efficiency *(Cracked/Keyspace)*	3.64E-09 **-7%**

VS

PassGPT LLM *Steps 14 + 15 + 16*	
Dictionary Size	8.64GB
Cracked	153,704
Keyspace	39,270,134,332,983
Guess Efficiency *(Cracked/Keyspace)*	3.91E-09 **+8%**

A small peak into some cracked hashes.

Here is the proof -- the model successfully cracked never-before-cracked hashes!

Most Popular | Passwords

Top 15

1.	42542807:123456	6.	5070941:111111	11.	4034851:abc123
2.	18313580:123456789	7.	4880569:qwerty123	12.	3897129:123123
3.	10713794:qwerty	8.	4486025:1q2w3e	13.	3663338:NULL
4.	10382543:password	9.	4351342:1234567	14.	3561705:DEFAULT
5.	6901438:12345678	10.	4130502:1234567890	15.	3508324:12345

Most Popular | Mask Patterns

Top 10

1.	481526598:?l?l?l?l?l?l?l?l	6.	205251980:?l?l?l?l?l?l?d?d
2.	321100471:?l?l?l?l?l?l	7.	144530489:?l?l?l?l?l?l?l?l
3.	319361204:?d?d?d?d?d?d	8.	126303027:?l?l?l?l?l?l?l?d
4.	225430345:?l?l?l?l?l?l	9.	123240391:?l?l?l?l?l?l?l?l?l
5.	215320674:?d?d?d?d?d?d?d	10.	122936007:?l?l?l?l?l?l?d

1. PassGPT Wordlist

259 Cracked

```
e95fbaa8fafac04d0141a42c717fc541:burgosfresa
f5244f78c0757cb2babc95b44382ed98:MorbidDiana
e17f17609ab95ef91231b2c4b85bac49:LASVEGAS&ME
577d9c1e13f042740d84b47ac7c48180:ABCabc123ert
55ad51225157a4aff52205b2448d7bbd:eulogiomateo
44e949e8e4d7c6633e38b6c5843b183f:12_sept_1987
11edda3c45bb636df4a5e43e779f37f8:amoremiopaulo
357d8f9c5d0f96653b7e7cf3ecd0ff26:1968ScoobyDoo
69f938370dbfc0090a5162f0fa32dce8:hidalgoferrer
0e62a5b96e4831f302912e192d372651:blazeandlottie
45c8710982905da7305fae54d1c86582:camiloesunbobo
b3d9b4c4ef945e047f345c321746dc09:milagrosteodio
7f58bfeb7be4ecb6d8b08c3b936a7653:figueroaventura
76153fc02710dbb8508e2dd5f580478e:iliketosucktoes
45f1b0807997acfa2b4af1d01f19dda3:medranotequiero
c5f6ae53328705d7e1ab1f22ee72cdeb:queenofdangerous
1e88268ad6af32bbb9b79a31338e626f:zacefronxsiempre
a5c1acba5783404f8ddf14e3eaad0b23:12369874!@#$%^&*
```

2. PassGPT Wordlist + OneRule

134,698 Cracked

```
82e3aeb7bc2565e9e0317cbc465563c8:mulba23re
058fbc9de5c6ab6a4399345455c6e35b:sunnySKE11
9c1c51026d6308e475bb0bf38a47b2f7:killerLSA23
214fd681c216f84cb7758c0ffb3b01cf:leandroMSP89
1333553437499be7ffd327c0a564ec76:facebookLoL15
bc31f1e63c9bae69ff5cfb950809a658:computeritb747
7cdc17db713fbfa645ff95308a5ced6e:alexandre09tuna
515fe0444b00ba04fddfae5431e6038e:alma1**********
d8080ebac595632350a90ffeb429d2cb:taylorswiftjcww12
9097daf5de29e2adfbbe461130d9c796:yosoyunkkkkkkkkkk
545d4e6c4c30cc1613fe6ad23e3c346f:mortalkombats192903
f920d33ba4efd7a80332c5d34b4be6a0:lamborghini700spyder
4c7eae92597f53c0fdaf6424aea19c8e:123456789saramagnante
5568d1983f5d9aa88f7d915b7e0e74a5:ssaajjaa112233445566**
dde00f08e3d32f205ca705905cad4e56:paradiseloser1234567890
88a231361ba96e0310b73c257807a1d0:undertaker30wrestlemania
6fef3ca1a479d7df4938195ce83ed73c:policeisawesome1234567890
e44943afe2dbc7361a2b29e63921d56b:ilovememeloveyou1234567890
500d718e9513dd1b862cd82eb46969df:01457816301457816301457816 3
269333027e5dbd338fc11ed77c808d6b:Diosesamor2324Diosesamor2324
bc75daf180ae8932cac63849572c6be6:mmp123456789l0mmp12345678910
5d935bac3b260329de4292a368518301:0868305325086830532508683053 25
a367dc2ae9a963d696d19238ac33d887:16septiembre200416septiembre2004
3c636198145c5d256bfd179d66352ac1:4nn2nn34nn2nn34nn2nn34nn2nn34nn2nn3
f6a36bdb507192df6e0a14efdcdb9c98:arielminiarielminiarielminiarielmini
be490ad4021821bdde2445ec432e5809:00121019110012101911001210191100121019 11
```

3. Recirculate + OneRule

18,747 Cracked

```
19890b1ab584f482338bc7cd5f4efd38:Ss@123=12
991bdf8334f6821fca9258e8949a2ad8:Lisa*emma2
ebf6cdde011b455f1f1b3048e5d9ad63:Kimshgood1!
b347a9e099a0b80de4b259aedcffaaee:kkisskkutya1
b8cd92afb7325ab4fb5ebc669fc10443:Patchieepeach
c6e74525c172fa5c98d4ecdf17af1e62:258258546852sm
cdb0fcd1877db57bdc653f2edce1d4f2:12341234VIDA123
2aedc5cc31efec54eab690b9965df93e:1707199521212301
9918ee06afb86f6e221191dca5ff6159:Ikertorrescabello
37f453b9f2343425bbf24cc22d09d67a:Eventy199419941994
1f8bc7b5572ecb61c901ff2514fe029a:Dragonballgodboi02!
3450d5450844c5c20a1469e256bf1b6d:Alejo2007patrick2007
9eb89ed416bf9f22019b9ff2f516b0f8:Fatimavaleriadanielal
5300fb27045762d4a04d74c147e88751:endutsilandendutsiland
18eea0c4b17e19fb6c631718ba56406a:Argentina2020porsiempre
14d3375fccd50f5dc260641aef7425e8:elenateamopadreteamo0719
4dd66d3634c756a6f9d0c29c0eb1dc8c:Policeisawesome1234567890.
dee2d7efd83d802be253f66e8bee8c17:VALENTINA123456789LISETTEEE
b10d1a02987113cc6cd9729642afee09:yamahafr940228yamahafr940228
bda5ffd3bab2fd1a845514bd797f032f:poopyhead154144poopyhead154144
```

Now let's take the model for a spin…

System Prep & Dependencies

First things first, let's get our system ready to run the LLM model.

1. **Use Windows 10 or 11 for Best Experience**
   GPU drivers cause trouble in Linux

2. **Install the Latest Version of Python**
   https://www.python.org/downloads

3. **Install Python Package Manager (pip)**
   https://www.youtube.com/watch?v=fJKdlf11GcI

4a. For CPU & CUDA GPU Support

Reboot Computer in Safe Mode
https://www.youtube.com/watch?v=Rf3u7PAiWpE

Uninstall Current GPU Driver with DDU
https://www.youtube.com/watch?v=v7KfnZ2wSog

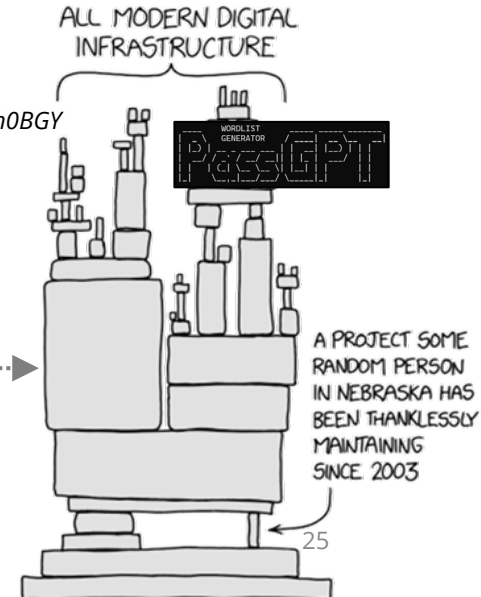
Install CUDA Toolkit & PyTorch GPU
https://www.youtube.com/watch?v=r7Am-ZGMef8

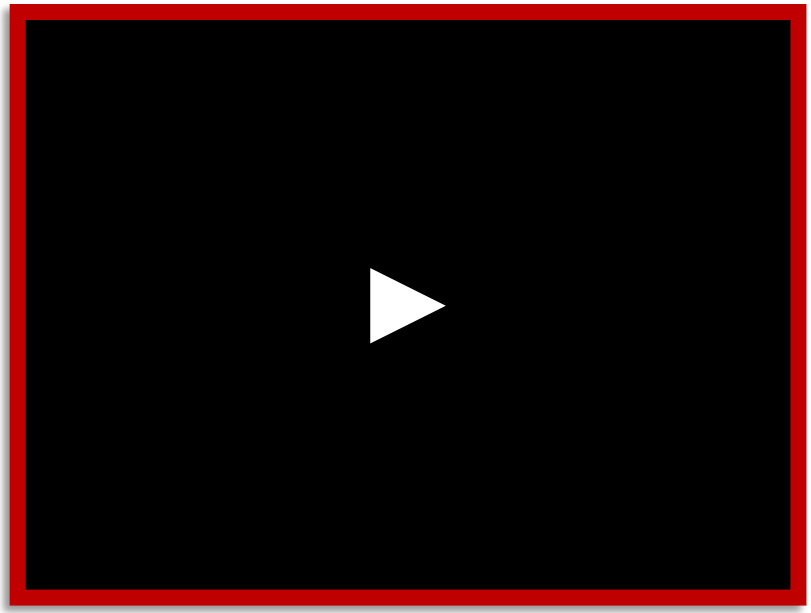
4b. For CPU-Only Support

Install PyTorch CPU
https://www.youtube.com/watch?v=2RkK73h0BGY

5. **[Recommended] Create a Virtual Environment (venv)**
   https://www.youtube.com/watch?v=APOPm01BVrk

6. **Install Dependencies (ideally via venv)**
   python.exe -m pip install transformers huggingface_hub["cli"]





https://youtu.be/NUQsfHW1Nek

☠ Demo Time ☠

When things go terribly wrong…

Final Thoughts & Surprises

Thanks for coming, I will make this presentation, some data, and tools public on my GitHub page, happy hacking!

- ✓ **AWS is Expensive** - Why rent when you can own?
- ✓ **PII in Passwords** - Email addresses, full names, birthdays.
- ✓ **External Thunderbolt NVME** - Makes transporting large files easy.
- ✓ **Leave No Hash Behind** - Data integrity after processing is tricky.
- ✓ **Read Hashcat's Documentation** - There are a lot of awesome tools.
- ✓ **ChatGPT** - Makes writing scripts extremely accessible for all.
- ✓ **Huge Thank You to PassGPT Team** - They provided access to their non-public 16-character model for educational use.

PassGPT Password Modeling and (Guided) Generation with Large Language Models: Javier Rando, Fernando Perez-Cruz, Briland Hitaj

Thank you!

Sean T Smith

www.seantsmith.me

linkedin.com/in/seantsmith

twitter.com/sts5017

github.com/sean-t-smith