_NSAKEY PRESENTS

HASHCAT

GPU PASSWORD CRACKING FOR MAXIMUM WIN

NOW IN TECHNICOLOR

This presentation has been
modified from
its original version.
It has been modified to
fit your screen.

TRIGGER WARNING

THIS TALK IS DESIGNED TO OFFEND THE FOLLOWING GROUPS:

PEOPLE WHO USE WEAK PASSWORDS
USERS OF WEAK HASHING ALGORITHMS
AMD GPU FANBOYS
BITCOIN COLLECTORS
JOHN USERS & DEVELOPERS

YOU HAVE BEEN WARNED.

A Little About Me

- hashcat beta tester
- Tor relay and bridge operator
- ANSI art enthusiast
- "not affiliated with the USA'S NSA"
  - HardenedBSD.org Donor's page
- "I think you're a Kremlin Troll."
  - John "20committee" Schindler

Quick Summary

- Basics of why and how
- Summary and benchmarks of attack types
- Lots of charts
- Analysis
- Q&A
- But first...

Top 10 Commonly Used Passes

#	battlefield	lizardsquad	2011	2012	2013	2014
0	123456	123456	password	password	123456	123456
1	password	lol123	123456	123456	password	password
2	qwerty	123456789	12345678	12345678	12345678	12345
3	123456789	12345	qwerty	abc123	qwerty	12345678
4	starwars	test123	abc123	qwerty	abc123	qwerty
5	killer	password123	monkey	monkey	123456789	123456789
6	12345678	password1	1234567	letmein	111111	1234
7	dragon	123123	letmein	dragon	1234567	baseball
8	battlefield	abc123	trustno1	111111	iloveyou	dragon
9	123123	qwerty	dragon	baseball	adobe123	football

* Lizard Stresser stored all its passes in plaintext.

Why Would Anyone Crack Passes?

- Let's start with good reasons
  - Security research (Whether internal or external)
  - Raising awareness (See Nate Anderson's "How I became a password cracker")
  - Password recovery
    - Hello, law enforcement

Why Would Anyone Crack Passes? (cont.)

- Now for the bad reasons:
  - Account hijacking
  - Accessing protected wifi
  - rming script kiddies
    - Ok, there's nothing actually wrong with this reason.

What Is A Password Hash?

- A hash is a one way function
- In this context, it's used to store passwords
- If the database gets stolen, user passes are not stored in plaintext
- Hashing buys your users time to change their passwords in case of a breach, so pick your algorithms well

Basics of Hashing Algorithms

- Unsalted = OK in 1977, not OK now
- Salting thwarts rainbow tables, which were all the rage in 1998
  - Rainbow tables = Pre-calculate hashes, store the results
- Tunable iteration count = You can keep up with Moore's Law
- tl;dr: bcrypt/scrypt/crypt(3) or quit your day job
- Examples of hashes on the next slide

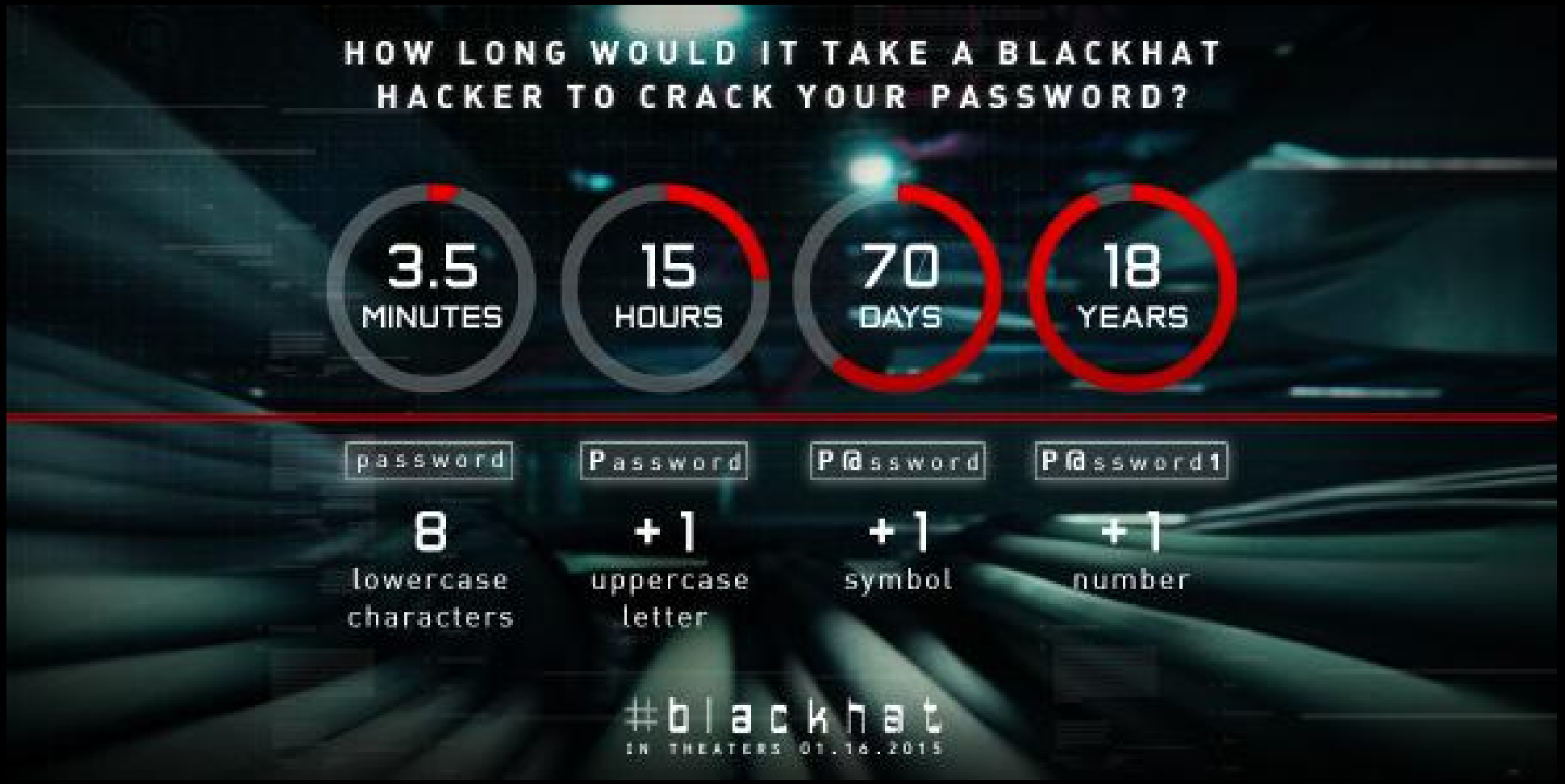
Examples of Hashes

- MD5: 5f4dcc3b5aa765d61d8327deb882cf99
- Salted SHA1: f0f0a169b6e32e99f4c6442754c91ad051757d85:629875d55
- bcrypt: $2a$08$qtju4ihIO264d9pUal5Ct.aicv4oGW6o/.ZT0SXpSHVEMUEjH.qCK
- All of these are hashes for "password"

What Is Hashcat?

- hashcat is an offline password cracker that uses CPUs
- oclHashcat and cudaHashcat are the GPU (video card) versions for AMD and NVIDIA cards
- Supports over 140 different algorithms (With more being added all the time)
- Supports up to 128 GPUs
- This talk will focus on GPU-enabled password cracking, despite the title

Why Use GPUs?



This infographic has no basis in fact

I cracked these in < 1 sec

Hardware Considerations

- unix-ninja does a great job of summarizing the hardware side of things
  - http://www.unix-ninja.com/p/Building_a_Cracking_Rig_for_Hashcat/
- The body of knowledge for hardware could take up a speaking slot in itself, but I'll dispense some GPU wisdom
  - A current example is Evil_Mog's recent DerbyCon talk: https://www.youtube.com/watch?v=1MiY44KS-y4

Hardware Considerations (cont.)

- Only use reference design cards
- If buying new: NVIDIA > AMD
  - NVIDIA closed the performance gap and uses 1/2 the power of AMD's cards
- ... But more beta testers have AMD cards

Hardware Considerations (cont.)

The Bitcoin mining rig conversion strategy I'm about to lay out might get me stabbed, but YOLO

The Dagmar Challenge

- In the late 90s, Dagmar's cracking experiments lead to him cracking 87% of the hashes from his employer in an hour under the following conditions:
- DES hashes
- Sub-500MHz AMD processor
- JtR with CPU-specific optimizations
  - Rough speed was ~10 KH/s
- Dictionary attacks only
  - He had dictionaries for multiple languages

Testing Conventions

- Debian Wheezy with the nvidia.com driver
- NVIDIA GTX 650 Ti BOOST (Benched at 1739 MH/s; real world is between 230-400 MH/s)
- MD5s from Battlefield Heroes Beta (550k users).csv
- Stock rulesets and mask files
  - Unless otherwise noted

Testing Conventions (cont.)

- rockyou.txt (134MB, 14,344,391 words)
  - Also ordered by popularity
- crackstation-human-only.txt (684MB, 63,941,069 words)
- holywow.txt (3.1GB, 311,087,678 words)
- holywow2.txt (859MB, 81,286,807 words)

Attack Types

- Dictionary
- Combinator
- Rules
- Mask

Dictionary

- Works best with passes from previous breaches
- Still, it's REALLY inefficient
- Also useful for validating founds
- rockyou.txt recovers 28.25% in 12 seconds
- crackstation-human-only recovers 35.79% in 19 seconds

Dictionary (cont.)

- Clearly humanity has made progress in choosing passwords since the late 90s
  - Not many people use "love," "sex," "secret," or "god" these days
- ...But password cracking has made even bigger gains in terms of speed
- We can get 87%, it just takes planning and GPU cycles

Combinator

- Tries every combo of words between 2 dictionaries
- Can do 3 dictionaries using hashcat-utils, which is the archnemesis of DiceWare
  - This is a solid DiceWare intro: https://www.youtube.com/watch?v=t7a56mC8E6k
- Can also apply rules to either or both dictionaries
  - Use combinator.rule to get passes-like-this

Why Combinator Didn't Get Tested

- rockyou.txt x 2 would take ~5 days
- rockyou.txt + crackstation-human-only.txt = ~22 days
- crackstation-human-only.txt x 2 = ~87 days

Rule-based attacks

- Mangles words into different words
- Can use multiple rulesets at once
  - They are executed together, not in order
- Compatible with John the Ripper and PasswordsPro
- Examples:
  - ":" tries "password" without modification
  - "c" turns "password" into "Password"
  - "c sa@" turns "password" into "P@ssword"
  - "c sa@ $1" turns "password" into "P@ssword1"

Benchmark Explainer

- The next few slides (And some others later on) have tables of benchmark results
- The dictionary specified in the title of each slide was used in combination with the rulesets in the "Name" column
- Sometimes a smaller ruleset will find more passes than a bigger ruleset with more computationally complex rules
- A real attacker wouldn't crack like this

rockyou.txt + rule sets (1/3)

Name	#Rules	Crack%	Speed	Time
dive	123289	74.16%	239.8 MH/s	1 hour, 54 mins
generated	14729	69.90%	325.9 MH/s	12 mins, 26 secs
rockyou-30000	30000	69.72%	331.9 MH/s	21 mins, 13 secs
d3ad0ne	35404	68.77%	322.4 MH/s	25 mins, 54 secs
generated2	65536	64.16%	260.5 MH/s	54 mins, 52 secs
T0XlC-v1	12000	63.80%	390.6 MH/s	7 mins, 38 secs
InsidePro-Has...	6497	61.42%	409.8 MH/s	4 mins, 37 secs
best64	78	59.07%	171.6 MH/s	10 secs

rockyou.txt + rule sets (2/3)

Name	#Rules	Crack%	Speed	Time
InsidePro-Pas...☻	3141	57.52%	336.0 MH/s	2 mins, 27 secs
T0X1C	4086	54.45%	363.6 MH/s	3 mins, 5 secs
specific	176	42.95%	236.6 MH/s	23 secs
T0X1C-insert_...♥	4016	37.80%	405.5 MH/s	3 mins, 9 secs
combinator	40	33.04%	267.2 MH/s	2 secs
toggles5	4943	30.16%	361.8 MH/s	3 mins, 30 secs
toggles4	1940	30.10%	388.9 MH/s	1 min, 28 secs
toggles3	575	30.04%	341.7 MH/s	34 secs

☻ InsidePro-PasswordsPro

♥ T0X1C_insert_00-99_1950-2050_toprules_0_F

rockyou.txt + rule sets (3/3)

Name	#Rules	Crack%	Speed	Time
toggles2	120	29.97%	243.8 MH/s	17 secs
toggles1	15	29.75%	60625.7 kH/s	13 secs
Incisive-leet...♦	15487	29.71%	76216.4 kH/s	49 mins, 1 sec
Ninja-leetspeak	2047	29.65%	82231 kH/s	5 mins, 58 secs
leetspeak	17	29.34%	260.5 MH/s	13 secs
T0XlC-insert_...♧	1601	29.28%	346.7 MH/s	1 min, 29 secs
T0XlC-insert_...♠	480	28.53%	381.5 MH/s	30 secs
oscommerce.rule	256	6.56%	354.1 MH/s	10 secs

♦ Incisive-leetspeak ♣ T0XlC-insert_top_100_passwords_1_G

♠ T0XlC_insert_space_and_special_0_F

crackstation-human-only + rule sets (1/3)

Name	#Rules	Crack%	Speed	Time
dive	123289	79.42%	125.5 MH/s	9 hours, 5 mins
rockyou-30000	30000	75.79%	322.2 MH/s	1 hour 38 mins
generated2	65536	75.40%	309.7 MH/s	4 hours, 14 mins
d3ad0ne	35404	74.73%	280.8 MH/s	1 hour, 55 mins
T0XlC-v1	12000	70.81%	377.6 MH/s	32 mins, 47 secs
generated	14729	70.19%	278.1 MH/s	53 mins, 59 secs
InsidePro-Has...≡	6497	68.85%	351.2 MH/s	18 mins, 50 secs
InsidePro-Pas...♫	3141	65.96%	336.0 MH/s	9 mins, 58 secs

≡ InsidePro-HashManager

♫ InsidePro-PasswordsPro

crackstation-human-only + rule sets (2/3)

Name	#Rules	Crack%	Speed	Time
T0XlC	4086	62.24%	342.5 MH/s	13 mins, 2 secs
best64	78	52.74%	138.7 MH/s	42 secs
specific	176	50.38%	256.0 MH/s	50 secs
T0XlC-insert_...*	4016	45.44%	337.3 MH/s	11 mins, 54 secs
combinator	40	40.51%	270.1 MH/s	10 secs
Incisive-leekspeak	15487	37.48%	278.1 MH/s	4 hours, 23 mins
Ninja-leetspeak	2047	37.42%	77591.7 kH/s	31 mins, 10 secs
toggles5	4943	37.41%	372.0 MH/s	14 mins, 10 secs

☼ T0XlC-insert_00-99_1950-2050_toprules_0_F

☺ toggles5 cracks 11 more hashes than toggles4

crackstation-human-only + rule sets (3/3)

Name	#Rules	Crack%	Speed	Time
toggles4	1940	37.41%	380.6 MH/s	5 mins, 31 secs
toggles3	575	37.39%	357.8 MH/s	1 min, 48 secs
toggles2	120	37.34%	224.8 MH/s	36 secs
toggles1	15	37.14%	50393.4 kH/s	21 secs
leetspeak	17	37.05%	49795.6 kH/s	24 secs
T0X1C-insert_...☻	480	37.12%	301.7 MH/s	5 mins, 35 secs
T0X1C-insert_...♥	1601	36.08%	361.1 MH/s	1 mins, 27 secs
oscommerce	256	8.68%	427.7 MH/s	40 secs

☻ T0X1C-insert_top_100_passwords_1_G

♥ T0X1C-insert_space_and_special_0_F

Ruleset Post-Mortem

- 'sort -u | wc -l' all the things!
- rockyou.txt + rules = 76.86%
- crackstation-human-only.txt + rules = 81.74%
- Combined founds from both = 81.82%
  - To find more, we must get creative...

Mask Attack

- Targeted brute forcing
- We know how humans create passes due to breaches
- ?u?s?l?l?l?l?l?l?d will crack "P@ssword1"
- Can create up to 4 custom character sets

Mask Attack Syntax

- ?u = uppercase (A to Z)
- ?l = lowercase (a to z)
- ?d digits (0 to 9)
- ?s = symbols (`~!@#$%^&*()-_=+ {[}]|\:;-'<,>.?/)
- ?a = all of the above
- ?b = binary (0x00 to 0xff)

Mask Results

Name	#Masks	Crack%	Speed	Time
rockyou-1-60	836	52.84%	800.3 MH/s	20 mins, 25 secs
rockyou-2-1800	2968	69.59%	672.8 MH/s	5 hours, 40 mins
rockyou-3-3600	3971	73.90%	817.9 MH/s	14 hours, 38 mins

All totals are cumulative

73.90% = easy passes found

This + other founds = 87.73%

Challenge = met, we can move on. But first...

Intermission

Ok, now what?

- Both holywows were fed to PACK
  - https://thesprawl.org/projects/pack/
- rulesgen.py made a 50GB and 15GB file
- Sorting left me with 48GB and 12GB of rules ordered by popularity
  - `sort | uniq -c | sort -rnk1 | head -nXXXX | awk '{print $2}' > holywow-[$RULESET].rule`
  - This still only brought the total found to 88.31%
- Ready for more ANSI spreadsheets?
  - Of course you are

rockyou vs holywow

Name	#Rules	Crack%	v1 %	v2 %
dive	123289	74.16%	74.12%	74.43%
generated	14729	69.90%	66.40%	67.30%
rockyou-30000	30000	69.72%	69.00%	69.75%
d3ad0ne	35404	68.77%	69.61%	70.30%
generated2	65536	64.16%	69.90%	72.29%
T0XlCv1	12000	63.80%	66.04%	66.60%
InsidePro-HashManager	6497	61.42%	63.75%	64.37%
best64	78	59.07%	43.35%	43.96%
InsidePro-PasswordsPro	3141	57.52%	61.16%	61.57%

crackstation-human-only vs holywow

Name	#Rules	Crack%	v1 %	v2 %
dive	123289	79.42%	79.46%	79.71%
rockyou-30000	30000	75.79%	75.16%	75.67%
generated2	65536	75.40%	77.56%	77.64%
d3ad0ne	35404	74.73%	75.65%	76.15%
T0XlCv1	12000	70.81%	72.48%	73.03%
generated	14729	70.19%	73.00%	73.64%
InsidePro-HashManager	6497	68.85%	70.78%	71.12%
InsidePro-PasswordsPro	3141	68.96%	68.56%	68.68%
T0XlC	4086	62.24%	69.33%	69.57%

Ph'nglui mnlw'nafh Cthulhu R'lyeh wgah'nagl fhtagn

- Any mask with over 100k matches was included
- Both holywows contained large (20+ char) masks
- 21 char lowercase would take ~21 trillion years
  - That's over 518 undecillion combos
- 33.317 ZH/s to try every combo in 6 months

Faked Mask Benchmarks

Name	#Masks	Crack%	Speed	Time
rockyou-1-60	836	52.83%	800.3 MH/s	20 mins, 25 secs
rockyou-2-1800	2968	69.59%	672.8 MH/s	6 hours, 40 mins
rockyou-3-3600	3971	73.90%	817.9 MH/s	14 hours, 38 mins
holywowv1	354	96.08%	n/a	n/a
holywowv2	88	90.15%	n/a	n/a

How I cheated:

```
awk -F, '{print $1}' holywow-100k.masks >
holywow.1

grep -f holywow.1 battlefield.masks | awk -F,
'{ sum+=$2} END {print sum}'
```

What Would A Real Attacker Do Next?

- Put founds in a dictionary, run attacks
- Dictionaries for alternate languages
- Keyboard walks (e.g. asdf, qwerty)
- Make a dictionary using WordHound
- Tmesis (Inserts words into other words)

  123456asdf

  a123456sdf

  as123456df

  asd123456f

  asdf123456

What's Left?

- PACK analysis
- Credits
- Q/A?

Length (PACK)

8: 29% (122967)	12: 03% (15483)	18: 00% (195)	2: 00% (4)
9: 16% (66769)	13: 01% (7063)	19: 00% (96)	1: 00% (2)
6: 15% (65844)	14: 00% (4000)	20: 00% (75)	21: 00% (2)
7: 13% (57553)	15: 00% (2094)	5: 00% (9)	22: 00% (2)
10: 11% (48658)	16: 00% (1090)	3: 00% (15)	23: 00% (2)
11: 05% (24425)	17: 00% (332)	4: 00% (8)	24: 00% (1)
			26: 00% (1)

Character Sets Used (PACK)

loweralphanum:	53% (222821)
loweralpha:	24% (103004)
numeric:	08% (35684)
mixedalphanum:	07% (32957)
mixedalpha:	01% (7484)
loweralphaspecialnum:	01% (4576)
upperalphanum:	00% (3384)
all:	00% (2207)
loweralphaspecial:	00% (1935)
upperalpha:	00% (1437)
mixedalphaspecial:	00% (580)
specialnum:	00% (375)
upperalphaspecialnum:	00% (174)
upperalphaspecial:	00% (51)
special:	00% (21)

Password Complexity (PACK)

digit: min(0) max(24)
0160910849270160910849 27

lower: min(0) max(23)
overmyheadbetteroffdead

upper: min(0) max(26)
QWERTYUIOPASDFGHJKLZXCVBNM

special: min(0) max(12) **********
and .,.,.,.,.,

Simple Masks (PACK)

stringdigit: 44% (184594)
string: 26% (111925)
digit: 08% (35684)
othermask: 07% (29591)
stringdigitstring: 06% (26103)
digitstring: 03% (16377)
digitstringdigit: 01% (6258)
stringspecialdigit: 00% (1975)
stringspecialstring: 00% (1451)
stringdigitspecial: 00% (1011)
stringspecial: 00% (590)

Advanced Masks (PACK)

?l?l?l?l?l?l?l?l?l: 05% (24525)
?l?l?l?l?l?l?l?l: 05% (23361)
?l?l?l?l?l?l?d?d: 04% (20352)
?l?l?l?l?l?l?l?l: 04% (18370)
?d?d?d?d?d?d?d: 03% (13988)
?l?l?l?l?l?l?l?l?l: 03% (13944)
?d?d?d?d?d?d?d?d: 02% (10729)
?l?l?l?l?l?l?l?l?l?l: 02% (10346)
?l?l?l?l?l?l?l?d?d: 02% (9202)
?l?l?l?l?l?l?l?d: 02% (8846)
?l?l?l?l?l?d?d?d: 01% (8008)
?l?l?l?l?d?d?d?d: 01% (7927)
?l?l?l?l?l?d?d: 01% (7696)
?l?l?l?l?l?d?d?d: 01% (7054)
?l?l?l?l?l?l?l?d?d: 01% (6766)
?l?l?l?l?l?l?l?l?d: 01% (6491)
?l?l?l?l?d?d: 01% (5793)
?l?l?l?l?l?d?d?d: 01% (5575)
?l?l?l?l?l?d?d?d: 01% (5563)
?l?l?l?l?l?l?d: 01% (5502)
?l?l?l?l?l?l?l?l?l?l: 01% (5470)
?d?d?d?d?d?d?d: 01% (5408)
?l?l?l?l?l?d: 01% (5067)

Protecting Yourself And Your Users

- Developers: Use password hashing algorithms like bcrypt and scrypt. Almost all of your other choices are riddled with peril
- Users: Randomly generate all passwords, store in either a password manager (If you're a mere mortal) or an encrypted text file (If you're a wizard)
  - Use good passphrases (Maybe look at Diceware) in situations when you can't avoid using a memorable pass
- Misc: LUKS containers use 4096 rounds of SHA1 by default. I changed the LUKS header on my containers to use 9001 rounds of SHA256

What I Covered Today

- Basics of hashing algorithms
- Beginner level hashcat info
- Lots of benchmarking
- Tasteful chiptune intermezzo
- Even more benchmarking
- Analysis of the test hash dump
- Very light amounts of Blue Teaming

References

- https://hashcat.net
  - https://hashcat.net/events/
  - https://hashcat.net/wiki/
- http://www.snipview.com/q/Names%20of%20large%20numbers
- http://thesprawl.org/projects/pack/
- https://bitbucket.org/mattinfosec/wordhound
- http://splashdata.com/press/PR111121.htm
- http://splashdata.com/press/pr121023.htm
- http://splashdata.com/press/worstpasswords2013.htm
- http://splashdata.com/press/worst-passwords-of-2014.htm
- https://abigisp.com/talks/hashcat/

Credits

Arnie Holder - chiptune enabler

atom - hashcat developer

benthemeek - Guinea Pig

Dagmar - Expectation & Scope Setter

Elonka - Slides Coach

epixoip - Hashing Historian

iphelix - PACK Author

NoFault - Number Cruncher

RangerZ - Large Numbers Consultant

Trash80 - Created "Girl From Intermission"

Contact

Send any questions, comments, death threats, and/or world domination conspiracies to:

E-mail: root@abigisp.com

PGP: https://abigisp.com/key.txt

Key ID: 0xF3C1BD78

Fingerprint: D748 92B2 FBC7 4B86 65B4 0210 DA0B 584C F3C1 BD78

Twitter: @_NSAKEY

GitHub: NSAKEY

The End?