Unleash The Kraken

The Art of Password Cracking

whoami

Hacker @ Northwestern Mutual

Previously: 403 Labs/Sikich, FIS/Metavante

What is password cracking?

The discovery of a plaintext value that produces a hash equal to the target hash.

What's a hash?

A hash is a one-way cryptographic algorithm.

Common algorithms:

- MD5
- SHA1
- NTLM

Example:

```
md5('password') = 5f4dcc3b5aa765d61d8327deb882cf99
```

Hashing != Encryption

Why Crack Hashes?

Attackers

- Lateral movement
- Privilege escalation
- Reuse (password and hash)

Defenders

- Adherence to password policy
- Baseline awareness training
- Monitor password reuse (internal & external)

Where are hashes stored?

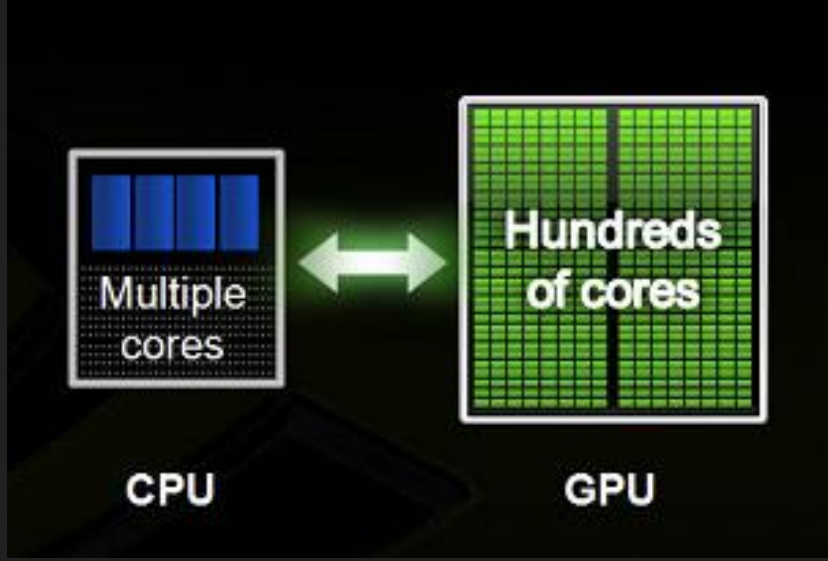
- Local Windows SAM file
  - LM/NTLM
- Active Directory
  - LM/NTLM
- Websites
  - Databases
  - 3rd party identiy providers

How to Obtain Hashes

- NTLM
  - Active Directory ntdsutil (https://technet.microsoft.com/en-us/library/cc753343(v=ws.11).aspx)
  - Lan Turtle (https://room362.com/post/2016/snagging-creds-from-locked-machines/)
  - Mimikatz
  - PowerShell post-exploitation tools
  - pwdump/fgdump
- Websites
  - SQL Injection
  - Admin interfaces
  - Direct DB access

Types of Cracking

- CPU - Standard CPUs
  - 24 threads ~5 Billion guesses/sec
- GPU - Graphics Cards
  - 6 Nvidia GPUs ~75 Billion guesses/sec
- FPGA/ASIC
  - Fast, but $$$$
- Rainbow Tables
  - ? Too slow



http://resources.infosecinstitute.com/password-cracking-evolution/


GEFORCE GTX
GEFORCE GTX
GEFORCE GTX
GEFORCE GTX
GEFORCE GTX
GEFORCE GTX

367k MD5 Hashes

Goto page 1, 2 Next

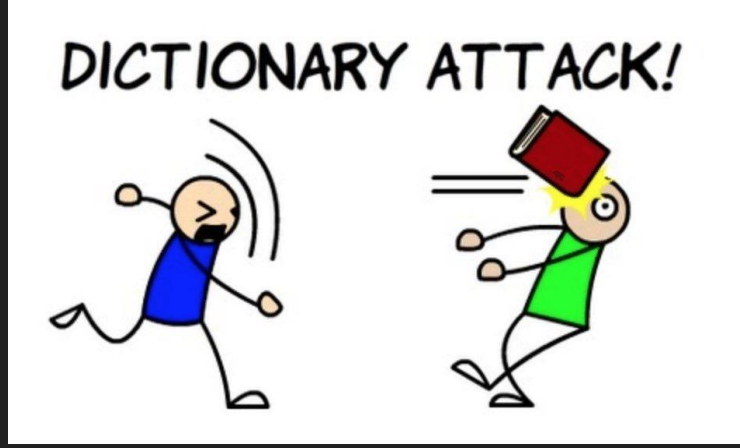
New Topic Post Reply InsidePro Software Forum Index -> Standard Hashes

Author	Mess
 vimeo1903 Joined: 10 Sep 2011 Posts: 356 Gold Member Reputation: 223	Posted: Mon May 29, 2017 2:40 pm Post subject: 367k MD5 Hashes Please help with 367k MD5 Hashes https://www.sendspace.com/file/5ymk42

The Art

Dictionary (Wordlist) Attacks

A text file with password candidates.



https://twitter.com/andyf/status/896032178622083077

- Common Dictionaries
  - John the Ripper, 500 worst, etc
- Literal dictionaries
  - Various languages
- Leaked
  - Rock You
  - Hashes.org (60 leaks including LinkedIn, eHarmony, Yahoo!, etc.)
  - Troy Hunt's Pwned Passwords (https://haveibeenpwned.com/Passwords)
- Compiled
  - Crackstation (https://crackstation.net/)
  - Purehate's Wordlist
- Custom
  - Cewl

cewl

wordlist

Rules

Transform or modify a password candidate

Input: password

Rule	**Transform**
T0	Password
$1	password1

rules

wordlist + rules

Good Rule Sets

- Default hashcat
  - T0XlCv1.rule
  - dive.rule
  - d3ad0ne.rule
- Hob0Rules (https://github.com/praetorian-inc/Hob0Rules)
- Make your own!

Brute Force

Exhausts all possible password candidates.

```
aaaaaaaa
aaaaaaab
aaaaaaac
aaaaaaad
aaaaaaae
aaaaaaaf
aaaaaaag
aaaaaaah
........
```

Keyspace

Calculated by the length to the potential character power.

8 characters upper, lower, digits & specials =

$8^{(26 + 26 + 10 + 34)}$ =

49732323640978664215538224814682084010045615079734771744046397689315949701253337553306

~23h30m of cracking time

brute force

Masks

A per-position targeted brute force attack

- ?l = abcdefghijklmnopqrstuvwxyz
- ?u = ABCDEFGHIJKLMNOPQRSTUVWXYZ
- ?d = 1234567890
- ?s = «space»!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
- ?a = ?l?u?d?s
- ?1 = custom char set 1
- ?2 = custom char set 2
- ?3 = custom char set 3
- ?4 = custom char set 4

Masks Reduce Keyspace

Humans use predictable password patterns

Packers1 = ?u?l?l?l?l?l?l?d (26*26*26*26*26*26*26*10 = 80.3 Billion)

- Regular brute force of 8 chars ~1 day
- This mask takes ~1 second
- Mask is also valid for Brewers3

Password Analysis and Cracking Kit (PACK)

- https://thesprawl.org/projects/pack/
- StatsGen - Statistically analyze a word list
- MaskGen - Generates masks based on common password patterns
- PolicyGen - Generate masks based on a password policy
- RuleGen - Analyze a password to detect rules and source word

masks

Reduce Keyspace

Example:

```
?l?s?d?d?d?d           = 8580000 (8.6M)
!@.$#*_- ,?l?1?d?d?d?d = 2340000 (2.3M) 73%
```

```
#!/usr/bin/env python
import sys

with open(sys.argv[1], 'rb') as fin:
    for line in fin:
        mask = line.strip()
        for c in mask:
            if c == 's':
                mask = '!@.$#*_- ,%s' % mask.replace('s', '1')
                break

        print(mask)
```

Markov Chains

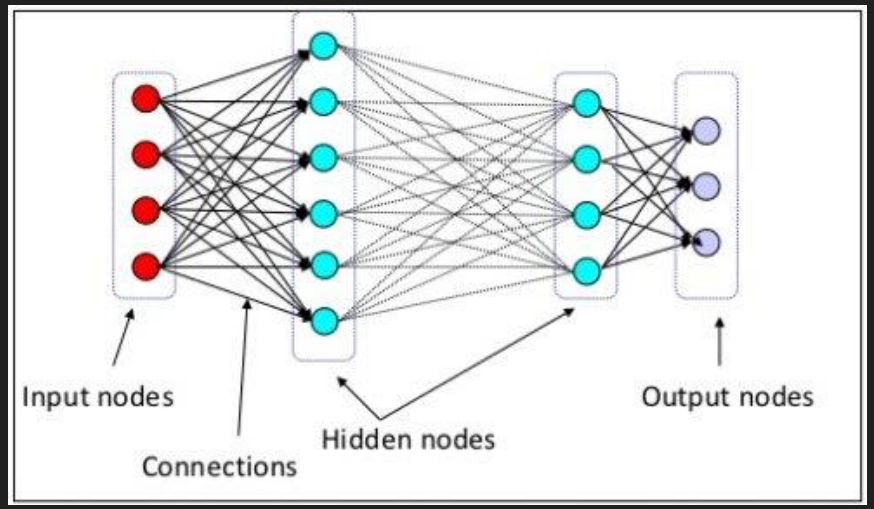
A probability mass function to determine the next state

```
> table(chain[['walk ']]) / length(chain[['walk ']])
 a   b   i   l   m   o   u
0.4 0.1 0.1 0.1 0.1 0.1 0.1
```

https://cartesianfaith.com/2016/02/23/is-deep-learning-a-markov-chain-in-disguise/

What is Machine Learning?

- Mislabeled as “Artificial Intelligence”
- Really an advanced statistics processor
- Can “predict” future outcomes based on the past



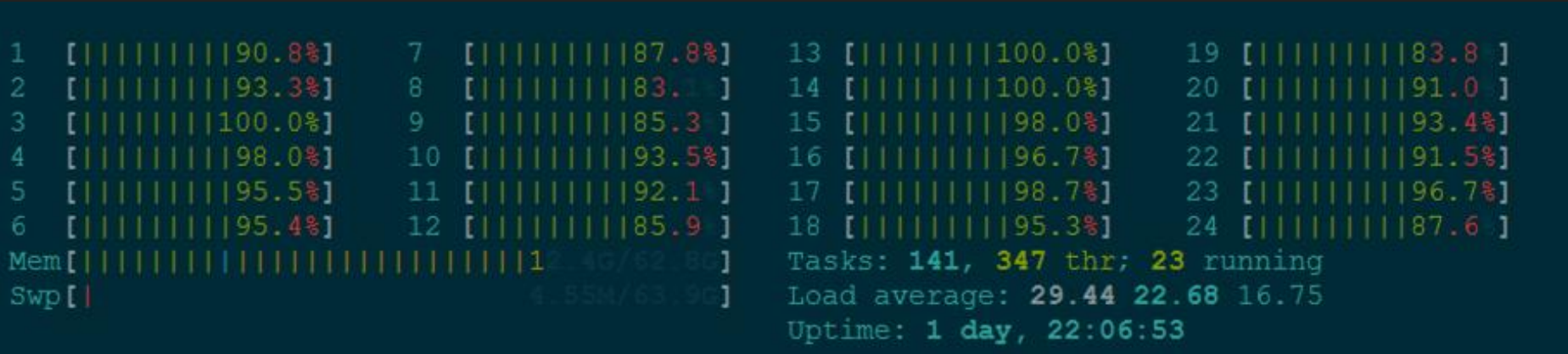
http://www.kdnuggets.com/2017/08/deep-learning-neural-networks-primer-basic-concepts-beginners.html

ML Applied to Password Cracking

1. Input already cracked passwords as a training set
2. Train the neural network
3. Generate password candidates to test effectiveness

ML Attempt 1 (Fail)

[root@kraken]# python learn_passwords.py cheddarcon
Using TensorFlow backend.
..snip..
[*] Total Characters: 1705765
[*] Total Vocab: 125
[*] Total Patterns: 1705665
[*] training model...
Epoch 1/50
  43500/1705665 [.....] - ETA: 17711s - loss: 3.5119



ML Attempt 2

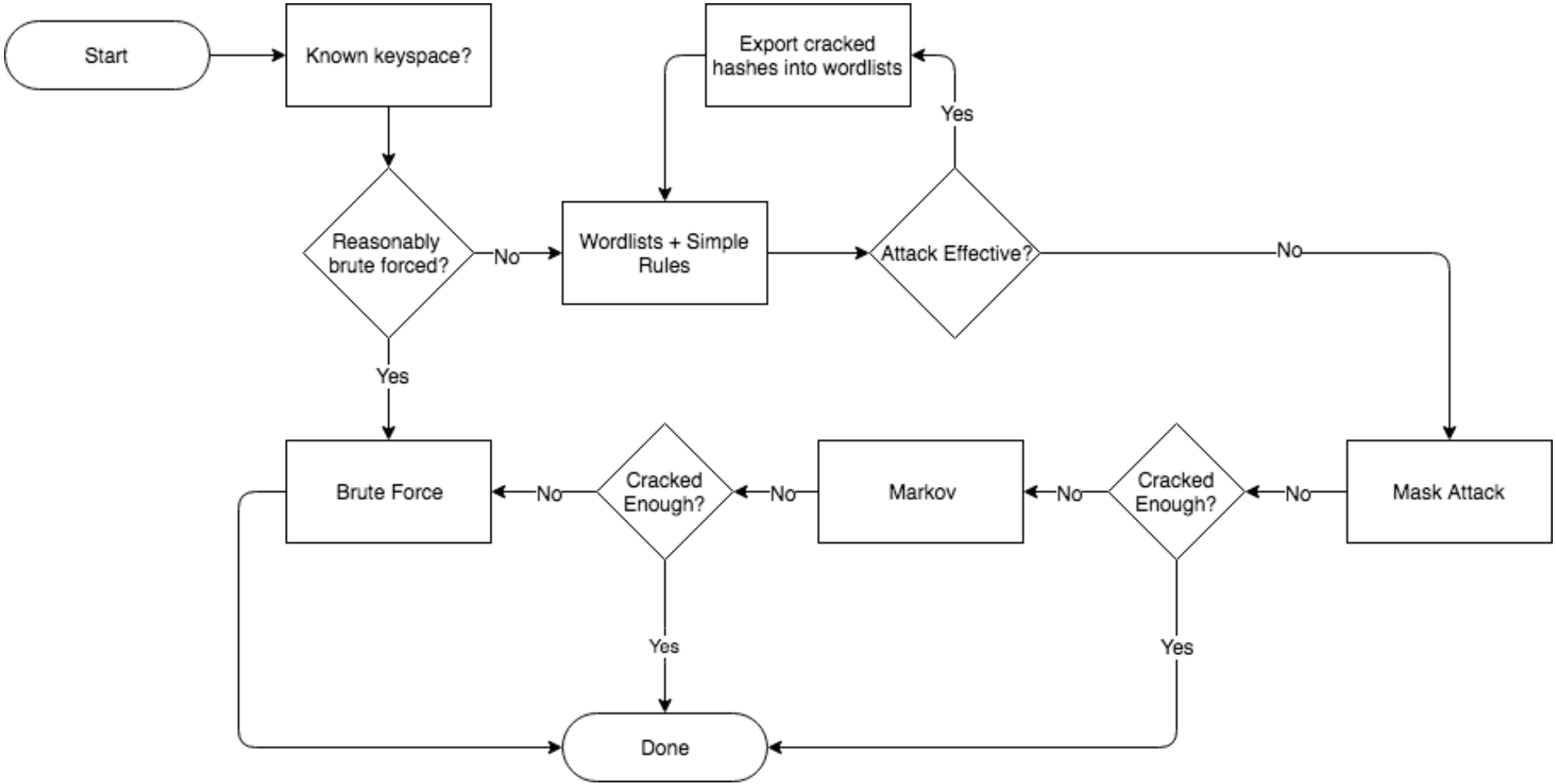
https://0day.work/using-neural-networks-for-password-cracking/

```
for i in $(ls cv/*.t7); do th sample.lua -length 100000 $i >
output/${i/t7/txt}; done
```

Success! 22 new passwords were discovered using this method.

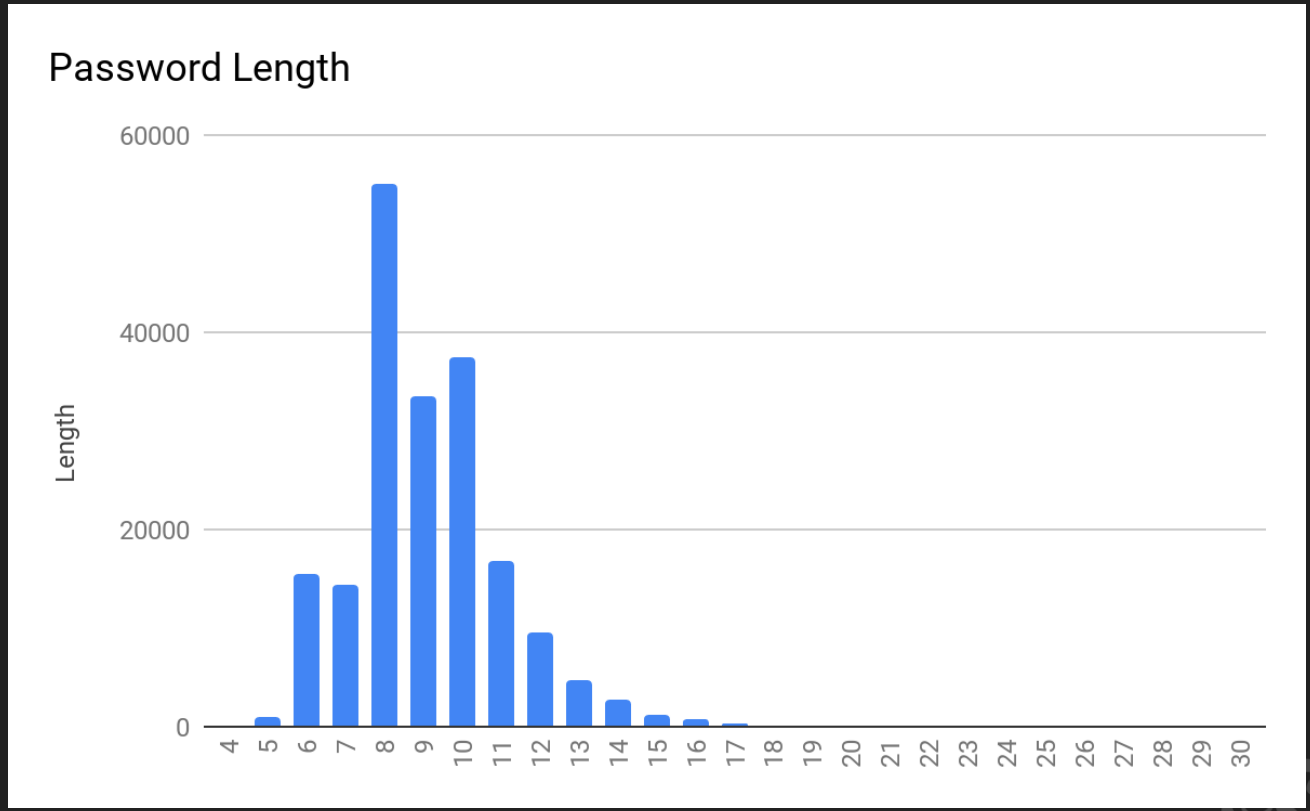
```
7a264f32ef16ff818199b9fece0355c0:themexp10120611
a0a328c6c8516ed9bea587e1c11e6cf3:guyu19855141
d36ed65fcb680bbb47a7b5a3d0350ec9:niirproject72
ff923e8cd9d0495f610d3198362e8c34:niirproject3
```

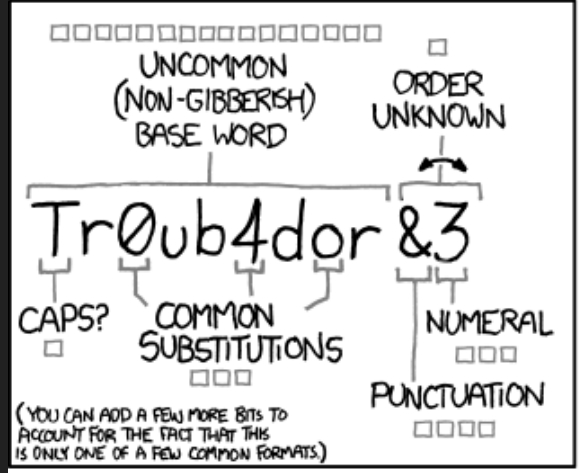
Methodology

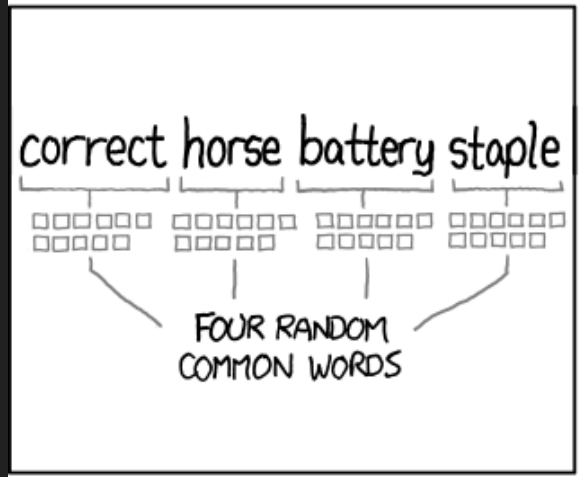

Start
Known keyspace?
Export cracked hashes into wordlists
Yes
Reasonably brute forced?
No
Wordlists + Simple Rules
Attack Effective?
No
Yes
Brute Force
No
Cracked Enough?
No
Markov
No
Cracked Enough?
No
Mask Attack
Yes
Yes
Done

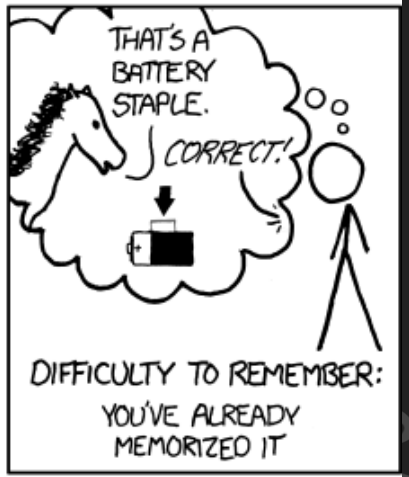
So how did our attacks fare?

Attack	% Cracked	Cumulative %	Elapsed Time
wordlists	20.44%	20.44%	2m13s
ztg.rule * 2	+35.29%	55.73%	9m44s
D3adhob0.rule	+6.79%	62.52%	26m15s
T0XlCv1.rule + ztg.rule	+15.3%	77.83%	20.6h
Brute all 1-7 chars	+0.25%	78.08%	16m59s
PACK mask attack >= 8	+6.85%	84.93%	11 hours (stopped)
Machine Learning	+0.0%	84.93%	<1m
Markov 8chr	+0.05%	84.98%	1 hour
rules/d3ad0ne.rule	+0.48%	85.46%	33m2s
Misc rules + Markov masks	+0.36%	85.82%	2h


Password Length
60000
40000
20000
0
Length
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30







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.

“Best Practices”

- Use a strong password hashing algorithm (PBKDF2, bcrypt)
- Salt passwords when possible
- Encourage long passphrases (longer == stronger)
- Allow all printable characters (😬)
- Enable two-factor authentication
- NIST SP 800-63-3 Digital Identity Guidelines - https://pages.nist.gov/800-63-3

Links - Tools

- https://hashcat.net/hashcat/
- https://hashcat.net/wiki/doku.php?id=statsprocessor
- https://thesprawl.org/projects/pack/
- https://digi.ninja/projects/pipal.php

Links - Wordlists

- https://crackstation.net/ (15G)
- https://haveibeenpwned.com/Passwords (12G)
- https://hashes.org/public.php
- https://wiki.skullsecurity.org/index.php?title=Passwords
- https://blog.g0tmi1k.com/2011/06/dictionaries-wordlists/
- https://dumps.wikimedia.org/
- https://digi.ninja/projects/cewl.php
- https://www.gutenberg.org
- https://isubtitles.in/

Links - Rules

Good non-default rules:

- https://github.com/praetorian-inc/Hob0Rules
- https://github.com/nyxgeek/nyxgeek-rules/tree/master/hashcat-rules
- http://contest-2010.korelogic.com/rules-hashcat.html

Links - Misc

Kraken logo: http://krakenrobotik.de/

Machine Learning

- https://0day.work/using-neural-networks-for-password-cracking/
- https://github.com/k3170makan/PyMLProjects/tree/master/passwords
- https://www.ece.cmu.edu/~lbauer/papers/2016/usenixsec2016-neural-passwords.pdf