Force multiplier - Guided password cracking

Tonimir Kišasondi, PhD
Laboratory for Open Systems and Security,
Faculty of Organization and Informatics,
Varaždin

Why should we care?

Passwords are the most widespread authentication method

Security is pretty much a depressing affair, and attackers can (and do) try password reuse and pivoting.

Update: LinkedIn Confirms Account Passwords Hacked

By Ian Paul, PCWorld

Jun 6, 2012 8:32 AM


news.cnet.com/8301-1009_3-57609753-83/adobe-hack-attack-affected-38-million-accounts/
c|net
English
Reviews
News
Download
CNET TV
How To
Deals
CNET › News › Security & Privacy › Adobe hack attack affected 38 million accounts
Adobe hack attack affected 38 million accounts
Next Story


CNNMoney
FORTUNE
Home
Video
Markets
Investing
Economy
Tech
All Latest Stories | Companies | World | The Buzz | Fortune 500 | Face to Face | Video
Target: Hacking hit up to 110 million customers
CNNMoney
By Chris Isidore @CNNMoney January 11, 2014: 6:20 PM ET
The attack actually involved 38 million active accounts.

Should you care?

Well, if you don’t test your systems…

Don’t worry, the kind people that live in the Internets will :)

The “psychology” of passwords

Ideal passwords:

e;]iC"8aG;vQ}#Qg=:(R15A

Real passwords:

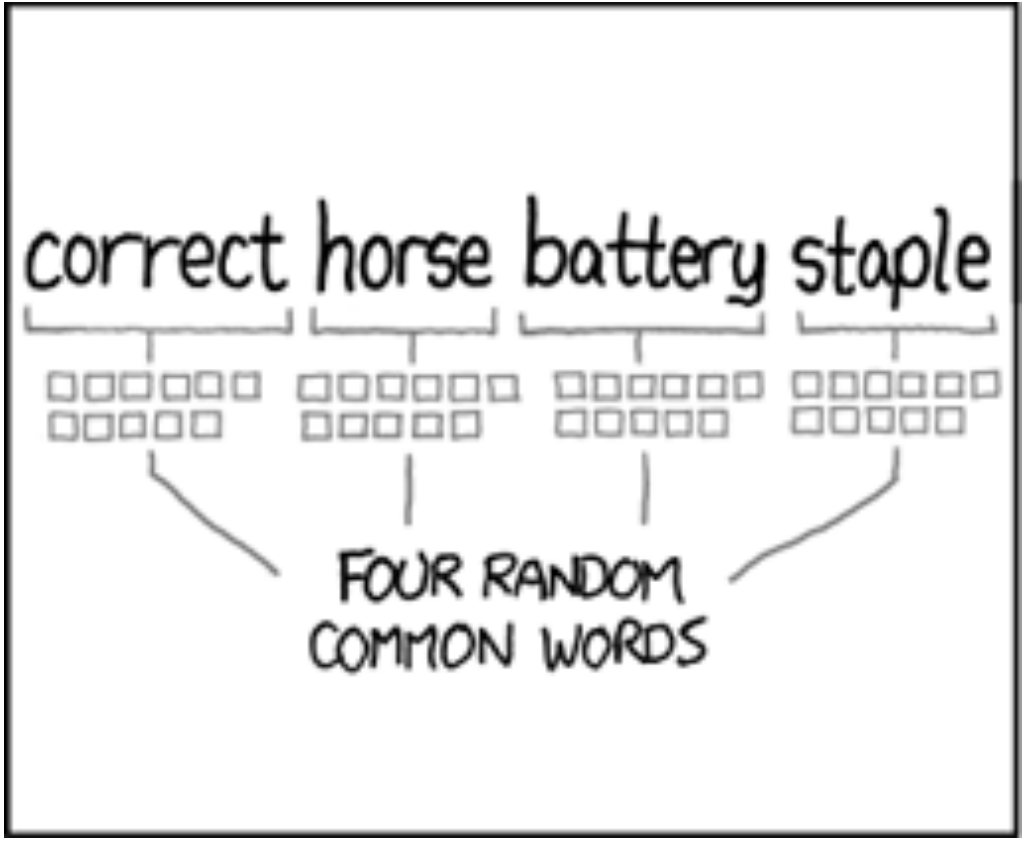
Password1! loveyou!

tkisason1!

Default passwords:

root support admin manager

Mandatory XKCD reference :)



The efficiency of default passwords

Forgotten devices

Testing systems / testing devices

Sloppy configs

Biggest problem: hardcoded backdoors

Srsly...

Hardcoded backdoors

Kronos Access control

SuperUser:2323098716

Morpho Itemiser 3 (Explosives & Narcotics)

Administrator 2:838635

SuperUser 2:695372

HP Storage

HPSupport:badg3r5

The unhash project

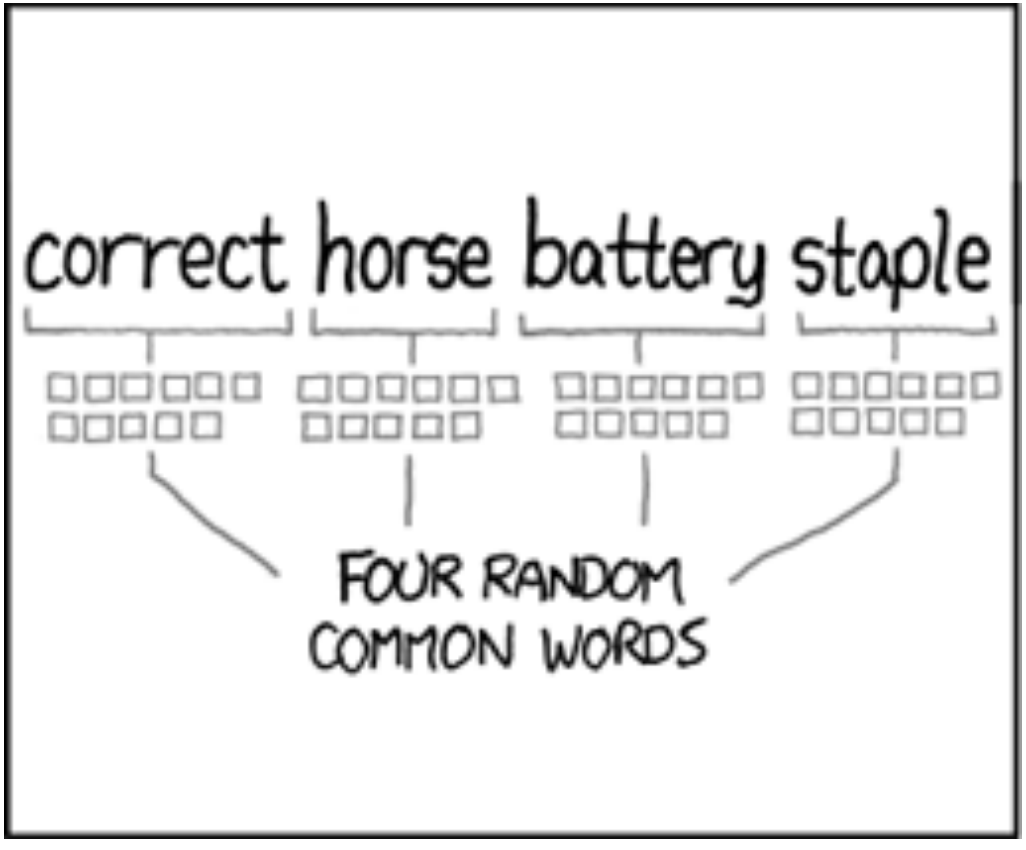
Developed as a part of my research into passwords (and the result of my thesis)

https://github.com/tkisason/unhash

Contribute elements into other tools (like Metasploit)

unhash github.com/tkisason/unhash

1. default_passwords
2. botpass
3. gwordlist
4. unhash


correct horse battery staple
FOUR RANDOM
COMMON WORDS

[{ 'f':word },’ ’,{ 'f':word },’ ’,{ 'f':word },’ ’,{ 'f':word }]

Collecting default user/pass pairs

My approach: Scraping posted default passwords databases on the net:

- Phenoelit, liquidmatrix, securityoverride, dexcms, cirtnet

Organized and weighted by occurrence

Useful to test out default passwords

Collecting default user/pass pairs

Availability:

default_passwords in unhash

For automatic updates

default_userpass_for_services_unhash.txt in Metasploit Framework (If you trust my maintainance)

Collecting datasets from attackers

sshpot.com collects data from ssh honeypots in the wild (if you run a ssh honeypot, ship data to them)

- Easier collecting of lists used in the wild and toolmarking attackers?

botpass in unhash to test against common attackers

Password classes

The previous tools are for online testing, what about offline password testing and attacks?

By checking out about 14 million unique passwords from all larger leaks, a few patterns emerged

Password classes

Weak patterns

Keyboard patterns

Inserts

Symmetric elements

Mutation (anything that changes a word)

Combination

Terminal classification (word, symbol, num, ws)

Sieving

Classification with the sieve algorithm

Mine for data elements while training

How to identify words, languages?

- Wikipedia database dumps (20gb in psql)
- pg_trgm and GIN

Sieving

Use the linguistic base to identify trends

Sieving about 33M (14M uniq) passwords yields models “how users create their passwords”

```
> sieve('lo(ikju&^%$')
[{'el': 'lo(ikju&^%$', 'len': 11, 'type':
'keyboard_pattern'}]

> sieve('foifoifoifoi')
[{'el': 'foifoifoifoi', 'len': 3, 'n': 4, 'type':
'weak_pattern'}]
```

```
> sieve('1984c0mpl1cated!(&%')
[{'el': '1984', 'len': 4, 'type': 'sequence'},
 {'REPLACE': {'i': '1', 'o': '0'}, 'type': 'mutation', 'x':
'complicated'},
 {'el': 'complicated',
  'langs': ['en', 'de', 'es', 'fr', 'hr', 'it', 'nl'],
  'len': 11, 'occ': 43561, 'type': 'word'},
 {'el': '!(&%', 'len': 4, 'type': 'special'}]
```

[{‘f’:numlen4},
{‘f’:dictlen11,’r’:{'i': '1', 'o': '0'}},
{‘f’:strlen4}]

unhash

The benefit of raw models is that they don’t discriminate wordlists :)

1) Merge mined sets with the spread from wikipedia (mixing generality with learned words) - default unhash wordlists

2) Obtain wordlists that are specific to the target.

gwordlist

Obtain wordlists that are specific to the target?

gwordlist can be used to scrape top N google results based on your keywords or google dorks and create wordlists

(yes, you can recurse!)

unhash

Default rulefiles and wordlists are already on github.

Comes with batteries (bullets) included.

use with:

pypy unhash.py rulefile | john --stdin ....

Results on linkedin list in 24hours

Keywords: linkedin, business, recruiting, networking, job, contacts

Type	Total	%
john-1.7.9-jumbo7	1.225.503	100,00%
unhash | john-1.7.9-jumbo7	1.509.001	123,13%
Diff	283.498	23,13%

Conclusion

It’s on github, have fun, don’t be evil

Passwords suck (do’h)

Passwords are the best thing we currently have.

    Yes, 2FA is cool too.

Use salted passphrases

Contributors / researchers are welcome

Thank you!

tonimir.kisasondi@foi.hr

@kisasondi