

# Password Cracking

Elliott Brink  
@ebrinkster

# /usr/bin/whoami

- Elliott Brink
- Twitter (@ebrinkster)
- Senior Penetration Tester
- Available for questions after the talk
- Disclaimer: All words (and images) are my own (or someones on the internet, mostly the videos) and do not reflect official views of my employer

# 2nd Disclaimer

- Last minute speaker cancellation.
- Slides prepped last night, be gentle.

# 10 Second Agenda — Password Cracking

- Turning an encrypted or hashed value into its original value
- Pentesters, security auditors, hackers alike, should know this

# Before this talk

- This (might) be you
- WHAT IS PASSWORD
- How do I crack passwords?!

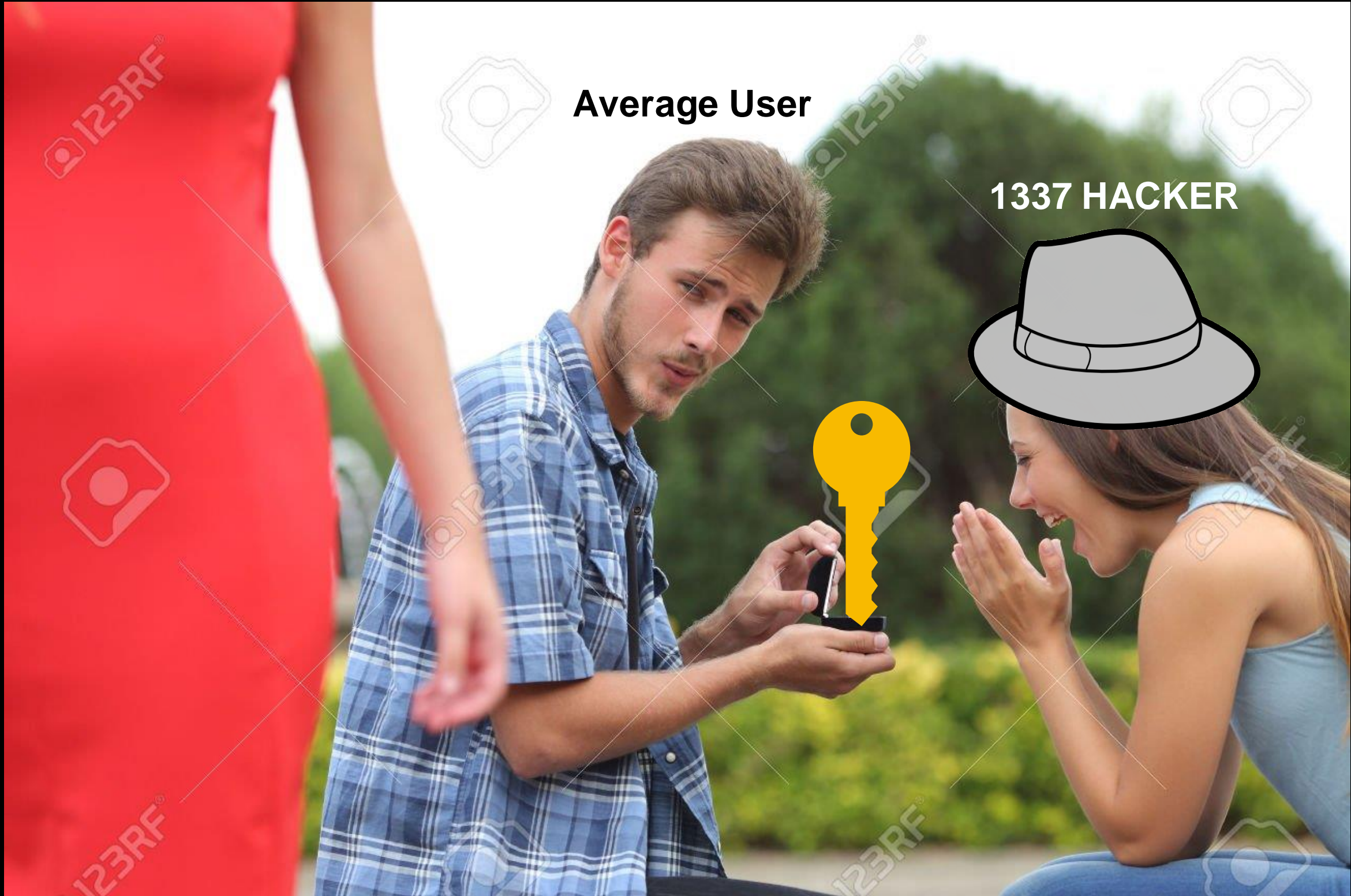




# After this talk

 **Average User**

**1337 HACKER**





0101 NAME ADDRESS  
010010100101011010010011  
N 101 LOGIN **PASSWORD**  
010010100101011010010011  
01010 NAME ADDRESS  
010010100101011010010011  
010101011010101101011010  
010010100101011010010011010  
010010100101011010011010



# WIFI PASSWORD HACKER

PASSWORD





# The End

- Slides will be made available later.
- Blah, blah, blah.
- <Insert Twitter Account Here>

# The End

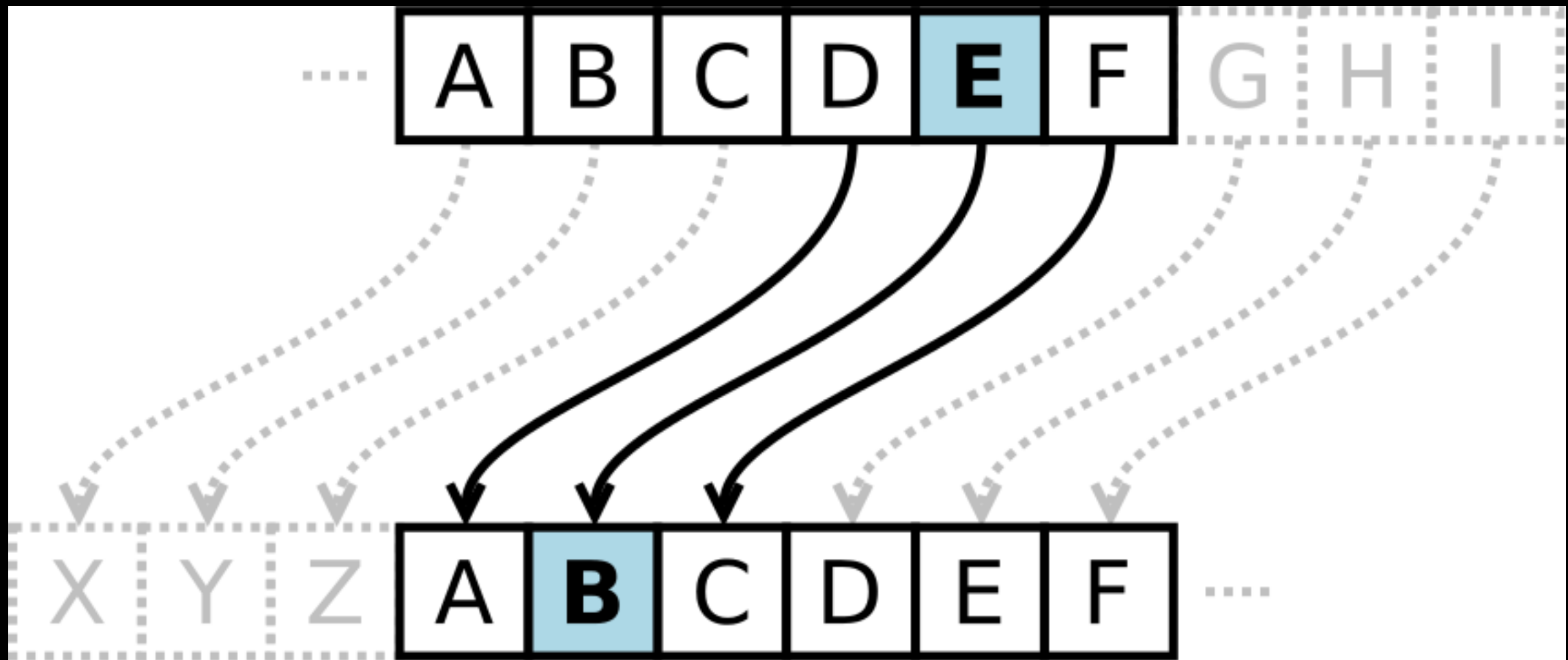
**Just kidding.**



# History of Passwords

- Much longer history than being used in the digital era
- Sentries for centuries would use “watchwords” to identify friend or foe
- Passwords also used as keys to encrypt messages. Think about the password you use to encrypt your SSH key
- One of earliest ciphers, Caesar cipher

# Caesar Cipher





# Passwords in Popular Culture (Prohibition)



# Computer Usage

- Early computers didn't have passwords
- MIT's Compatible Time-Sharing System (1961) — stored in plaintext
- 1962 Allan Scherr performed privilege escalation by punch card (#hardcore)
- Discovered that he could enter in command via punch card to print other user passwords
- Like a boss





# Password Hacking

- Bruteforce
  - Attempt every single combination
  - a -> z
  - 0000 -> 9999
- Dictionary
  - Go through every line of this text file
- Hybrid (mask)
  - Go through every line of this text file and add stuff
  - Password1!
  - Password1!!
  - Etc

# Password Cracking (Modern Day)

- John The Ripper — <http://www.openwall.com/john/>
  - Older than hashcat
  - Originally developed on Unix
- Hashcat — <https://hashcat.net/hashcat/>
  - Had proprietary codebase until about 2015
  - Can be used for CPU and GPU cracking
  - Used to come in 2 variants
    - hashcat (CPU based)
    - oclHashcat/cudaHashcat (GPU based)
  - Now just one version, hashcat



# Notice

- Due to time constraints, I'll focus on john the ripper (JtR) for this talk

# hashcat 101

- Typical choice when using GPU (discussed briefly later)
- <https://hashcat.net/wiki/doku.php?id=hashcat#options>
- RTFM, it's a good tool.

```
- [ Hash modes ] -
```

#	Name	Category
900	MD4	Raw Hash
0	MD5	Raw Hash
5100	Half MD5	Raw Hash
100	SHA1	Raw Hash
1300	SHA-224	Raw Hash
1400	SHA-256	Raw Hash
10800	SHA-384	Raw Hash
1700	SHA-512	Raw Hash
5000	SHA-3 (Keccak)	Raw Hash
600	BLAKE2b-512	Raw Hash
10100	SipHash	Raw Hash
6000	RIPEMD-160	Raw Hash
6100	Whirlpool	Raw Hash
6900	GOST R 34.11-94	Raw Hash
11700	GOST R 34.11-2012 (Streebog) 256-bit	Raw Hash
11800	GOST R 34.11-2012 (Streebog) 512-bit	Raw Hash
10	md5(\$pass.\$salt)	Raw Hash, Salted and/or Iterated
20	md5(\$salt.\$pass)	Raw Hash, Salted and/or Iterated
30	md5(utf16le(\$pass).\$salt)	Raw Hash, Salted and/or Iterated
40	md5(\$salt.utf16le(\$pass))	Raw Hash, Salted and/or Iterated
3800	md5(\$salt.\$pass.\$salt)	Raw Hash, Salted and/or Iterated
3710	md5(\$salt.md5(\$pass))	Raw Hash, Salted and/or Iterated
4010	md5(\$salt.md5(\$salt.\$pass))	Raw Hash, Salted and/or Iterated
4110	md5(\$salt.md5(\$pass.\$salt))	Raw Hash, Salted and/or Iterated
2600	md5(md5(\$pass))	Raw Hash, Salted and/or Iterated
3910	md5(md5(\$pass).md5(\$salt))	Raw Hash, Salted and/or Iterated
4300	md5(strtoupper(md5(\$pass)))	Raw Hash, Salted and/or Iterated
4400	md5(sha1(\$pass))	Raw Hash, Salted and/or Iterated
110	sha1(\$pass.\$salt)	Raw Hash, Salted and/or Iterated

```
- [ Outfile Formats ] -
```

#	Format
1	hash[:salt]
2	plain
3	hash[:salt]:plain
4	hex_plain
5	hash[:salt]:hex_plain
6	plain:hex_plain
7	hash[:salt]:plain:hex_plain
8	crackpos
9	hash[:salt]:crack_pos
10	plain:crack_pos
11	hash[:salt]:plain:crack_pos
12	hex_plain:crack_pos
13	hash[:salt]:hex_plain:crack_pos
14	plain:hex_plain:crack_pos
15	hash[:salt]:plain:hex_plain:crack_pos

# John the Ripper

- There are two main versions, it is important to know this
- John regular (original)
- John “jumbo” —  
<https://github.com/magnumripper/JohnTheRipper>
- John jumbo has numerous added features, including a lot of python/perl scripts to convert files into john password hashes.




 1password2john.py

 7z2john.pl

 DPAPImk2john.py

 aix2john.pl

 aix2john.py

 alnum.chr

 alnumspace.chr

 alpha.chr

 androidfde2john.py

 apex2john.py

 aruba2john.py

 ascii.chr

 axcrypt2john.py

 benchmark-unify


 best64.conf

 bestcrypt2john.py

 bitcoin2john.py

 electrum2john.py


 encfs2john.py

 enpass2john.py

 ethereum2john.py

 filezilla2john.py

 fuzz.dic

 fuzz\_option.pl

 geli2john.py

 genincstats.rb

 hextoraw.pl

 htdigest2john.py

 hybrid.conf

 ibmiscanner2john.py

 ikescan2john.py


 ios7tojohn.pl

 itunes\_backup2john.pl

 iwork2john.py

 sap2john.pl

 sha-dump.pl

 sha-test.pl

 sipdump2john.py

 snmp2john.lua

 ssh2sshng.py

 sshng2john.py

 stats

 strip2john.py

 sxc2john.py

 truecrypt2john.py

 unrule.pl

 upper.chr

# JtR Options

- --pot=potfile.pot
  - Specify which pot file you want to use
- --format=format
  - Specify the format of the hash
- --wordlist=/path/to/wordlist
  - Which wordlist you want to use
- --session=session-name
  - Name of session (think log)
- --fork=#
  - How many concurrent threads (for CPU cracking, good to specify # of threads)
- --crack-status
  - Will provide you with passwords cracked as it occurs

# JtR Example

- `./john --crack-status --fork=4 --session=myhashes --  
format=netntlmv2 --  
wordlist=/usr/share/dict/rockyou.txt  
/path/to/cracked/hashes.txt`

# JtR

- Don't know the format?
- `./john --list=formats`
- Here are some common ones:
  - NT (NTLM)
  - LM (LanMan)
  - netlm (Network LM)
  - netlmv2 (Network LM v2)
  - netntlm (Network NTLM)
  - netntlmv2 (Network NTLMv2)
  - krb5pa-md5 (Microsoft Kerberos)



# JtR Show Hashes

- `./john --show --pot=pot-file /path/to/ hashes.txt`
- There are pros and cons to using separate pot files, I like them isolated
- If you want to use the builtin john.pot file, be sure to backup on system if you change builds, it stores cracked hashes and can make lookups for common passwords over time easier
- Up to you

# JtR

- Hash formats, they matter
- Here are two resources for help:
  - <http://pentestmonkey.net/cheat-sheet/john-the-ripper-hash-formats>
  - <http://www.openwall.com/john/doc/EXAMPLES.shtml>

# hashid

- <https://github.com/psypanda/hashID>
- Helps identify different hashes

```
$ ./hashid.py '$P$8ohUJ.1sdFw09/bMaAQPTGDNi2BIUt1'
Analyzing '$P$8ohUJ.1sdFw09/bMaAQPTGDNi2BIUt1'
[+] Wordpress ≥ v2.6.2
[+] Joomla ≥ v2.5.18
[+] PHPass' Portable Hash

$ ./hashid.py -mj '$racf$*AAAAAAAA*3c44ee7f409c9a9b'
Analyzing '$racf$*AAAAAAAA*3c44ee7f409c9a9b'
[+] RACF [Hashcat Mode: 8500][JtR Format: racf]

$ ./hashid.py hashes.txt
--File 'hashes.txt'--
Analyzing '*85ADE5DDF71E348162894C71D73324C043838751'
[+] MySQL5.x
[+] MySQL4.1
Analyzing '$2a$08$VPzNKPAY60FsAbnq.c.h5.XTCZtC1z.j3hnLDFGImN9FcpfR1QnLq'
[+] Blowfish(OpenBSD)
[+] Woltlab Burning Board 4.x
[+] bcrypt
--End of file 'hashes.txt'--
```

# Wordlists



# A couple favs

- Rockyou wordlist - (built into Kali)
  - From rockyou breach
- UNIQPASS - <https://dazzlepod.com/uniqpass/>
  - Paid but some good results
- CrackStation - <https://crackstation.net/buy-crackstation-wordlist-password-cracking-dictionary.htm>
- WeakPass 2 - <https://weakpass.com/download>

# My Wordlist

## “Hack Back” Wordlist

*On August 12, 2015, in Information Security, by Elliott*

**Last Updated:** 8/12/2015

Provided here is a consolidated wordlist of all unique passwords gathered from my 9 honeypot systems in the USA, China, Singapore and Russia. Any additional wordlist files found on HFS systems have also been included in this. There are approximately ~900,000 unique passwords. This wordlist has been named the “Hack Back” wordlist due to the way it was gathered 😊

[Download Hack Back Wordlist](#)

Enjoy!

- <http://elliottbrink.com/2015/08/12/hack-back-wordlist/>

# Massive wordlist collection

- [wordbook.xyz/download/](http://wordbook.xyz/download/)
- Lots of good material

# Working With Wordlists

- Removing lines/words from a file that contain non-ascii characters
  - `perl -nle 'print if m{^[[:ascii:]]+$}' input-file.txt >> output-file.txt`
- Keeping lines/words from a file that are between 1 and 24 characters long
  - `egrep -x '{1,24}' input-file.txt >> output-file.txt`
- Sorting wordlists
  - `sort --parallel=4 -S 70% -T /media/external/temp /path/to/large/wordlist.txt >> /path/to/large/wordlist-sorted.txt`
- Remove duplicates
  - `uniq /path/to/large/wordlist-sorted.txt >> /path/to/large/wordlist-sorted-uniq.txt`



# KoreLogic Custom Rules

- <http://contest-2010.korelogic.com/rules.html>
- JtR rules created from DEFCON "Crack Me If You Can" 2010

## **KoreLogicRulesAppendNumbers\_and\_Specials\_Simple:**

This rule is a "catch all" for the most common patterns for appending numbers and/or specials to the end of a word. Use this rule `_first_` before attempting other rules that use special characters

## **KoreLogicRulesPrependSeason:**

This rule prepends any word in the wordlist with a season (Fall FALL Winter WINTER etc). Use this rule with wordlists such as 2letters.dic or 2EVERYTHING.dic

## **KoreLogicRulesAppendSeason:**

This rule appends any word in the wordlist with a season (Fall FALL Winter WINTER etc). Use this rule with wordlists such as 2letters.dic or 2EVERYTHING.dic

## **KoreLogicRulesPrependHello:**

This rule prepends any word in the wordlist with the word 'Hello' (Hello hEllo heLLo hello etc). Use this rule with wordlists such as 2letters.dic or 2EVERYTHING.dic

## **KoreLogicRulesPrependYears:**

This rule prepends any word in the wordlist with a year (from 1949 to 2019). These are common birth years of users, or their family members.

# RSMangler

- RSMangler
- Will add randomization to wordlists similar to masks. Nice to run from time to time for 3-4 input words.
- Provides added storage tradeoff
- <https://github.com/digininja/RSMangler>

# GPU Cracking

- There are lots of guides out there
- Won't talk about it in too much depth here
- If you want to play with it, I recommend trying out hashcat with AWS GPU instance
- High level advice:
  - 1x Single Nvidia GEFORCE GTX 1080 Ti
  - SSD
  - Intel i5 or i7
  - 16GB RAM
  - Mobo and case (special deals)

# References

- <https://www.trustedsec.com/2015/05/passwordstorage/>
- [https://en.wikipedia.org/wiki/Password\\_cracking](https://en.wikipedia.org/wiki/Password_cracking)
- <https://beerinfood.files.wordpress.com/2007/12/april7behindhotelunloadingbeer2.jpg>
- <https://i.pinimg.com/originals/8e/b2/a8/8eb2a8c66b9e27c3ec56cb03faf13ab3.gif>
- <https://securecdn.pymnts.com/wp-content/uploads/2016/08/password-hacking.jpg>
- [https://lh3.googleusercontent.com/Zqsi0CbSuK99tFTolsG9tRseaRH4cA\\_tPWcFsUP375reetJ0l3vI-JnfZJlkP1vPEw=h900](https://lh3.googleusercontent.com/Zqsi0CbSuK99tFTolsG9tRseaRH4cA_tPWcFsUP375reetJ0l3vI-JnfZJlkP1vPEw=h900)
- <https://twitter.com/darkstockphotos>
- (And all other links references within the talk)

# Thank You

