Hashes, hashes everywhere, but all I see is plaintext...
Will Hunt

- Co-founder of in.security
- 10+ years in cyber
- Hacker, formerly digital forensics
- Trainer/speaker @ Black Hat USA/EU, Nolacon, 44CON, Infosec Europe IICSG etc
- Assists UK government
- @Stealthsploit
Agenda

• Traditional cracking recap
• Password guidelines – old vs new
• Passphrases
• Creative / advanced cracking
• GPUs vs CPUs: A crypto-wallet story
• Foreign language cracking in h3x
• Secure p@ssw0rd advice
Before We Start

92f0bda3efc61c339dc50c06a264e444: discussing offline password attacks

Session..........: hashcat
Status..........: Cracked
Hash.Name........: NTLM
Hash.Target......: 92f0bda3efc61c339dc50c06a264e444
Time.Started.....: Fri Apr 26 22:54:57 2019 (1 sec)
Time.Estimated...: Fri Apr 26 22:54:58 2019 (0 secs)
Guess.Base.......: File (dictionary)
Guess.Mod.........: Rules (rules/OneRuleToRuleThemAll.rule)
Common Techniques

Dictionary + Rules

Mask / Hybrid

Success criteria
Algorithm complexity
Password length / complexity
Known / predictable elements
Hardware

Rainbow Table

Pure Brute Force
Common Techniques

• Dictionary and rules
  
  `hashcat -m1000 hashes.txt rockyou.txt -r OneRuleToRuleThemAll.rule`

• Mask
  

• Dictionaries
  
  `https://github.com/xajkep/wordlists`

• OneRuleToRuleThemAll
  
  `https://github.com/stealthsploit/Optimised-hashcat-Rule`
Common Techniques

• Hybrid

hashcat -m0 hashes.txt -a6 rockyou.txt ?d?d?d

• Pure brute force


• Rainbow tables – rarely situationally useful for single unsalted hashes
Key Space

• Key space = char set ^ length
  • Mixed alpha-numeric + special = 95 printable ASCII chars
  • 9 chars using mixed alpha-numeric + special
  • 10 chars using mixed alpha-numeric

Which password is stronger?

9 chars = 95^9 = 630,249,409,724,609,375
10 chars = 62^10 = 839,299,365,868,340,224
It's all about the length
Pure Brute Force

- 8x NVIDIA GTX 2080 Ti cracking NTLM @ 822 GH/s* (95 char set)

  - 8 char = <2.5 hours
  - 9 char = 9 days
  - 10 char = 2.3 years
  - 11 char = 219 years
  - 12 char = 20,840 years
  - 13 char = 2 million years

* https://twitter.com/hashcat/status/1095807014079512579/
Pure Brute Force

Estimated Password Recovery Times — 1x Terahash Brutalis, 44x Terahash Inmanis (448x Nvidia RTX 2080)

Full US keyboard mask attack with Terahash Hashstact

<table>
<thead>
<tr>
<th></th>
<th>NTLM</th>
<th>MD5</th>
<th>NetNTLMv1 / NetNTLMv1-ESS</th>
<th>LM</th>
<th>SHA1</th>
<th>SHA2-256</th>
<th>NetLMv2</th>
<th>SHA2-512</th>
<th>decrypt, DEB (Unix), Traditional DEB</th>
<th>Kerberos 5, etype 23, TGS-REP</th>
<th>Kerberos 5, etype 23, AS-REQ Pre-Auth</th>
<th>md5crypt, MDS (Unix), Cisco-IOS S15 (MD5)</th>
<th>LastPass + LastPass snifed</th>
<th>macOS 10.8+ (PBKDF2-SHA512)</th>
<th>WPA-EAPOL-PBKDF2</th>
<th>TrueCrypt RIFEMD160 + XTS S12 stolen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>31.82 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>3 mins 20 sec</td>
<td>5 hrs 30 mins</td>
<td>3 mins O day</td>
<td>5 yrs 7 mos</td>
<td>520 yrs 1 mo</td>
<td>51.2 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>17.77 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>6 mins 14 sec</td>
<td>9 hrs 50 mins</td>
<td>1 mo 1 wk</td>
<td>10 yrs 1 mo</td>
<td>960 yrs 4 mos</td>
<td>91.6 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>16.82 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>6 mins 35 sec</td>
<td>10 hrs 24 mins</td>
<td>1 mo 1 wk</td>
<td>10 yrs 8 mos</td>
<td>1 mil</td>
<td>96.8 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>15.81 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 hr 30 mins</td>
<td>5 days 23 hrs</td>
<td>1 yr 6 mos</td>
<td>547 yrs 10 mos</td>
<td>14.1 mil</td>
<td>135.5 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>5.89 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>18 mins 47 sec</td>
<td>1 day 5 hrs</td>
<td>3 mins 3 wks</td>
<td>30 yrs 7 mos</td>
<td>2.9 mil</td>
<td>278.3 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>2.42 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>45 mins 36 sec</td>
<td>3 days 0 hr</td>
<td>5 mins 1 wk</td>
<td>74 yrs 4 mos</td>
<td>7.1 mil</td>
<td>671.9 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>1.32 TH/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 hr 30 mins</td>
<td>5 days 23 hrs</td>
<td>1 yr 6 mos</td>
<td>547 yrs 10 mos</td>
<td>14.1 mil</td>
<td>135.5 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>200.19 Gbps</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 min 19 secs</td>
<td>2 hrs 17 mins</td>
<td>1 wk 2 days</td>
<td>2 yrs 4 mos</td>
<td>214 yrs 9 mos</td>
<td>21.4 mil</td>
<td>202.9 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>206.78 Gbps</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>5 mins 38 sec</td>
<td>8 hrs 54 mins</td>
<td>1 yr 4 days</td>
<td>273 yrs 3 mos</td>
<td>26.5 mil</td>
<td>251.3 mil</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>7.81 Gbps</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 min 37 secs</td>
<td>2 hrs 30 mins</td>
<td>1 wk 3 days</td>
<td>2 yrs 7 mos</td>
<td>249 yrs 5 mos</td>
<td>23.7 mil</td>
<td>252.6 mil</td>
<td>213.9 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>1.78 Gbps</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>6 mins 52 secs</td>
<td>10 hrs 52 mins</td>
<td>1 yr 1 wk</td>
<td>9 yrs 2 mos</td>
<td>69 yrs 7 mos</td>
<td>9.3 mil</td>
<td>90.9 mil</td>
<td>91.2 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>339.00 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>10 mins 34 secs</td>
<td>2 days 9 hrs</td>
<td>1 yr 7 mos</td>
<td>57 yrs 7 mos</td>
<td>5.7 mil</td>
<td>538.2 mil</td>
<td>5117.7 mil</td>
<td>485.7 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>277.23 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>57 mins 52 secs</td>
<td>3 days 19 hrs</td>
<td>11 mos 3 wks</td>
<td>4 yrs 3 mos</td>
<td>9 mil</td>
<td>650.5 mil</td>
<td>61799.3 mil</td>
<td>5679931 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>72 yrs 8 mos</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 hr 7 mins</td>
<td>4 days 10 hrs</td>
<td>1 yr 1 mos</td>
<td>10 yrs 1 mos</td>
<td>596 yrs 2 mos</td>
<td>91.1 mil</td>
<td>865.4 mil</td>
<td>80995.9 mil</td>
<td>785455.6 mil</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>181.51 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 hr 4 mins</td>
<td>6 hrs 18 mins</td>
<td>1 yr 9 mos</td>
<td>167 yrs 2 mos</td>
<td>15.9 mil</td>
<td>1509.7 mil</td>
<td>143419.6 mil</td>
<td>1306481.4 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>119.46 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 min 9 secs</td>
<td>4 hrs 24 mins</td>
<td>2 yrs 9 mos</td>
<td>402 yrs 10 mos</td>
<td>40.2 mil</td>
<td>381.1 mil</td>
<td>362723.1 mil</td>
<td>34549506.1 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>47.13 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>30 mins 38 secs</td>
<td>4 days 5 mos</td>
<td>4 yrs 5 mos</td>
<td>217 yrs 8 mos</td>
<td>217 yrs 8 mos</td>
<td>217 yrs 8 mos</td>
<td>217 yrs 8 mos</td>
<td>217 yrs 8 mos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>26.15 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>2 hrs 15 mins</td>
<td>4 mins 0 sec</td>
<td>7 yrs 5 mos</td>
<td>7 yrs 5 mos</td>
<td>7 yrs 5 mos</td>
<td>7 yrs 5 mos</td>
<td>7 yrs 5 mos</td>
<td>7 yrs 5 mos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>27.51 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>2 hrs 27 mins</td>
<td>4 mins 1 day</td>
<td>7 yrs 6 mos</td>
<td>717 yrs 10 mos</td>
<td>48.9 mil</td>
<td>648.1 mil</td>
<td>61799.7 mil</td>
<td>5656978.5 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>20.41 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>2 hrs 28 mins</td>
<td>4 mins 2 days</td>
<td>7 yrs 6 mos</td>
<td>717 yrs 10 mos</td>
<td>48.9 mil</td>
<td>648.1 mil</td>
<td>61799.7 mil</td>
<td>5656978.5 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>13.76 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 hr 29 mins</td>
<td>4 mins 2 days</td>
<td>17 yrs 5 mos</td>
<td>18 yrs 5 mos</td>
<td>1.8 mil</td>
<td>167 mil</td>
<td>150683.2 mil</td>
<td>14339159.0 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>8.78 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>1 hr 59 mins</td>
<td>4 mins 2 days</td>
<td>17 yrs 5 mos</td>
<td>18 yrs 5 mos</td>
<td>1.8 mil</td>
<td>167 mil</td>
<td>150683.2 mil</td>
<td>14339159.0 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>5.55 MB/s</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>Instant</td>
<td>29 mins 18 secs</td>
<td>2 days 9 hrs</td>
<td>6 months 2 wks</td>
<td>6 yrs 8 mos</td>
<td>4.6 mil</td>
<td>504.1 mil</td>
<td>507437.3 mil</td>
<td>45090289.7 mil</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

https://twitter.com/TerahashCorp/status/1155128018156892160
Guidelines – Old vs New

Old Guidelines
• Minimum 8 chars
• Mixed alpha-numeric and special
• Change password every 90 days

New Guidelines
• NIST SP 800-63B*
• “..at least 64 characters” (still min 8)
• Complexity recommended not required
• No truncation, password hints or KBA
• No expiry without justification
• Compare to dictionaries / breaches


tl;dr – passphrases over passwords
Guidelines – Old vs New

- **Tr0ub4dor &3**
  - **Common Substitutions**
  - **NUMERAL**
  - **PUNCTUATION**
  - **CAPS?**
  - **UNCOMMON (NON-GIBBERISH) BASE WORD**

  *Difficulty to guess:* EASY
  *Difficulty to remember:* HARD

- **correct horse battery staple**
  - **FOUR RANDOM COMMON WORDS**

  *Difficulty to guess:* HARD
  *Difficulty to remember:* YOU'VE ALREADY MEMORIZED IT

*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.*

*~28 bits of entropy*

*~44 bits of entropy*

~2⁸ = 3 days at 1000 guesses/sec (probable attack on a web remote, web service Vick, cracking it takes 3 days under a bruteforce attack, but it's not what the average user is going to want)

~2⁴⁴ = 530 years at 1000 guesses/sec

Was it trombone? No, troubador, and one of the Os was a zero...

AND THERE WAS SOME SYMBOL...

Thats a battery staple.

Correct!
SOME RANDOM WORDS YOU SAY?

EXCELLENT...
Attacking Passphrases

- Combinator attacks - dictionary combined with a dictionary
  
  - Google top 10,000 words [https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt](https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt)
  
  - Google top 20,000 words [https://github.com/first20hours/google-10000-english/blob/master/20k.txt](https://github.com/first20hours/google-10000-english/blob/master/20k.txt)
  
- @netmux has a great guide - [https://www.netmux.com/blog/cracking-12-character-above-passwords](https://www.netmux.com/blog/cracking-12-character-above-passwords)

```
hashcat -m0 hashes.txt -a1 20k.txt 20k.txt
```
Moar Words

• Combinator - https://github.com/hashcat/hashcat-utils

  ./combinator 20k.txt 20k.txt > 20k-combined

  ▶ head -5 20k-combined
  thethe
  theof
  theand
  theto
  thea

• 3 or 4 words

  hashcat -m0 hashes.txt -a1 20k-combined 20k-combined.txt

  ▶ hashcat -a1 20k-combined 20k.txt
     --stdout | more
  thethethe
  thetheof
  thetheand
  thetheto

  ▶ hashcat -a1 20k-combined 20k-combined
     --stdout | more
  thethethe
  thetheof
  thetheand
  thetheto
NOT SURE IF MY PASSPHRASE IS A PHRASE

OR JUST WORDS WITHOUT SPACES
Delimiters

awk '{print $0 " "}' 20k.txt > 20k-space

head -2 20k-space
the
of

./combinator 20k-space 20k.txt > 20k-combined-mid-space

head -2 20k-combined-mid-space
the the
the of

awk '{print length}' 20k-combined-mid-space head -2
Combinator rules

-j, --rule-left | Rule | Single rule applied to each word from left wordlist | -j 'c'
-k, --rule-right | Rule | Single rule applied to each word from right wordlist | -k '^-'

hashcat -m0 hashes -a1 20k-combined-mid-space -j '$ ' 20k.txt
  the the the
  the the of

hashcat -m0 hashes -a1 20k-combined-mid-space -j ' $ ' 20k-combined-mid-space
  the the the the
  the the the of
Add another space and pipe to hashcat

```
awk '{print $0 " "}' 20k-combined-mid-space > 20k-combined-mid-end-space

head -2 20k-combined-mid-end-space
the the
the of

./combinator 20k-combined-mid-end-space 20k-combined-mid-space | hashcat -m1000 hashes.txt -r rules/OneRuleToRuleThemAll.rule -w4
```

- Optimised kernels (-O) increases speed but limits length
- E.g. MD5 optimised = 31 / NTLM optimised = 27
Expander

- [https://github.com/hashcat/hashcat-utils/](https://github.com/hashcat/hashcat-utils/)
- Splits candidates into single chars, mutates & reconstructs
- Recompile with **LEN_MAX 8** and always unique output

```
./expander < wordlist.txt | sort -u
```

```c
#include <ctype.h>
#include <errno.h>
#include <fcntl.h>
#include "utils.c"

#define LEN_MIN 1
#define LEN_MAX 4
```

```
  echo hashcat l ./expander l sort -u l head -10
  echo hashcat l ./expander8 l sort -u l head -10
```

```
a
as
ash
ashc
at
ath
c
ca
cat
cath
```

```
a
as
ash
ashc
ashca
ashcat
ashcath
at
ath
athas
```
Fingerprint Attack

https://hashcat.net/wiki/doku.php?id=fingerprint_attack

1) Expand previously cracked passwords
2) Combo the resulting file with itself
3) Update (expand) wordlist
4) Repeat and rinse

```bash
cut -d: -f2- < hashcat.potfile | ./expander | sort -u > word.list
hashcat -m1000 hashes.txt --remove -a1 word.list word.list -o word.list2

cut -d: -f2- < word.list2 | ./expander | sort -u > word.list3
hashcat -m1000 hashes.txt --remove -a1 word.list3 word.list3 -o word.list4
```
PRINCE

PRObability INfinite Chained Elements (PRINCE)

- princeprocessor - Jens Steube "Atom" @hashcat
- Single dictionary input --> builds chains of 1 to n words
- E.g. if princeprocessor was guessing 4 char candidates...
  - 4 letter word
  - 2 letter word + 2 letter word
  - 1 letter word + 3 letter word
  - 3 letter word + 1 letter word
  - 1 letter word + 1 letter word + 2 letter word
  - 1 letter word + 2 letter word + 1 letter word
  - 2 letter word + 1 letter word + 1 letter word
  - 1 letter word + 1 letter word + 1 letter word + 1 letter word

https://github.com/hashcat/princeprocessor
PRINCE Attacks

• De-dupe dictionary before running prince

PRINCE Attack

./pp.bin rockyou.txt --pw-min=8 | hashcat -m1000 hashes.txt

PRINCEPTION - @jmgosney

./pp.bin rockyou.txt --pw-min=8 | ./pp.bin --pw-min=8 | hashcat -m1000 hashes.txt -g 300000

• Delimiter request open:
  https://github.com/hashcat/princeprocessor/issues/49
Passphrases with PRINCE

• Traditional delimited combinator

-j, --rule-left | Rule | Single rule applied to each word from left wordlist | -j 'c'
-k, --rule-right | Rule | Single rule applied to each word from right wordlist | -k '^-'

hashcat -m0 hashes -a1 20k-combined-mid-space -j '$ ' 20k-combined-mid-space

• pp variant (thanks @TychoTithonus & @Chick3nman512)

./pp.bin --elem-cnt-min=4 --elem-cnt-max=4 --pw-min=8 20k-space |
hashcat -j ] -m1000 hashes.txt -w3 -O -r prince_optimized.rule

• If dictionary & combo attacks fail – USE PRINCE
Purple Rain

...and as a last resort

• Shuffled wordlist --> different output

• https://www.netmux.com/blog/purple-rain-attack

```bash
shuf rockyou.txt | ./pp.bin --pw-min=8 | hashcat -m0 hashes --remove -w3 -O -g 300000
```
<table>
<thead>
<tr>
<th>Attack (Lastfm-top500k (358863 deduped))</th>
<th>Cracked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rockyou.txt + OneRuleToRuleThemAll</td>
<td>290337/358863 (80.9%)</td>
</tr>
<tr>
<td>5-char brute force -- Fingerprint attack 1</td>
<td>615/68526 (0.9%)</td>
</tr>
<tr>
<td>Fingerprint attack 2</td>
<td>9194/67911 (13.54%)</td>
</tr>
<tr>
<td>Fingerprint attack 3</td>
<td>5860/58717 (9.98%)</td>
</tr>
<tr>
<td>Fingerprint attack 4</td>
<td>805/52857 (1.52%)</td>
</tr>
<tr>
<td>Fingerprint attack 5</td>
<td>84/52052 (0.16%)</td>
</tr>
<tr>
<td>Fingerprint attack 6</td>
<td>9/51968 (0.02%)</td>
</tr>
<tr>
<td>Fingerprint attack 7</td>
<td>0/51959</td>
</tr>
<tr>
<td>2hr - Prince</td>
<td>82/51959 (0.16%)</td>
</tr>
<tr>
<td>2hr - Prince + prince optimized rule</td>
<td>3325/51878 (6.41%) – 1 in potfile</td>
</tr>
<tr>
<td>2hr - Purple rain + 300000 self-generated rules</td>
<td>4298/48558 (8.85%) – 6 in potfile</td>
</tr>
<tr>
<td>2hr - Purple rain + 300000 self-generated rules</td>
<td>1610/44270 (3.64%) – 10 in potfile</td>
</tr>
<tr>
<td>2hr - Purple rain + OneRuleToRuleThemAll</td>
<td>7011/42660 (16.43%) – 1 in potfile - 35649 (Total 323214 (90% cracked))</td>
</tr>
</tbody>
</table>
Crypto Wallet Example

- Ethereum wallets - MyEtherWallet, Geth, Mist
- JSON keystore file option
- scrypt used in testing

~ git:(master) x hashcat --help|grep -i ethereum
16300 | Ethereum Pre-Sale Wallet, PBKDF2-HMAC-SHA256
15600 | Ethereum Wallet, PBKDF2-HMAC-SHA256
15700 | Ethereum Wallet, SCRYPT
Crypto Wallet Example

- Ethereum wallets – MyEtherWallet, Geth, Mist
- JSON keystore file option
- scrypt used in testing

```bash
~ git:(master) × hashcat --help|grep -i ethereum
16300 | Ethereum Pre-Sale Wallet, PBKDF2-HMAC-SHA256
15600 | Ethereum Wallet, PBKDF2-HMAC-SHA256
15700 | Ethereum Wallet, SCRYPT
```
Ethereum Wallets

- GPU cracking viability depends on $n$, $r$

- [https://stealthsploit.com/2017/06/12/ethereum-wallet-cracking/](https://stealthsploit.com/2017/06/12/ethereum-wallet-cracking/)

---

UTC-2017-06-10T11-51-33.675Z-f418f8185f2c1163ae953bf778acc6877b9bc203

{"version":3,"id":"5cf4711d-3f69-4636-89d0-b304a7e23b75","address":"f418f8185f2c1163ae953bf778acc6877b9bc203","Crypto":null,"ciphertext":"7f5c865554d67604394ae54d7a4f9735bdb85c90e606a672d18add1d167d793b","cipherparams":null,"cipher":"aes-128-ctr","kdf":"scrypt","kdfparams":null,"clklen":32,"salt":"437964c9bd1b5f63bde56560808c894792f8f670694590b776e22381e32dd33b","n":1024,"r":8,"p":1,"mac":"96f2a849321cc04cb6c0fcee1bd4b195ca681ca28064dc45000f02e47230c5b6"}

**hashcat Format**

```
$ethereum$s*1024*8*1*437964c9bd1b5f63bde56560808c894792f8f670694590b776e22381e32dd33b*7f5c865554d67604394ae54d7a4f9735bdb85c90e606a672d18add1d167d793b*96f2a849321cc04cb6c0fcee1bd4b195ca681ca28064dc45000f02e47230c5b6
```
Calculating Memory

• Determine memory requirements using \( n, r \)

**Step 1: Calculate Single Computation per GPU**

\[
\text{size}_\text{scrypt} = (128 \times r) \times n
\]

**Step 2: Calculate Parallel Computations per GPU**

Threads per compute unit(1) * no. of compute units(2) = no. parallel computations

**Step 3: Calculate RAM requirement per GPU**

\[
\text{size}_\text{scrypt} \times \text{no. parallel computations}
\]

<table>
<thead>
<tr>
<th></th>
<th>NVIDIA</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>(2)</td>
<td>Depends on GPU</td>
<td></td>
</tr>
</tbody>
</table>
Calculating Memory

• In earlier post: \( n = 1024, \ r = 8 \)

**Step 1: Calculate Single Computation per GPU**

\[
\text{size_scrypt} = (128 \times r) \times n \\
(128 \times 8) \times 1024 = 1048576 \text{ bytes (1 MB)}
\]

**Step 2: Calculate Parallel Computations per GPU**

Threads per compute unit(1) \times \text{no. of compute units(2)}

\(32 \times 68 = 2176\)

**Step 3: Calculate RAM requirement per GPU**

\[
\text{size_scrypt} \times \text{no. parallel computations} \\
1 \text{ MB} \times 2176 = 2176 \text{ MB per GPU}
\]

NVIDIA

<table>
<thead>
<tr>
<th></th>
<th>NVIDIA</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>(2)</td>
<td>Depends on GPU</td>
<td></td>
</tr>
</tbody>
</table>

https://www.techpowerup.com/gpu-specs/geforce-rtx-2080-ti.c3305
Calculating Memory

\[ \text{size}_{\text{scrypt}} = (128 \times r) \times n \]
\[ (128 \times 8) \times 131072 = 134217728 \text{ bytes (128 MB)} \]

Threads per compute unit(1) \times \text{no. of compute units(2)} = \text{no. parallel computations}
\[ 32 \times 68 = 2176 \]

\[ \text{size}_{\text{scrypt}} \times \text{no. parallel computations} \]
\[ 128 \text{ MB} \times 2176 = 278528 \text{ MB} = 272 \text{ GB RAM per GPU} \]
GPU vs CPU

- Cracking recent wallets on GPU will likely error, hang, blue screen etc.
- CPU to the rescue!

-D 1 -d <deviceID>
GPU vs CPU

- Convert with `ethereum2john` - `ethereum2john.py <wallet file>`
  
  ```
  → Dictionaries git:(master) × hashcat -m15700 "$ethereum$s*131072*8*1*919d207466fc4fe352fa75a6db8e2499c1f62f0c14db71a1da6286522ee3f270*2f8da158b01a9f144d9f89bfb04dac19643049c87488071100d068942c46c632*91bd22afeb668d89800627dc760730709de03df38c54aea2a939bc465b90efe4' rockyou.txt -r OneRuleToRuleThemAll.rule -w3 -D1 -d1
  hashcat (v5.1.0-905-g7c6970db) starting...
  ```

OpenCL Platform #1: Apple

* Device #1: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4096/16384 MB allocatable, 8MCU
* Device #2: Intel(R) HD Graphics 630, skipped
* Device #3: AMD Radeon Pro 555 Compute Engine, skipped

Time Started.....: Thu Jun 20 01:09:04 2019 (1 sec)
Time Estimated....:
Guess.Base.......: File (rockyou.txt)
Guess.Mod........: Rules (OneRuleToRuleThemAll.rule)
Guess.Queue.......: 1/1 (100.00%)
Speed.#1.........: ? (285.00ms) @ Accel:1 Loops:1 Thr:1 Vec:1
GPU vs CPU

- Convert with ethereum2john - `ethereum2john.py <wallet file>`
  - [GitHub Link](https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/run/ethereum2john.py)

  ```bash
  → Dictionaries git:(master) x hashcat -m15700 "\$ethereum\$s*131072*8*1*919d207466fc4fe352fa75a6db8e2499c1f62f0c14db71a1da6286522ee3f270*2f8da158b01a9f144d9f89bfb04dac19643049c87488071100d068942c46c632*91bd22afeb668d89800627dc760730709de03df38c54aea2a939bc465b90efe4' rockyou.txt -r OneRuleToRuleThemAll.rule -w3 -D1 -d1
  hashcat (v5.1.0-905-g7c697db0) starting...
  ```

OpenCL Platform #1: Apple

```
* Device #1: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4096/16384 MB
  * Device #2: Intel(R) HD Graphics 630, skipped.
  * Device #3: AMD Radeon Pro 555 Compute Engine, skipped.
```

```
Time.Started.....: Thu Jun 20 01:09:04 2019 (1 sec)
Time.Estimated...: Sun Nov 5 22:49:12 2975 (956 years, 138 days)
Guess.Base.......: File (rockyou.txt)
Guess.Mod........: Rules (OneRuleToRuleThemAll.rule)
Guess.Queue......: 1/1 (100.00%)
Speed.#1..........: 25 H/s (285.00ms) @ Accel:1 Loops:1 Thr:1 Vec:1
```
Non-ASCII Characters

• 256 ASCII characters – single byte encoding

• UTF-8 (1-4 byte encoding) for other characters

• Find the hex and pass to hashcat with \texttt{--hex-charset}

<table>
<thead>
<tr>
<th>Character</th>
<th>Hex Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>41</td>
<td>LATIN CAPITAL LETTER A</td>
</tr>
<tr>
<td>B</td>
<td>42</td>
<td>LATIN CAPITAL LETTER B</td>
</tr>
<tr>
<td>C</td>
<td>43</td>
<td>LATIN CAPITAL LETTER C</td>
</tr>
<tr>
<td>V</td>
<td>ce 92</td>
<td>GREEK CAPITAL LETTER BETA</td>
</tr>
<tr>
<td>Γ</td>
<td>ce 93</td>
<td>GREEK CAPITAL LETTER GAMMA</td>
</tr>
<tr>
<td>Δ</td>
<td>ce 94</td>
<td>GREEK CAPITAL LETTER DELTA</td>
</tr>
</tbody>
</table>

https://www.utf8-chartable.de/unicode-utf8-table.pl
Non-ASCII Characters

• E.g. Arabic alphabet uses the following UTF-8 (hex) chars

• First byte
  d8 d9 da db

• Second byte
  80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f 90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf

• The attack
  hashcat -m0 a095531811f39557054b5340e5d2b182 -1 d8d9dad0b 8081828384858687888898a8b8c8d8e8f909192939495969798999a9b9c9d 9e9fa0a1a2a3a4a5a6a7a8a9aaabacadaefb0b1b2b3b4b5b6b7b8b9babbb bcbdbefbf --hex-charset \1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?\1?
  w3 -a3
S3cure YOurs3lf

• Password managers – Lastpass, 1Password, Dashlane, Keepass etc
• Passphrases – 5+ random words
• Use spaces (other/differing delimiters)
• Avoid elements based on KBA
• 2FA / MFA
• Don’t reuse passwords
• HIBP – https://haveibeenpwned.com/

notify me of pwnage
Thank you!

https://in.security

contact@in.security

@Stealthsploit / @insecurity_ltd

Have a crack at our challenge
https://in.security/password-cracking-ctf/