Building Dictionaries and Destroying Hashes using Amazon EC2

Steve Werby
[President | Security Researcher | Security Consultant]
Befriend
whois –h stevewerby

- Security Researcher and Consultant at Befriend
- CISO at [UTSA | VCU | VDOC]
- In infosec since 1999

- ISACA Certified Information Security Manager (CISM), 2010 (not quite)
- (ISC)^2 Certified Information Systems Security Professional (CISSP), 2010
- GIAC Security Leadership Certification (GSLC), 2008
- GIAC Certified Forensics Analyst (GCFA), 2007
- GIAC Web Application Security Certificate (GWAS), 2007
- GIAC Security Essentials Certification (GSEC), 2007
- GIAC Certified Incident Handler (GCIH), 2006
- MBA, Virginia Commonwealth University
- BS, Industrial and Systems Engineering, Virginia Tech
whois –h stevewerby
whois –h stevewerby

Steve Werby
@stevewerby

A SQL Injection walks into a bar, starts to quote something but stops, drops a table, then dashes out.
The first rule of Presentation Club…

- Have a question? Ask away!
- Have a comment? Share!
- I will ask you some questions too.
Disclaimer (thank the lawyers)

The opinions shared represent my views, the views of my clients, the views of my past employers, and most importantly, the views of my future employers.
Disclaimer (OK, the real one)

The opinions shared represent my views, the views of my clients, the views of my past employers, and most importantly, the views of my future employers.
Takeaways

1. Can EC2 provide me value?
2. How can I crack passwords?
3. Buy or rent?
4. Passphrases or passwords?
5. What hashing algo should I use?
Hashing
Leaks

- RockYou – 32.6M plaintext
- eHarmony – 1.5M unsalted MD5
- LinkedIn – 6.5M unsalted SHA1
- Gawker – 1.3M unsalted DES
- IEEE – 100,000 plaintext
Cracking

- Online or offline
- Dictionary, rule-based, brute force, precomputation, etc.
Why do I care?

- Analyze external leaks to gather intelligence
- Assess password strength of user base
- Determine a password
- Strengthen your argument for…
  - a different password policy
  - a different hashing algorithm
- 2FA
Key space

- Scenario A
  - 5 LC or UC or numeric or symbol
  - \(95^5 = 7,737,809,375\)

- Scenario B
  - 8 LC or UC
  - \(52^8 = 53,459,728,531,456\)

- Scenario C
  - 15 LC
  - \(26^{15} = 1,677,259,342,285,730,000,000\)
Cracking tools

- John the Ripper
- Hashcat[-plus|-lite]
- Cryptohaze Multiforcer
Build or lease?

- Pros?
- Cons?
- When?
Amazon EC2 – getting started

- Create AWS account
  - Provide CC, phone #
- Create instance
  - Access EC2 Console
  - Create key pair
  - Create security group
  - Create a new instance
  - Login via SSH
- Configure system (install software and scripts)
Amazon EC2

Getting Started

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US East (N. Virginia) region.

Service Health

Service Status

Current Status Details

My Resources

You are using the following Amazon EC2 resources in the US East (N. Virginia) region:

- 2 Running Instances
- 2 Elastic IPs
- 4 EBS Volumes
- 0 EBS Snapshots
- 1 Key Pair
- 0 Load Balancers
- 0 Placement Groups
- 3 Security Groups
Amazon EC2

Create a New Instance

Select an option below:

- **Classic Wizard**
  Launch an On-Demand or Spot instance using the classic wizard with fine-grained control over how it is launched.

- **Quick Launch Wizard**
  Launch an On-Demand instance using an editable, default configuration so that you can get started in the cloud as quickly as possible.

- **AWS Marketplace**
  AWS Marketplace is an online store where you can find and buy software that runs on AWS. Launch with 1-Click and pay by the hour.

**Choose a Key Pair:**
Public/private key pairs allow you to securely connect to your instance after it launches.
- **Select Existing**
- **Create New**
- **None**

Choose a meaningful name, e.g. Web Server

**Name Your Instance:**

**Choose a Launch Configuration:**

- **Ubuntu Server 12.04 LTS**
  Ubuntu Server 12.04 LTS with support available from Canonical ([http://www.ubuntu.com/cloud/services](http://www.ubuntu.com/cloud/services)).

**Note:** You can customize your settings in the next step.
Amazon EC2

Create a New Instance

Your instance is now launching.
Instance: Yoda (i-6c156216)

Note: Your instance may take a few minutes to launch, depending on the software you are running.
Note: Usage hours on your new instance will start immediately and continue to accrue until you stop or terminate your instance.

You can perform the following tasks while your instance is launching:

- Create Status Check Alarm
  You can use status check alarms to be notified if this instance fails status checks (additional charges may apply).
- Create EBS Volumes (Additional charges may apply.)
- View your instances on the Instances page
- Submit feedback
  This dialog has recently changed and we would greatly appreciate any feedback you have on the new process of launching an instance.
# Amazon EC2

## Navigation

- **Region:** US East (N. Virginia)
- **EC2 Dashboard**
  - Events
- **INSTANCES**
  - Instances
  - Spot Requests
  - Reserved Instances
- **IMAGES**
  - AMIs
  - Bundle Tasks
- **ELASTIC BLOCK STORE**
  - Volumes
  - Snapshots
- **NETWORK & SECURITY**
  - Security Groups
  - Elastic IPs
  - Placement Groups
  - Load Balancers
  - Key Pairs

## My Instances

### Viewing: All Instances

<table>
<thead>
<tr>
<th>Name</th>
<th>Instance</th>
<th>AMI ID</th>
<th>Root Device</th>
<th>Type</th>
<th>State</th>
<th>Status Check</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zork</td>
<td>i-58f32222</td>
<td>ami-967edc1f</td>
<td>ebs</td>
<td>t1.micro</td>
<td>running</td>
<td>✔ 2/2 check</td>
<td>no</td>
</tr>
<tr>
<td>Xzibit</td>
<td>i-cc088eb5</td>
<td>ami-ecf6285</td>
<td>ebs</td>
<td>cg1.4xlarge</td>
<td>stopped</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vanguard</td>
<td>i-6c412c16</td>
<td>ami-ecf6285</td>
<td>ebs</td>
<td>cg1.4xlarge</td>
<td>running</td>
<td>✔ 2/2 check</td>
<td>no</td>
</tr>
<tr>
<td>Yoda</td>
<td>i-6c156216</td>
<td>ami-82fa58eb</td>
<td>ebs</td>
<td>m1.large</td>
<td>running</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Block Devices:

- sda1

### Network Interfaces:

- **Public DNS:** ec2-23-20-163-94.compute-1.amazonaws.com
- **Private DNS:** ip-10-157-13-46.ec2.internal
- **Private IPs:** 10.157.13.46
- **Secondary Private IPs:**
  - 2012-09-11 14:45 CDT (less than an hour)

**Product Codes:**
Amazon EC2 - pricing

- On demand | reserved | spot
- Configuration
- Linux | Windows
- Geographic region
- Data transfer, elastic IPs, metrics
- EBS storage and I/O
Amazon EC2 - pricing

<table>
<thead>
<tr>
<th>Standard On-Demand Instances</th>
<th>Linux/UNIX Usage</th>
<th>Windows Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small (Default)</td>
<td>$0.080 per Hour</td>
<td>$0.115 per Hour</td>
</tr>
<tr>
<td>Medium</td>
<td>$0.160 per Hour</td>
<td>$0.230 per Hour</td>
</tr>
<tr>
<td>Large</td>
<td>$0.320 per Hour</td>
<td>$0.460 per Hour</td>
</tr>
<tr>
<td>Extra Large</td>
<td>$0.640 per Hour</td>
<td>$0.920 per Hour</td>
</tr>
</tbody>
</table>
Amazon EC2 - pricing

Free tier (t1.micro)
Amazon EC2 - architecture

- Ubuntu Server 12.04 LTS
  - m1.large
  - 7.5 GB of RAM
  - 4 EC2 Compute Units, 64 bit

- Cluster GPU Amazon Linux AMI 2012.03
  - cg1.4xlarge
  - 22 GB of RAM
  - 33.5 EC2 Compute Units, 64 bit
Baseline - architecture

- Dell Latitude
  - Intel Core 2 Duo, 2 Ghz, 32 bit
  - 2 GB of RAM
Cost

- Zelda
  1 Dell Latitude
  $0.00+ / hour

- Yoda
  1 Ubuntu Server
  $0.32 / hour

- Xzibit
  1 GPU Amazon Linux Cluster AMI
  $2.10 / hour

- Wiggum
  Yoda + 5 GPU Amazon Linux Cluster AMIs
  Tell you later!
Testing - dataset

- First 1 million of 1.5 million eHarmony passwords posted online in June 2012
- Unsalted MD5s
Testing - Hashcat-plus tests

- Dictionary
- Combination
- Rules
- Masking (granular brute force)
Regular process – cyclical/learning

- Define sequence of jobs (tests) to run
- Run fast jobs against small key spaces
- Analyze results (during and after job)
- Eliminate or adjust jobs based on results
- Create new dictionaries
- Re-run jobs against new dictionaries
Hashcat-plus syntax

Combination attack:

```
/crack/oclHashcat-plus-0.09/cudaHashcat-plus64.bin \
-a 1 \
-m 0 \n/crack/ hashes_first_1000000.txt \
/crack/dict_combined_dict.txt \
/crack/dict_combined_words.txt \n-o /crack/hcp_4_2_found.txt \n> /crack/hcp_4_2_run_data.txt
```
Hashcat-plus syntax

Masking attack:

```
/crack/oclHashcat-plus-0.09/cudaHashcat-plus64.bin \ 
-a 3 \ 
-m 0 -a 3 -1 ?u -2 ?s \ 
?1?1?1?1?1?1?2 \ 
-o /crack/hcp_3_6_found.txt \ 
?1?1?1?1?1?1 \ > /crack/hcp_3_6_run_data.txt
```
Testing - Hashcat-plus tests

```
JOB=3_2

$HCP $FILE_HASHES \n-m 0 -a 3 -l ?u \n-o ${FILE_OUTPUT_ALL_PREFIX}${JOB}.txt \n?1?12121?1 > ${FILE_JOB_METRICS_PREFIX}${JOB}.txt
cat ${FILE_OUTPUT_ALL_PREFIX}${JOB}.txt | cut -d":" -f1 | sort > ${FILE_OUTPUT_HASHES_PREFIX}${JOB}.txt
cat ${FILE_OUTPUT_ALL_PREFIX}${JOB}.txt | cut -d":" -f2 | sort > ${FILE_OUTPUT_PWDS_PREFIX}${JOB}.txt

if [ -f ${FILE_OUTPUT_HASHES_PREFIX}${JOB}.txt ]; then
    awk 'NR==FN{a[$0];next} !($0 in a)' ${FILE_OUTPUT_HASHES_PREFIX}${JOB}.txt ${FILE_HASHES_AFTER_JOB_PREFIX}${JOB}.txt \n> ${FILE_HASHES_AFTER_JOB_PREFIX}${JOB}.txt
else
    cp ${FILE_HASHES_AFTER_JOB_PREFIX}${JOB}.txt ${FILE_HASHES_AFTER_JOB_PREFIX}${JOB}.txt
fi

JOB_MATCHES='cat ${FILE_JOB_METRICS_PREFIX}${JOB}.txt | grep "Recovered | cut -d":" -f2 | cut -d="/" -f1 | cut -d"" -f2
JOB_MATCHES='cat ${FILE_JOB_METRICS_PREFIX}${JOB}.txt | grep "Progress | cut -d ":" -f2 | cut -d="/" -f2 | cut -d"" -f1
JOB_TIME='cat ${FILE_JOB_METRICS_PREFIX}${JOB}.txt | grep "Time.Util | cut -d":" -f2 | cut -d="/" -f1 | cut -d"" -f2 | cut -d"" -f1

BATCH_MATCHES='cat ${FILE_STATS} | tail -1 | cut -d":" -f3
BATCH_MATCHES='expr $BATCH_MATCHES_PREV + $JOB_MATCHES
COLLECTION_MATCHES='cat ${FILE_STATS} | tail -1 | cut -d":" -f4
COLLECTION_MATCHES='expr $COLLECTION_MATCHES_PREV + $JOB_MATCHES

BATCH_MATCHES='cat ${FILE_STATS} | tail -1 | cut -d":" -f9
BATCH_strings_TESTED='expr $BATCH_strings_TESTED_PREV + $JOB_regexes_TESTED

BATCH_TIME='cat ${FILE_STATS} | tail -1 | cut -d":" -f12
BATCH_TIME='expr $BATCH_TIME_PREV + $JOB_TIME

COLLECTION_TIME='cat ${FILE_STATS} | tail -1 | cut -d":" -f13
COLLECTION_TIME='expr $COLLECTION_TIME_PREV + $JOB_TIME
```
Testing - dictionary

- Existing dictionaries
- Word lists of any kind
- Leaked password files
- Bigger != better
### Testing - masking

- Muuuuuuuuuuuuch better than brute force
- Can greatly reduce key space

<table>
<thead>
<tr>
<th>Xzibit</th>
<th>Strings</th>
<th>Time (s)</th>
<th>Rate 1</th>
<th>Cracked</th>
<th>Rate 2</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uppercase and lowercase (5)</td>
<td>380,204,032</td>
<td>27</td>
<td>14,081,631</td>
<td>14,896</td>
<td>552</td>
<td>3</td>
</tr>
<tr>
<td>Uppercase (6)</td>
<td>308,915,776</td>
<td>75</td>
<td>4,118,877</td>
<td>59,573</td>
<td>794</td>
<td>3.2</td>
</tr>
<tr>
<td>Uppercase (5), numeric (1)</td>
<td>118,813,760</td>
<td>10</td>
<td>11,881,376</td>
<td>2,855</td>
<td>286</td>
<td>3.3</td>
</tr>
<tr>
<td>Uppercase and numeric (5)</td>
<td>60,466,176</td>
<td>33</td>
<td>1,832,308</td>
<td>29,769</td>
<td>902</td>
<td>3.4</td>
</tr>
<tr>
<td>Uppercase and numeric (6)</td>
<td>2,176,782,336</td>
<td>139</td>
<td>15,660,305</td>
<td>166,216</td>
<td>1,196</td>
<td>3.5</td>
</tr>
<tr>
<td>Uppercase (5) and special (1)</td>
<td>392,085,408</td>
<td>8</td>
<td>49,010,676</td>
<td>0</td>
<td>0</td>
<td>3.6</td>
</tr>
<tr>
<td>Uppercase and numeric (5) and special (1)</td>
<td>1,995,383,808</td>
<td>10</td>
<td>199,538,381</td>
<td>0</td>
<td>0</td>
<td>3.7</td>
</tr>
<tr>
<td>Uppercase and numeric and special (6)</td>
<td>107,918,163,081</td>
<td>216</td>
<td>499,621,125</td>
<td>166,216</td>
<td>770</td>
<td>3.8</td>
</tr>
<tr>
<td>Uppercase and numeric (7)</td>
<td>78,364,164,096</td>
<td>218</td>
<td>359,468,643</td>
<td>170,721</td>
<td>783</td>
<td>3.9</td>
</tr>
<tr>
<td>Numeric (8)</td>
<td>100,000,000</td>
<td>9</td>
<td>11,111,111</td>
<td>0</td>
<td>0</td>
<td>3.10</td>
</tr>
<tr>
<td>Numeric (9)</td>
<td>1,000,000,000</td>
<td>12</td>
<td>83,333,333</td>
<td>3,787</td>
<td>316</td>
<td>3.11</td>
</tr>
<tr>
<td>Numeric (10)</td>
<td>10,000,000,000</td>
<td>22</td>
<td>454,545,455</td>
<td>8,553</td>
<td>389</td>
<td>3.12</td>
</tr>
<tr>
<td>Numeric (11)</td>
<td>100,000,000,000</td>
<td>68</td>
<td>1,470,588,235</td>
<td>153</td>
<td>2</td>
<td>3.13</td>
</tr>
<tr>
<td>T (1), uppercase and numeric (7)</td>
<td>78,364,164,096</td>
<td>65</td>
<td>1,205,602,525</td>
<td>7,897</td>
<td>121</td>
<td>3.14</td>
</tr>
<tr>
<td>Z (1) and uppercase and numeric (7)</td>
<td>78,364,164,096</td>
<td>57</td>
<td>1,374,809,896</td>
<td>1,277</td>
<td>22</td>
<td>3.15</td>
</tr>
<tr>
<td>X (1) and uppercase and numeric (7)</td>
<td>78,364,164,096</td>
<td>55</td>
<td>1,424,802,984</td>
<td>412</td>
<td>7</td>
<td>3.16</td>
</tr>
<tr>
<td>Q (1) and uppercase and numeric (7)</td>
<td>78,364,164,096</td>
<td>57</td>
<td>1,374,809,896</td>
<td>438</td>
<td>8</td>
<td>3.17</td>
</tr>
<tr>
<td>A (1) and uppercase and numeric (7)</td>
<td>78,364,164,096</td>
<td>66</td>
<td>1,187,335,820</td>
<td>8,461</td>
<td>128</td>
<td>3.18</td>
</tr>
<tr>
<td>Total, batch 3</td>
<td>3,515,745,706,409</td>
<td>2,852</td>
<td>1,232,729,911</td>
<td>648,191</td>
<td>227</td>
<td></td>
</tr>
<tr>
<td>Total, batch 3, unique</td>
<td>3,515,745,706,409</td>
<td>2,852</td>
<td>1,232,729,911</td>
<td>404,651</td>
<td>142</td>
<td></td>
</tr>
</tbody>
</table>
Testing - rules

- Highly flexible

- Prepend, append, duplicate, substitute, shift, etc.

- Used dictionaries, word lists, password dumps, and batch 3 (combination attack results) as input
A loud piercing alarm went off
Everyone left the room
It turns out there was a massive fire*
Eventually everyone returned

* There was no fire
Testing - combination

- Concatenates words from 2 lists

<table>
<thead>
<tr>
<th>Xzibit</th>
<th>Strings</th>
<th>Time (s)</th>
<th>Rate 1</th>
<th>Cracked</th>
<th>Rate 2</th>
<th>Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>combinator dict_combined_dict*2</td>
<td>149,044,639,969</td>
<td>319</td>
<td>467,224,577</td>
<td>123,927</td>
<td>388</td>
<td>5.1</td>
</tr>
<tr>
<td>combinator dict_combined_dict * dict_combiner</td>
<td>42,677,720,398</td>
<td>199</td>
<td>214,460,907</td>
<td>111,457</td>
<td>560</td>
<td>5.2</td>
</tr>
<tr>
<td>combinator dict_combined_words*2</td>
<td>12,220,418,116</td>
<td>185</td>
<td>66,056,314</td>
<td>102,498</td>
<td>554</td>
<td>5.3</td>
</tr>
<tr>
<td>combinator dict_combined_words * dict_combiner</td>
<td>42,677,720,398</td>
<td>285</td>
<td>149,766,837</td>
<td>112,461</td>
<td>395</td>
<td>5.4</td>
</tr>
<tr>
<td>Total, batch 5</td>
<td>246,620,498,881</td>
<td>988</td>
<td>249,615,890</td>
<td>450,343</td>
<td>456</td>
<td></td>
</tr>
<tr>
<td>Total, batch 5, unique</td>
<td>246,620,498,881</td>
<td>988</td>
<td>249,615,890</td>
<td>170,434</td>
<td>173</td>
<td></td>
</tr>
<tr>
<td>Total, batch 3 to 5</td>
<td>3,763,762,989,744</td>
<td>4,663</td>
<td>807,154,834</td>
<td><strong>557,202</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hashcat[-plus] speed

- Xzibit – 1.6B/s => 5.8T/h
- Yoda – 6.2M/s => 22.3B/h
- Zelda – 14k/s => 50.4M/h

- Xzibit = 114,286 * Zelda
- Xzibit = 258 * Yoda
- Yoda = 443 * Zelda
Hashcat[+plus] speed

- Xzibit = 114,286 * Zelda
- Xzibit = 258 * Yoda
- Yoda = 443 * Zelda

- Xzibit = $2.10 / hour
- Yoda = $0.32 / hour
- 1 hour of processing on Xzibit is equivalent to 258 hours on Yoda
  = 258 * $0.32 = $82.56
- = 3,831% more
What is your constraint?

- Deadline?
- Budget?
- Other?
My constraint is a deadline

- Utilize GPUs
- Analyze and optimize
- Split workload across servers
My constraint is budget

- EC2 spot instances
- DIY
Cost

- **Zelda**
  1 Dell Latitude
  $0.00+ / hour
- **Yoda**
  1 Ubuntu Server
  $0.32 / hour
- **Xzibit**
  1 GPU Amazon Linux Cluster AMI
  $2.10 / hour
- **Wiggum**
  Yoda + 5 GPU Amazon Linux Cluster AMIs
  $0.35 / hour per machine
My constraint is budget

- EC2 spot instances
- DIY
Distributed process – cyclical/learning

- Define sequence of jobs (tests) to run
- Run fast jobs against small key spaces
- Analyze results (during and after job)
- Eliminate or adjust jobs based on results
- Schedule and split jobs
- Update uncracked hash file
- Create new dictionaries
- Re-run jobs against new dictionaries
Cryptohaze Multiforcer

- Masking attacks
- No dictionary attacks
- Distributed computing support
- 2.0 B/s for MD5
  - 25% faster than Hashcat-plus
Cryptohaze Multiforcer

```
<table>
<thead>
<tr>
<th>'p' to pause</th>
<th>SHA1</th>
<th>Passwords Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hash type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current PW len</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Total hashes</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Cracked hashes</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Total time</td>
<td>00:21:57</td>
<td></td>
</tr>
<tr>
<td>WUs: 434/16260 (2.7%)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4: 54.242.37.118
Td 1: CID 23807.
Td 0: CID 56401.
Starting pw len 7
Waiting for workunits...
Waiting for workunits...
Td 1: out of WU.
Td 0: out of WU.
3: 184.72.133.141
2: 107.20.8.127
Td 1: CID 9158.

0: GPU: 422.68M/s
1: GPU: 422.68M/s
2: NET: 845.85M/s
3: NET: 842.57M/s
4: NET: 842.47M/s

TOTAL: 3.38B/s
```
Cryptohaze Multiforcer syntax

New-Multiforcer -h MD5 \n-c /root/Cryptohaze-Build-Latest/charsets/charsetuppernumeric \n-f /crack/_hashes/hashes_eharmony_first_1000000.txt \n--min=5 --max=5 --nocpu --noopencl \n-o /crack/output/cryptohaze_out__3__4_found.txt

Can also specify a different character set for each character position
Building dictionaries

- Dictionary files
- Word lists
- Any dataset of strings
Building dictionaries

- But there’s something missing
Passphrases > p@$$w0rds

- **Tr0ub4dor & 3**
  - Caps?
  - Common Substitutions
  - Numerical
  - Punctuation
- **correct horse battery staple**
  - Four random common words

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**: $2^{29} = 3$ days at 1000 guesses/sec
  - Difficulty to guess: **Easy**
  - Difficulty to remember: **Hard**

- **44 bits of entropy**: $2^{44} = 550$ years at 1000 guesses/sec
  - Difficulty to guess: **Hard**
  - Difficulty to remember: **You've already memorized it**

Was it trombone? No, troubador. And one of the Os was a zero? And there was some symbol...
Passphrases > p@$$w0rds

Participants who were instructed to create passwords that were at least 16 characters long created codes that were considerably stronger and easier to remember than those created by participants told to make complex eight-character passwords.

Study by CUPS (Carnegie Mellon University's CyLab Usable Privacy and Security Laboratory) http://bit.ly/PPPc3t
Passphrases > p@$@$w0rds
Passphrases in action

15 – 6,810
16 – 4,799
17 – 3,043
...
28 – 106
Passphrases

- Sentences
- Word combinations
- Mnemonics (acronyms)
- Transformations similar to password construction
Passphrases

How can we find passphrases to build a dictionary?

- Crowdsource
- Beg for orgs to share them
Passphrases

Where can we find passphrases to build a dictionary?

- E-books
- Movie scripts
- Song lyrics
- Tweets
Passphrase builder

- Dictator – instructs on what files to get
- Miner – acquires files
- Hasher – hashes for uniqueness
- Hoarder – adds to queue
Passphrase builder

- Grabber – pulls file from queue
- Converter – converts to plaintext
- Massager – converts to lower
- Splitter 1 – splits by sentence
- Splitter 2 – splits by word
- Parser – generates strings and acronyms
- Recorder – adds to DB
Passphrase builder

- Generator – sort, create acronyms, create output
Passphrase builder

- A person who never made a mistake never tried anything new.
- pwnmamntan
- a person who never
- person who never
- person who never made

- Ranking
  - Search engine results
  - Frequency in DB
  - Matches against leaks
Passphrase builder - problems

- Need passphrases to test against
- Automated search engine search limitations
- Frequency adjustments
  - Headers, footers, common strings
- Splitting inconsistencies
Takeaways

1. Can EC2 provide me value?
2. How can I crack passwords?
3. Buy or rent?
4. Passphrases or passwords?
5. What hashing algo should I use?
Q&A

Steve Werby
steve@werby.com
@stevewerby