Credential Breaches
Some Breached Companies

- LinkedIn
- Ashley Madison
- Sony
- Adobe
- Gawker
- 000webhost.com
- Yahoo!
- Stratfor
Data-Driven Statistical Attacks

- (2009) 32 million passwords: rockyou

- (2016) 117 million passwords: LinkedIn

- (2017) 3 billion passwords: Yahoo!

- Total: >10 billion passwords stolen from >500 services
Have I Been Pwned (HIBP)

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pwned websites</td>
<td>511</td>
</tr>
<tr>
<td>Pwned accounts</td>
<td>10,599,375,985</td>
</tr>
<tr>
<td>Pastes</td>
<td>113,991</td>
</tr>
<tr>
<td>Paste accounts</td>
<td>199,574,641</td>
</tr>
</tbody>
</table>

**Largest breaches**

- 772,904,991 Collection #1 accounts
- 763,117,241 Verifications.io accounts
- 711,477,622 Onliner Spambot accounts
- 622,161,052 Data Enrichment Exposure From PDL Customer accounts
- 593,427,119 Exploit.In accounts
- 457,962,538 Anti Public Combo List accounts
- 393,430,309 River City Media Spam List accounts
- 359,420,698 MySpace accounts
- 268,765,495 Wattpad accounts
- 234,842,089 NetEase accounts

**Recently added breaches**

- 645,786 Filmai.in accounts
- 10,585 NurseryCam accounts
- 358,822 People's Energy accounts
- 1,436,435 NetGalley accounts
- 110,156 CityBee accounts
- 2,481,121 Ge.tt accounts
- 1,047,200 StoryBird accounts
- 1,906,808 Pixlr accounts
- 1,422,717 MeetMindful accounts
- 2,811,929 Bonobos accounts
Password Cracking

Statistical Metrics For Passwords

• Traditionally: Shannon entropy
• Recently: $\alpha$-guesswork
• Disadvantages of statistical approaches
  – Entropy does not consider human tendencies
  – Usually no per-password estimates
  – Huge sample required
  – Not real-world attacks
Parameterized Guessability

- How many guesses a particular cracking algorithm with particular training data would take to guess a password
Attack Model

80d561388725fa74f2d03cd16e1d687c
1. $h(“123456”) = e10adc3949ba59abbe56e057f20f883e$
Attack Model

1. $h(“123456”) = e10adc3949ba59abbe56e057f20f883e$
2. $h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99$
Attack Model

1. $h(“123456”) = e10adc3949ba59abbe56e057f20f883e$
2. $h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99$
3. $h(“monkey”) = d0763edaa9d9bd2a9516280e9044d885$
Attack Model

1. \( h(“123456”) = e10adc3949ba59abbe56e057f20f883e \)
2. \( h(“password”) = 5f4dcc3b5aa765d61d8327deb882cf99 \)
3. \( h(“monkey”) = d0763edaa9d9bd2a9516280e9044d885 \)
4. \( h(“letmein”) = 0d107d09f5bbe40cade3de5c71e9e9b7 \)

\( 80d561388725fa74f2d03cd16e1d687c \)
1. \( h(“123456”) = e10adc3949ba59abbe56e057f20f883e \)
2. \( h(“password”) = 5f4dcc3b5aa765d61d817423c042e0e2c8 \)
3. \( h(“monkey”) = d0763edaa9d9bd2a9516280e9044d885 \)
4. \( h(“letmein”) = 0d107d09f5bbe40cade3de5c71e9e9b7 \)
5. \( h(“p@ssw0rd”) = 0f359740bd1cda994f8b55330c86d845 \)
1. $h(\text{"123456"}) = e10adc3949ba59abbe56e057f20f883e$
2. $h(\text{"password"}) = 5f4dcc3b5aa765d61d8327deb882cf99$
3. $h(\text{"monkey"}) = d0763edaa9d9bd2a9516280e9044d885$
4. $h(\text{"letmein"}) = 0d107d09f5bbe40cade3de5c71e9e9b7$
5. $h(\text{"p@ssw0rd"}) = 0f359740bd1cda994f8b55330c86d845$
6. $h(\text{"Chic4go"}) = 80d561388725fa74f2d03cd16e1d687c$
Guess # 6

Chic4go
jamesb0nd007!

Guess # 366,163,847,194
n(c$JZX!zKc^bIA^X^N

Guess # past cutoff
Broad Cracking Approaches

- Brute force (or selective brute force)
- Wordlist
- Mangled wordlist
  - Hashcat and John the Ripper
- Markov models
- Probabilistic Context-Free Grammar
- Neural networks
- In practice: manual, iterative updates
Mangled Wordlist Attack

Wordlist

Super
Password
Chicago
Mangled Wordlist Attack

<table>
<thead>
<tr>
<th>Wordlist</th>
<th>Rulelist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super</td>
<td>1. Append “1”</td>
</tr>
<tr>
<td>Password</td>
<td>2. Replace “a” → “4”</td>
</tr>
<tr>
<td>Chicago</td>
<td>3. Lowercase all</td>
</tr>
</tbody>
</table>
Mangled Wordlist Attack

Wordlist
- Super
- Password
- Chicago

Rulelist
1. Append “1”
2. Replace “a” → “4”
3. Lowercase all

Guesses
- Super1
Mangled Wordlist Attack

Wordlist
- Super
- Password
- Chicago

Rulelist
1. Append “1”
2. Replace “a” → “4”
3. Lowercase all

Guesses
- Super1
- Password1
Mangled Wordlist Attack

Wordlist
- Super
- Password
- Chicago

Rulelist
1. Append “1”
2. Replace “a” → “4”
3. Lowercase all

Guesses
- Super1
- Password1
- Chicago1
Mangled Wordlist Attack

Wordlist
- Super
- Password
- Chicago

Rulelist
1. Append “1”
2. Replace “a” → “4”
3. Lowercase all

Guesses
- Super1
- Password1
- Chicago1
- Super
- P4ssword
- Chic4go
# Mangled Wordlist Attack

<table>
<thead>
<tr>
<th>Wordlist</th>
<th>Rulelist</th>
<th>Guesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super</td>
<td>1. Append “1”</td>
<td>Super1</td>
</tr>
<tr>
<td>Password</td>
<td>2. Replace “a” → “4”</td>
<td>Password1</td>
</tr>
<tr>
<td>Chicago</td>
<td>3. Lowercase all</td>
<td>Chicago1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Super</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Super</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P4ssword</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chic4go</td>
</tr>
<tr>
<td></td>
<td></td>
<td>super</td>
</tr>
<tr>
<td></td>
<td></td>
<td>password</td>
</tr>
<tr>
<td></td>
<td></td>
<td>chicago</td>
</tr>
</tbody>
</table>
Example Wordlists and Rulelists

Wordlist

PGS (≈ 20,000,000)
Linkedin (≈ 60,000,000)
HIBP (≈ 500,000,000)
Example Wordlists and Rulelists

Wordlist

- PGS (≈ 20,000,000)
- Linkedin (≈ 60,000,000)
- HIBP (≈ 500,000,000)

Rulelist

- Korelogic (≈ 5,000)
- Megatron (≈ 15,000)
- Generated2 (≈ 65,000)
Example Wordlists and Rulelists

**Wordlist**
- PGS ($\approx 20,000,000$)
- Linkedin ($\approx 60,000,000$)
- HIBP ($\approx 500,000,000$)

**Rulelist**
- Korelogic ($\approx 5,000$)
- Megatron ($\approx 15,000$)
- Generated2 ($\approx 65,000$)

$10^9 \text{ - } 10^{15}$ guesses
Example Wordlists and Rulelists

<table>
<thead>
<tr>
<th>Wordlist</th>
<th>Rulelist</th>
</tr>
</thead>
<tbody>
<tr>
<td>PGS (≈ 20,000,000)</td>
<td>Korelogic (≈ 5,000)</td>
</tr>
<tr>
<td>Linkedin (≈ 60,000,000)</td>
<td>Megatron (≈ 15,000)</td>
</tr>
<tr>
<td>HIBP (≈ 500,000,000)</td>
<td>Generated2 (≈ 65,000)</td>
</tr>
</tbody>
</table>

+ Hackers’ private word/rule lists

$10^9 - 10^{15}$ guesses
John the Ripper

- Wordlist mode requires:
  - Wordlist (passwords and dictionary entries)
  - Mangling rules
- Guesses variants of input wordlist
- Speed: Fast
- “JTR”
John the Ripper

- wordlist
- rules

→

- guesses
John the Ripper

usenix
security

wordlist
rules

guesses
John the Ripper

\[ \text{usenix} \]
\[ \text{security} \]

\[\text{wordlist}\]
\[\text{[add 1 at end]}\]
\[\text{[change e to 3]}\]

\[\text{guesses}\]
John the Ripper

usenix

security

[add 1 at end]

[change e to 3]

usenix

security

rules

wordlist

usenix

security

usenix1

security1

us3nix

s3curity

guesses
John the Ripper

- wordlist:
  - use
  - nix
  - security

- rules:
  - [add 1 at end]
  - [change e to 3]

- guesses:
  - use
  - nix
  - security
  - use
  - nix1
  - security1
  - us3nix
  - s3curity
John the Ripper

**Users**

usenix
security

**Wordlist**

[ ]
[add 1 at end]
[change e to 3]

**Rules**

usenix
security

**Guesses**

usenix
security
usenix1
security1
us3nix
s3curity
Hashcat

• Wordlist mode requires:
  – Wordlist (passwords and dictionary entries)
  – Mangling rules
• Guesses variants of input wordlist
• Speed: Fast
Hashcat

wordlist

rules

guesses
Hashcat

usenix
security

[ ]
[add 1 at end]
[change e to 3]

wordlist

rules

guesses
Hashcat

\[ \text{usenix} \]
\[ \text{security} \]

[ ]
[add 1 at end]
[change e to 3]

wordlist

rules

guesses

\[ \text{usenix} \]
\[ \text{usenix1} \]
\[ \text{us3nix} \]

security

security1

s3curity
Hashcat

usenix

security

[ ]
[add 1 at end]
[change e to 3]

wordlist

rules

guesses

usenix
usenix1
us3nix
security
security1
s3curity
# Hashcat: Rule Language

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Description</th>
<th>Example Rule</th>
<th>Input Word</th>
<th>Output Word</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothing</td>
<td></td>
<td>do nothing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower-case</td>
<td>i</td>
<td>Lowercase all letters</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper-case</td>
<td>u</td>
<td>Uppercase all letters</td>
<td>P@ssWORD</td>
<td>P@ssWORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capitalize</td>
<td>c</td>
<td>Capitalize the first letter and lower the rest</td>
<td>p@ssWOrd</td>
<td>P@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Invert</td>
<td>c</td>
<td>Lowercase first found character, uppercase the rest</td>
<td>P@ssWORD</td>
<td>P@ssWORD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toggle Case</td>
<td>t</td>
<td>Toggle the case of all characters in word</td>
<td>p@ssWOrd</td>
<td>P@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toggle TN</td>
<td>T</td>
<td>Toggle the case of characters at position N</td>
<td>P@ssWOrd</td>
<td>P@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reverse</td>
<td>r</td>
<td>Reverse the entire word</td>
<td>p@ssWOrd</td>
<td>dr0Wssgjdp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicate</td>
<td>d</td>
<td>Duplicate entire word</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duplicate N</td>
<td>pN</td>
<td>Append duplicated word N times</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reflect</td>
<td>f</td>
<td>Duplicate word reversed</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotate L</td>
<td>l</td>
<td>Rotates the word left</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotate R</td>
<td>r</td>
<td>Rotates the word right</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Append char</td>
<td>X</td>
<td>Append character X to end</td>
<td>s1</td>
<td>P@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prepend char</td>
<td>X</td>
<td>Prepend character X to front</td>
<td>s1</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Truncate L</td>
<td>l</td>
<td>Deletes first character</td>
<td>l</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Truncate R</td>
<td>r</td>
<td>Deletes last character</td>
<td>r</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete N</td>
<td>D</td>
<td>Deletes character at position N</td>
<td>D</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extract range</td>
<td>xNM</td>
<td>Extracts M characters, starting at position N</td>
<td>x04</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete range</td>
<td>O</td>
<td>Deletes M characters, starting at position N</td>
<td>O12</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert char</td>
<td>i</td>
<td>Inserts character X at position N</td>
<td>i1</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overwrite</td>
<td>oX</td>
<td>Overwrites character at position N with X</td>
<td>o33</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Truncate N</td>
<td>T</td>
<td>Truncate word at position N</td>
<td>T</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replace</td>
<td>sXY</td>
<td>Replace all instances of X with Y</td>
<td>s53</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purge</td>
<td>pX</td>
<td>Purge all instances of X</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Functions

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Description</th>
<th>Example Rule</th>
<th>Input Word</th>
<th>Output Word</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap front</td>
<td>k</td>
<td>Swaps first two characters</td>
<td>k</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Swap back</td>
<td>K</td>
<td>Swaps last two characters</td>
<td>K</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Swap @ N</td>
<td>*NM</td>
<td>Swaps character at position N with character at position M</td>
<td>*34</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Bitwise shift left</td>
<td>LN</td>
<td>Bitwise shift left character @ N</td>
<td>L2</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Bitwise shift right</td>
<td>RN</td>
<td>Bitwise shift right character @ N</td>
<td>R2</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Ascii increment</td>
<td>+N</td>
<td>Increment character @ N by 1 ascii value</td>
<td>+2</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Ascii decrement</td>
<td>-N</td>
<td>Decrement character @ N by 1 ascii value</td>
<td>-1</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Replace N + 1</td>
<td>N</td>
<td>Replaces character @ N with value at @ N plus 1</td>
<td>.1</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Replace N – 1</td>
<td>N</td>
<td>Replaces character @ N with value at @ N minus 1</td>
<td>.1</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Duplicate block front</td>
<td>yN</td>
<td>Duplicates first N characters</td>
<td>y2</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Duplicate block back</td>
<td>yN</td>
<td>Duplicates last N characters</td>
<td>y2</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Title</td>
<td>E</td>
<td>Lower case the whole line, then upper case the first letter and every letter after a space</td>
<td>E</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
<tr>
<td>Title w/separator</td>
<td>eX</td>
<td>Lower case the whole line, then upper case the first letter and every letter after a custom separator character</td>
<td>e-</td>
<td>p@ssWOrd</td>
<td>p@ssWOrd</td>
<td></td>
</tr>
</tbody>
</table>

## Example Rules

* = e.g. palindrome
Hashcat: Rule Language

*05 003 d '7

Switch the first and the sixth char;
Delete the first three chars;
Duplicate the whole word;
Truncate the word to length 7;
Hashcat (Other Modes)

- Mask attack (brute force within a specified character-class structure)
- Combinator attacks
- Hybrid attacks
- Many more!
Markov Models

- Predicts future characters from previous
- Approach requires weighted data:
  - Passwords
  - Dictionaries
- Speed: Slow
- Smoothing is critical
2-gram model (1 character of context):
[start] $\Rightarrow$ c (1.0)
4 $\Rightarrow$ g (1.0)
c $\Rightarrow$ h (0.5), 4 (0.5)
g $\Rightarrow$ o (1.0)
h $\Rightarrow$ i (1.0)
i $\Rightarrow$ c (1.0)
o $\Rightarrow$ o (0.67) [end] (0.33)
Probabilistic Context-Free Grammar

- Generate password grammar
  - Structures
  - Terminals
- Speed: Slow
- "PCFG"
PCFG

password
password
password123
usenix3
5ecurity
iloveyou
nirvanaa123
PCFG

Structure Model:
\[ L_{16} \quad (1/6) \]
\[ L_8 D_3 \quad (2/6) \]
\[ L_6 D_1 \quad (1/6) \]
\[ D_1 L_7 \quad (1/6) \]
\[ L_8 \quad (1/6) \]
PCFG

password

password

usenix

security

iloveyou

nirvanaa

Digit Model:
\[ D_1 \rightarrow 3 \ (0.5) \ 5 \ (0.5) \]
\[ D_3 \rightarrow 123 \ (1.0) \]

Repeat for letters, etc.
Professionals ("Pros")

• Proprietary wordlists and configurations
  – $10^{14}$ guesses
  – Manually tuned, updated
• For example: KoreLogic
  – Password audits for Fortune 500 companies
  – Run DEF CON “Crack Me If You Can”
Approach

4 password sets

password
iloveyou
teamo123
...

password
iloveyou
1QaZ2W@x
...

Pa$$w0rd
iLov3you!
1QaZ2W@x
...

pa$$word1234
12345678asDF
!q1q!q1q!q1q
...

× 5 approaches

hashcat
advanced password recovery

John the Ripper

KoreLogic
SECURITY
Configuration Is Crucial

LongComplex

Percent guessed vs. Guesses graph showing various configurations like HC-MWR-big, HC-MWR, HC-Generated2-big, HC-Generated2, HC-SpiderLabs-big, HC-SpiderLabs, HC-Best64-big, and HC-Best64.
Comparison for Complex Passwords
Per-Password Highly Impacted

Password!
Per-Password Highly Impacted

- JTR guess # 801
Per-Password Highly Impacted

• JTR guess # 801
• Not guessed in $10^{14}$ PCFG guesses
Per-Password Highly Impacted

- JTR guess # 801
- Not guessed in $10^{14}$ PCFG guesses
Neural Networks For Passwords

Better Password Scoring

- Real-time feedback
- Runs entirely client-side
- Accurately models password guessability

Recurrent Neural Networks (RNNs)

LSTM Architecture
Generating Passwords
Generating Passwords

password → 0 or maybe 0 or O or ...
Generating Passwords

Next char is:
A: 3%
B: 1%
C: 0.6%
...
O: 55%
...
Z: 0.01%
0: 20%
1: ...
Generating Passwords

Prob: 100%

Next char is:
A: 3%
B: 2%
C: 5%
...
O: 2%
...
Z: 0.2%
0: 1%
1: ...
END: 2%
Generating Passwords

```
Prob: 100%

Next char is:
A: 3%
B: 2%
C: 5%
...
O: 2%
...
Z: 0.2%
0: 1%
1: ...
END: 2%
```

Prob: 100%
Generating Passwords

“C”
Prob: 5%
Generating Passwords

<table>
<thead>
<tr>
<th>Character</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10%</td>
</tr>
<tr>
<td>B</td>
<td>1%</td>
</tr>
<tr>
<td>C</td>
<td>4%</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td>8%</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0.02%</td>
</tr>
<tr>
<td>0</td>
<td>3%</td>
</tr>
<tr>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>END</td>
<td>6%</td>
</tr>
</tbody>
</table>

“C”
Prob: 5%
Generating Passwords

“C”
Prob: 5%

Next char is:
A: 10%
B: 1%
C: 4%
...
O: 8%
...
Z: 0.02%
0: 3%
1: ...
END: 6%
Generating Passwords

“CA”
Prob: 0.5%

Next char is:
A: 3%
B: 10%
C: 7%
...
O: 1%
...
Z: 0.03%
0: 2%
1: ...
END: 12%
Generating Passwords

“CAB”
Prob: 0.05%

Next char is:
- A: 3%
- B: 10%
- C: 7%
- ...
- O: 1%
- ...
- Z: 0.03%
- 0: 2%
- 1: ...
- END: 3%
Generating Passwords

“CAB”
Prob: 0.05%

Next char is:
A: 4%
B: 3%
C: 1%
...
O: 2%
...
Z: 0.01%
0: 4%
1: ...
END: 12%
Generating Passwords

“CAB”
Prob: 0.05%

Next char is:
A: 4%
B: 3%
C: 1%
...
O: 2%
...
Z: 0.01%
0: 4%
1: ...
END: 12%
Generating Passwords

“CAB”
Prob: 0.006%
Descending Probability Order

CAB - 0.006%
CAC - 0.0042%
ADD1 - 0.002%
CODE - 0.0013%
...

Design Space

• Model size: 3mb (browser) vs. 60mb (GPU)
• Transference learning
  – Novel password-composition policies
• Training data
  – Natural language
• (Many others)
Key Results

- Neural networks produce better guesses than previous methods
- Larger model not a major advantage
- Browser implementation in Javascript
Intelligibility (Explanations)
Building a Data-Driven Meter

We designed & tested a meter with:
1) Principled strength estimates
2) Data-driven feedback to users
We designed & tested a meter with:
1) Principled strength estimates (RNN)
2) Data-driven feedback to users
We designed & tested a meter with:
1) Principled strength estimates
2) Data-driven feedback to users
Provide Intelligible Explanations

Unic0rns

Don't use simple transformations of words or phrases (unicorns $\rightarrow$ Unic0rns)

Capitalize a letter in the middle, rather than the first character

- 21 characteristics
- Weightings determined with regression
After Requirements Are Met...

Create Your Password

Username
blase

Password

Don't use dictionary words or words used on Wikipedia
Consider inserting digits into the middle
Consider making your password longer

Show Password & Detailed Feedback

Confirm Password

See Your Password With Our Improvements

How to make strong passwords

Continue
Displays Score Visually
…Provides Text Feedback

Create Your Password

Username
blase

Password
 ************

Show Password & Detailed Feedback

Confirm Password

Your password could be better.

- Don’t use dictionary words or words used on Wikipedia
- Consider inserting digits into the middle
- Consider making your password longer

See Your Password With Our Improvements

How to make strong passwords

Continue
…Gives Detail (Password Shown)

Create Your Password

Username
blase

Password
CryptoUnicorn3|

Show Password & Detailed Feedback

Confirm Password

Your password could be better.

- Don't use dictionary words (Unicorn) or words used on Wikipedia (Crypto)
- Consider inserting digits into the middle, not just at the end
- Consider making your password longer than 14 characters

A better choice: C3ryptoUnicorn@

How to make strong passwords
…Offers Explanations

Create Your Password

Username
blase

Password
CryptoUnicorn3|

Your password could be better.
- Don’t use dictionary words (Unicorn) or words used on Wikipedia (Crypto)
- Consider inserting digits into the middle, not just at the end
- Consider making your password longer than 14 characters

Show Password & Detailed Feedback

Confirm Password

A better choice: CRYPTOUnicorn@

How to make strong passwords

(Why?)
Explanations Shown in Modal

A better choice: **C3ryptoUnicorn@**

**Your password could be better.**

- Don’t use dictionary words (Unicorn) or words used on Wikipedia (Crypto)
  Attackers use software that automatically guesses millions of words commonly found in dictionaries, wordlists, or other people’s passwords

- Consider inserting digits into the middle, not just at the end
  38% of people also put digits at the end of the password

- Consider making your password longer than 14 characters
  In recent years, attackers have gotten much better at guessing passwords under 16 characters

**How to make strong passwords**
Standard Feedback

Create Your Password

Username
blase

Password

Your password could be better.

- Don’t use dictionary words (Unicorn) or words used on Wikipedia (Crypto)

A better choice: C3ryptoUniCorn@

Confirm Password

A better choice: C3ryptoUniCorn@

Continue

How to make strong passwords