Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries

Dario Pasquini, Marco Cianfriglia, Giuseppe Ateniese and Massimo Bernaschi
Password Security in a slide:

Adversary model

Guess number
(how many guesses $A$ has to perform to guess it)

$A(\text{"pa$$word!"}) = 1123$

Weak password

Strong password

Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries
Password Security in a slide:

Adversary model

\[ A(\text{"7H1d6~3. qA"}) = 10^{14} \]

Guess number
(how many guesses \(A\) has to perform to guess it)

Weak password

Strong password

Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries
Why measuring password strength?

Questions Bob must answer:
• Should I make the hash function more expensive? (e.g., increasing the work-factor)
• Should I impose a different password composition policy?
• Is this new password composition policy making users choose stronger passwords?
• etc..

Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries
Reality check:

Our model of the real-world password cracker

The real-world password cracker

Dictionary attacks with Mangling rules:

• How they work:

**Wordlist:**
- princess
- jimmy
- iloveyou
- passwd0rd
- 123467

**Rules-set:**
- \( u \)
- \( $9 \ $1 \ ] ] ] \)
- \( "princess" \otimes u \)
- \( "princess" \otimes $9 \ $1 \]
- \( "princess" \otimes ] ] ] \)

(Mangling rule)

(Guess)

- \( = "PRINCESS" \)
- \( = "princess91" \)
- \( = "princ" \)
Dictionary attacks with Mangling rules:

• How they work:

**Wordlist:**
- princess
- jimmy
- iloveyou
- passw0rd
- 123467
- ...

**Rules-set:**
- $u$
- "$9 \$1$
- "[ ] [ ]"
- "\(\text{"princess" } \otimes \)"
- "[ ] [ ]"
- "\(\text{"princess" } \otimes \)"

(Mangling rule)

(Guess)

\(\text{"princess91" } = \text{"PRINCESS"}\)
\(\text{"princ" } = \text{"PRINCESS"}\)

• Why Dictionary attacks are so important:
  • Password crackers use Dictionary Attacks:
    • Dictionary Attacks are fast (offline attacks are also about time)
    • Dictionary Attacks are flexible (every target is different)
A real Dictionary Attack:

- Real-world password crackers:
  1. Use **optimized setups**: 
     - Carefully chosen Word-lists
     - Carefully crafted Rules-sets
  2. **Manually tailor** the setup for the very **target**:
     - Collect information on the target before the attack
     - Tune in consideration of the initial passwords guessed during the attack
A real Dictionary Attack:

• Real-world password crackers:
  1. Use **optimized setups:**
     • Carefully chosen Word-lists
     • Carefully crafted Rules-sets
  2. **Manually tailor** the setup for the very target:
     • Collect information on the target before the attack
     • Tune in consideration of the initial passwords guessed during the attack

Manual processes that require a profound domain knowledge.

UNDERESTIMATION
Let’s fill the gap

Trying to get from this:

Attacker’s expertise

To this:

How we do it:

1. Adaptive Mangling Rules
   (simulate high-end configurations)

2. Dynamic Dictionary (hits-recycling)
   (simulates customized setups for the target)

Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries
On the functional interaction between rules-set and dictionary:

Adaptive Mangling Rules:

The idea: It is a good attack if the wordlist and the rule-set work well together
Optimizing compatibility between Wordlist and Rule-set

- Not all the rules equally interact with all the dictionary words:

```
Wordlist:
princess
zxcvbn
iloveyou
pa$$w0rd
!@#$%^  

Rules-set:
$006
$9 $1
]]]
]$o
]$a   

Guesses:
princess91
princ
princeso
princesa
Low-quality (wasted) guesses
```
Optimizing compatibility between Wordlist and Rule-set

- Not all the rules equally interact with all the dictionary words:

- In unoptimized setups, most of the guesses are generated by low-compatibility pairs (rule/word).
- Most of the generated guesses have very low-quality!
Adaptive Mangling Rules: The idea

- Every word from the dictionary gets its own rules-set:

  1. Get a dedicated rules-set:
  2. Apply the dedicated rules-set:

\[
\text{Wordlist: } \rightarrow \text{“princess”}, \quad \text{Rules-set: } \rightarrow \text{“princess”}, \quad \text{Rules-set: (for “princess”)} \rightarrow \text{“princess”} \rightarrow \text{Guesses: “Princess” “princess1” “princess2” “ssecnirp” …}
\]
Adaptive Mangling Rules: The idea

• Every word from the dictionary gets its own rules-set:

2. Apply the dedicated rules-set.

• It works at run-time! (we’ll see why later)
• Let’s model rule/word compatibility with a Deep Neural Network
  • It’s a multi-label classification task (each mangling rule is a class)
  • A rule-set is fixed (e.g., `generated2.rule`) and the network outputs a “compatibility score” for each rule in the set with a single inference.

Dictionary word: 

```

```

Compatibility scores:

```

```

```
• Let’s model rule/word compatibility with a **Deep Neural Network**
  • It’s a multi-label classification task (each mangling rule is a class)
  • A rule-set is fixed (e.g., `generated2.rule`) and the network outputs a “compatibility score” for each rule in the set with a single inference.

---

**Dictionary word:**

**Compatibility scores:**

- \( P("O26") = 0.05 \)
- \( P("*16") = 0.12 \)
- \( P("$9$9") = 0.93 \)
- \( P("06*t") = 0.01 \)
- \( P("se3\") = 0.62 \)

**Guesses:**

- “princess99”
- “princ3ss”

---

**Resnet (Deep Conv Net)**

\( \beta = 0.5 \)

Only rules with a score higher than \( \beta \) are selected and applied to the input string.

Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries
Is it Effective? An example:

Setup:
- **Rule engine**: HashCat
- **Wordlist**: MyHeritage.com
- **Rules-set**: InsidePro-PasswordsPro (3120 rules)
- **Target**: animoto.com

- It’s also fast:

  **Numbers of compatibility scores per second** *(On a NVIDIA V100)*

<table>
<thead>
<tr>
<th><code>generated2.rule</code></th>
<th><code>generated.rule</code></th>
<th><code>PasswordPro.rule</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>#rules: 64528</td>
<td>#rules: 14424</td>
<td>#rules: 3120</td>
</tr>
<tr>
<td>1.528.346.738 c/s</td>
<td>611.793.885 c/s</td>
<td>164.854.041 c/s</td>
</tr>
</tbody>
</table>

  Intrinsic parallel + It can run on tensor cores
  (totally idle during the hash computation on GPU).

  **The effect of $\beta$:**

  ![Effect of $\beta$ graph]

  **Performance gain:**
  ~90 – 80%
Hits-Recycling: Simple, yet effective

- Passwords created under the same environment (e.g., webapp) are not independent from each other.
- Therefore, the passwords you guessed during the attack can easily bring you to guess new related passwords that occur in the target.
Hits-Recycling: Simple, yet effective

- Passwords created under the same environment (e.g., webapp) are not independent from each other.
- Therefore, the passwords you guessed during the attack can easily bring you to guess new related passwords that occur in the target.
- **Hits-Recycling**: The wordlist is dynamically augmented with the passwords guessed during the running attack.

---

**Attacked set X**: fc8ef33f9bed6cbbc11d300c15adac2a cb1acd2ba854d4a5eefc5a00529ec97b 567abc1411789d3ace45c6c7d7356586 7a7b7661f7c73ddbb8da953af74c1a 628f6eb3c63d76857ce96412b5e8368 fd5009773509a6690f8693040feac50 7fa15f0c276e0950cb44ffca0ba1 9a94ebd82479a357c98f0588b29f47b 9b6f3083403b2499619f2451c683a695 B687608b7301d4141b417fad277cc20d

**Wordlist:**
- princess
- zxcvbn
- iloveyou
- ...

**Rules-set:**
- apply
- u
- $9$
- $1$
- ...

**PRINCESS**
- princess91

**MISS**
- HIT
**Hits-Recycling: Simple, yet effective**

- Passwords created under the same environment (e.g., webapp) are not independent from each other.
- Therefore, the passwords you guessed during the attack can easily bring you to guess new related passwords that occur in the target.
- **Hits-Recycling:** The wordlist is dynamically augmented with the passwords guessed during the running attack.

---

**Attacked set $X$:**

```
fc8ef33f9bed6cbbc11d300c15adca2a
cb1acd2ba854d4a5eef5a00529ec97b
567abc1411789d3ace45c677d7356586
7a7b76619f7c3d3d8a9d53af74c1a
628f9e6367e6857ce96412b5e8368
fd5009773509ad6690f8693040feac50
7fa15f0c276e0f95dd0cb44ff9dca0b1
9a94ed82479a357c98f80588b29f47b
9b6f0383403b2499619f2451c68a695
B687608b7301d4141b417fad277cc20d
```
Hits-Recycling: the Effect

Setup:
- **Rule engine**: HashCat
- **Rules-set**: InsidePro-PASSWORDSPro (3120 rules)
Hits-Recycling: the Effect

Reducing Bias in Modeling Real-world Password Strength via Deep Learning and Dynamic Dictionaries

**Setup:**
- **Rule engine:** HashCat
- **Rules-set:** InsidePro-PasswordsPro (3120 rules)

Reduces performance variance.
Hits-Recycling: the Effect

**Setup:**
- **Rule engine:** HashCat
- **Rules-set:** InsidePro-PasswordsPro (3120 rules)
Fixing Hits-Recycling with the Adaptive rules

- Hits-recycling + Adaptive Rules = Adaptive, Dynamic Mangling rules \((ADA\text{Ms})\).
- Adaptive mangling rules offers a run-time optimization for the dynamic created dictionary.

**\(ADA\text{Ms} \text{ vs Standard}** attack with same configuration (rules-set/dictionary):

**Rules-set**: \textit{generated2} (64528 rules)
Conclusion:

The proposed techniques:

• Make dictionary attacks more resilient to misconfigurations.
• Get closer to real-world guessing strategies.

• The code and trained models are available.

Use it instead of plain dictionary attacks in our security analysis!

https://github.com/TheAdamProject/adams
Thanks for your Attention!

For any questions: pasquini@di.uniroma1.it

We are working on a full-fledged (GPU-based) implementation with additional modules:

Adam, The First Cracker