The **TypTop** System

Personalized Typo-Tolerant Password Checking

R. Chatterjee, J. Woodage, Y. Pnueli, A. Chowdhury, T. Ristenpart
Password checking systems and typos

Salted, slow cryptographic hash

Bob

- passw92
- Success
- small typo
- Passw92
- Fail

Salted, slow cryptographic hash

H(passw92) = a5idoiaU7p...
H(Passw92) ≠ a5idoiaU7p...
H(pASSW92) ≠ a5idoiaU7p...
H(passw92) = a5idoiaU7p...

Apply Caps lock Corrector
Apply shift Corrector

Top-5 correctors correct 20% of all typos

Typo-tolerant password checking
Allow registered password or typos of it

Oakland ’16

pASSWORD tYPOS and How to Correct Them Securely

Rahul Chatterjee*, Anish Athalye†, Devdatta Akhawe‡, Ari Juels*, Thomas Ristenpart*
* Cornell Tech, † MIT, ‡ Dropbox

Abstract—We provide the first treatment of typo-tolerant password authentication for arbitrary user-selected passwords. Such a system, rather than simply rejecting a login attempt with an incorrect password, tries to correct common typographical typos made by users. We perform preliminary experiments with Amazon Mechanical Turk (MTurk) in which we task human workers with transcribing passwords drawn from the RockYou password leak. This does not perfectly model user...
Typo-tolerance improves utility

Login will increase by 3%

Does not degrade security

Salted, slow cryptographic hash

H(passw92) ≠ a5idoiaU7p...
H(Passw92) ≠ a5idoiaU7p...
H(passw92) = a5idoiaU7p...
H(Passw92) = a5idoiaU7p...

Apply Caps lock Corrector

Apply shift Corrector

Top-5 correctors correct 20% of all typos
... corrects only the tip of the iceberg

Limitations

To correct more with correctors would be
1. Expensive – slow hash function
2. Wasteful – not all users make same mistakes
3. Insecure – too many corrections for each guess

80% of typos are left uncorrected

Top-5 correctors correct 20% of all typos

How to correct more typos?
We propose: Personalized typo-tolerance

• Introduce personalized typo-tolerant password checking: allow only the typos that a user makes

• Design TypTop, a password checker that learns user’s frequent typos and allows login with them. Rigorously analyze TypTop’s security.

• Build a prototype for rendering computer logins typo-tolerant

  https://typtop.info
Adaptive typo-tolerance

2. How to figure out this is a legitimate typo? E.g., within edit-distance 1 from real password

Allow previously seen typos

1. Do users repeat their typos?

If only we could store passwords in plaintext...
Do users repeat their typos?
Simulate password typing behavior at

• Asked workers
  • to register a password for an imaginary email service
  • and then, login by typing the password over multiple days

271 workers logged in for 8,739 times, median 30 times

35% made at least two typos in two different logins

50% more users will benefit compared to prior approach

45% of them repeat their typos
Is it legitimate?

$H(\text{passe92}) \neq \{\text{a5idoiaU7p...}\}$

How to build a secure adaptive typo-tolerant password checking?
Design of TypTop: Registration

\[ \text{Register} \quad \text{passw92} \]

\((E, D)\): password-based encryption scheme

\[ E(\text{passw92}, sk) : \]

\[ k \leftarrow \text{PBKDF} (\text{passw92}) \]

\[ C \leftarrow AEnc (k, sk) \]

return \(C\)

\[ \begin{array}{c}
\text{Cache} \\
E(\text{passw92}, sk) \\
\end{array} \]

\[ \begin{array}{c}
\text{Waitlist} \\
\end{array} \]

\[ \begin{array}{c}
\text{Aux Info} \\
\text{typTop’s state} \\
\text{pk} \\
\end{array} \]

Generates a public-key key pair: \(sk, pk\)

\[ \text{Slow, secure, authenticated encryption scheme} \]
Design of TypTop: Login

$p_k \not\equiv \bot$

\[ E(\text{passw92}, sk) \]

\[ \text{D(} \text{passw92,} \) \neq \bot \]
Design of TypTop: Login

-passw92

Fail

-passw92

Success

\[ E(\text{passw92}, sk) \]

\[ E(\text{passe92}, sk) \]

\[ D(\text{passw92}, sk) \]

\[ \text{passe92} \]

\[ pk \]

\[ pk \]

\[ sk \]

Typo Policy Check
Design of TypTop: Login with a typo

Adaptive typo-tolerant password checking without storing the password or any typos in clear
Design of TypTop: Some more details

- Cache:
  - `E(`passw92, sk`)`
  - `E(`passe92, sk`)`
  - `#*D&abττητν`
  - Size: 5+1

- Waitlist:
  - `ラβτU823`
  - `了んど0隨`
  - `تَخَلَشَتْشَك`
  - Size: 10

- Aux Info:
  - `cpk`

- Frequency of typos, etc.

- Randomly permute after every cache update; benefits security

- Fill cache and waitlist with random values to hide # of entries

- Check edit-distance + typo strength
  - Used `zxcvbn` strength meter

- Typo Policy Check
What about Security?
Smash and grab attack (Offline attack)

More interesting, and we detail this in the talk

Remote guessing attack (Online attack)

- Analysis is similar to Oakland ’16 paper
- Showed negligible security loss
- Please see paper for details
Smash and grab attack (Offline attack)

**Attacker’s Goal**
Learn the registered password

just like attacking traditional password checkers

**Obvious Strategy**
Brute-force guess the password

Can the attacker do better?
No!

Cache
\[ E(\text{passw92}, sk) \]
\[ #*\&aβτṛēm \]
\[ E(\text{passe92}, sk) \]

Waitlist
\[ \text{Passw92}_{pk} \]
\[ \text{passe92}_{pk} \]

Aux Info
\[ \text{TypTop’s state} \]
\[ pk \]
Obvious strategy is the best an attacker can do.
TypTop’s state appears random

\[
\text{Cache} \quad \text{Waitlist} \quad \text{Aux Info}
\]

[Chinese characters]

\[
\text{E(passw92, sk)} \quad \#*D&abCtryem
\]

[Chinese characters]

\[
\text{E(passe92, sk)} \quad \text{Assuming underlying encryption schemes are secure}
\]

⇒ Attacker learns nothing unless he can guess an entry in the cache
Guessing against the cache entries

Decryption is as slow as normal password hashes

Typo entries are randomly permuted

Can attacker ever get higher advantage by trying to decrypt a typo entry in the cache?
Guessing typo is beneficial if...

...there is a typo that is always in the cache, the attacker can break TypTop by guessing that typo against all slots.

That scenario is quite unnatural
t-Sparse

t-sparse: if no typo is frequently in the cache of many passwords

\[ \forall \tilde{w}, \sum_{w} \tilde{\tau}_w(\tilde{w}) \leq t \]

Cache inclusion probability \((\tilde{\tau}_w)\)

\[ \tilde{\tau}_w(\tilde{w}) = \Pr[\tilde{w} \text{ in cache} \mid w], \]

Depends on the typo-distribution, and TypTop’s caching policy

\(w\) : Password
\(\tilde{w}\) : Typo
\(\tilde{\tau}_w\) : Cache inclusion probability
\(t\) : # of typos allowed in cache
t-Sparse ⇒ TypTop ≡ Normal Pw checker

Theorem

If typo-distribution is t-sparse under TypTop’s caching policy, then best attack is to brute-force guess the registered password.
Guessing typo is sub-optimal if t-sparse

Every guess against a typo can be replaced by a guess against the real password that provides equal or more probability of success
Empirically verified that real world typo-distributions are **t-sparse** for the configurations we considered for TypTop.

Attacking TypTop is no easier than attacking traditional password checkers.
TypTop is secure against online and offline attacks, and it improves utility.

Let’s build one!
TypTop: a smart password checker for Unix

• Created a password authentication module (PAM)
• Renders computer logins typo-tolerant
• Added a logging module
  • To collect anonymous statistics about typos for our study
  • Users can disable logging, and still keep using TypTop

TypTop
A smart password checker that lets you make mistakes

https://typtop.info
TypTop pilot deployment study

• Installed TypTop in 24 volunteers’ laptops
  • 5 on Linux platform, 19 on MAC
  • for median 27 days.
  • Total typos observed: 501

TypTop provides 3x improvement over prior approach

Fraction of typo corrected

<table>
<thead>
<tr>
<th></th>
<th>Prior Approach</th>
<th>TypTop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraction of typo</td>
<td>22%</td>
<td>63%</td>
</tr>
</tbody>
</table>
TypTop in one slide

• Designed TypTop, a secure personalized typo-tolerant password checking system, that adapts to user’s mistakes

• Rigorously analyzed its security

• You can try TypTop now! Visit https://typtop.info

Typo-tolerant password checking might encourage users to adopt better security practices