How to End Password Reuse On the Web

Michael K. Reiter
Lawrence M. Slifkin Distinguished Professor
University of North Carolina at Chapel Hill

Joint work with Ke Coby Wang.

Password Reuse :: Definition

same user,
same or similar password,
multiple websites.
Leaked Passwords

Announced data breaches in the past four months:

- Photography site 500px resets 14.8 million passwords after data breach
- Coinmama suffers a data breach of 450,000 emails and hashed passwords
- Houzz discloses data breach, asks some users to reset passwords

Password Reuse :: Threats

“Credential stuffing”

Valid user ID password pairs* via database breaches, phishing, malware, social engineering, etc.
Password Reuse :: Harm

Web service providers:
- Stolen credential ransom
- Costs on preventing & detecting account takeovers
- Customer churn

Web service users:
- Financial loss
- Privacy violations

Leaked passwords → Credential stuffing → Account takeovers

--- Alex Stamos (former CSO, Facebook)
The reuse of passwords is the No. 1 cause of harm on the internet.

--- Patrick Heim (Head of Trust & Security, Dropbox)
99% of compromised user accounts come from password reuse.

--- Troy Hunt (Regional Director, Microsoft)
Credential stuffing is enormously effective due to the password reuse problem.

* via database breaches, phishing, malware, social engineering, etc.
Existing Works

**Pre-attack**
- Detecting & cross-checking leaked passwords
- Leaked passwords
- Password reuse

**Attack**
- Credential stuffing

**Post-attack**
- Account takeovers
- Honey accounts, account activity monitoring, etc.
- Two-factor, multi-factor authentication, etc.
Existing Works

Pre-attack

Detecting & cross-checking leaked passwords

Leaked passwords

Credential stuffing

Password reuse

Attack

Post-attack

Account takeovers

Two-factor, multi-factor authentication, etc.

Detecting password reuse on the client side

Honey accounts, account activity monitoring, etc.
Our Work

Pre-attack

Detecting & cross-checking leaked passwords

Leaked passwords

Password reuse

Our work: server side

Attack

Credential stuffing

Post-attack

Account takeovers

Two-factor, multi-factor authentication, etc.

Honey accounts, account activity monitoring, etc.

Goals
Goals :: Functionality

(Re)set password

User
(Alice)

Requester
(Website A)
Goals :: Functionality

User (Alice) → (Re)set password → Requester (Website A)

Same/similar password exists for Alice?

Responders (Websites where Alice already has accounts)

Goals :: Functionality

User (Alice) → (Re)set password → Requester (Website A)

Accept/Reject

Same/similar password exists for Alice?

Responders (Websites where Alice already has accounts)
Goals :: Functionality :: Examples

**Same password reuse**

<table>
<thead>
<tr>
<th>User</th>
<th>Requester</th>
<th>Responder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Amazon.com</td>
<td>Websites where Alice already has accounts</td>
</tr>
<tr>
<td>UserID: <a href="mailto:alice@gmail.com">alice@gmail.com</a></td>
<td>Password: &quot;happywednesday&quot;</td>
<td>Previously registered at Dropbox.com</td>
</tr>
</tbody>
</table>

UserID: alice@gmail.com
Password: "happywednesday"
Previously registered at Dropbox.com

"happywednesday"
"gotarheels"
Same password reuse

UserID: alice@gmail.com
Password: “happywednesday”
Previously registered at Dropbox.com

User (Alice)

Requester (Amazon.com)

Sorry, please input another one.

Responder
(Websites where Alice already has accounts)
Similar password reuse

Goals :: Functionality :: Examples

Requester (Amazon.com)

User (Alice)

Responder (Websites where Alice already has accounts)

Password: "happytuesday"

Sorry, please input another one.

Password: "happytuesday"

Congrats! You can use this pwd.

Similar password set, e.g., including "happytuesday"
Goals :: Deployment

- We don’t require a universal adoption of our framework
- A simple estimate shows that if these 20 websites adopted our framework, then each Internet user would have ~4-5 different and dissimilar passwords

> “If multiple passwords cannot be avoided, four or five is the maximum for unrelated, regularly used passwords that users can be expected to cope with.”

<table>
<thead>
<tr>
<th>Website</th>
<th>Users (M)</th>
<th>Websites</th>
<th>Users (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook</td>
<td>2167</td>
<td>Taobao</td>
<td>580</td>
</tr>
<tr>
<td>YouTube</td>
<td>1500</td>
<td>Outlook</td>
<td>400</td>
</tr>
<tr>
<td>WhatsApp</td>
<td>1300</td>
<td>Sina Weibo</td>
<td>376</td>
</tr>
<tr>
<td>Yahoo</td>
<td>1000</td>
<td>Twitter</td>
<td>330</td>
</tr>
<tr>
<td>Gmail</td>
<td>1000</td>
<td>Amazon</td>
<td>310</td>
</tr>
<tr>
<td>WeChat</td>
<td>980</td>
<td>Baidu Tieba</td>
<td>300</td>
</tr>
<tr>
<td>QQ</td>
<td>843</td>
<td>LinkedIn</td>
<td>260</td>
</tr>
<tr>
<td>Instagram</td>
<td>800</td>
<td>Snapchat</td>
<td>255</td>
</tr>
<tr>
<td>Tumblr</td>
<td>794</td>
<td>Reddit</td>
<td>250</td>
</tr>
<tr>
<td>iCloud</td>
<td>782</td>
<td>Pinterest</td>
<td>200</td>
</tr>
</tbody>
</table>

Table: Top 20 websites ranked by number of active users. In addition, there are 3.58 billion active Internet users worldwide.


Goals :: Users

- Should be largely invisible to those who already use distinct passwords at distinct sites
Goals :: Users

- Should be largely invisible to those who already use distinct passwords at distinct sites
- For others, our goal is to compel them to use distinct passwords for accounts (e.g., via a password manager)

---

Goals :: Users

- Should be largely invisible to those who already use distinct passwords at distinct sites
- For others, our goal is to compel them to use distinct passwords for accounts (e.g., via a password manager)

Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another
Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another
- **Account security**: Interfere with password reuse while not qualitatively degrading account security in other ways
Goals :: Security and Privacy

- **Account location privacy**: Participating websites are not disclosed to one another
- **Account security**: Interfere with password reuse while not qualitatively degrading account security in other ways

![Diagram showing the interaction between a User, Requester, and Responders]

**User** (Alice) 

**Requester** (Website A) 

**Responders** (Websites where Alice already has accounts)

Design
Design :: Private Membership Test (PMT)

- **User** (Alice) submitted password
- **Requester** (Amazon.com)
  - Query message
  - Response message
- **Responder(s)** (Websites where Alice already has accounts)
  - Same or similar password exists?
  - Same or similar password exists?

**Membership Test: Is \( p \) in \( S \)?

- **Requester** (element, \( p \))
- **Responder** (set: \( S \))
  - \( p \) remains private.
  - \( S \) remains private.
  - Learns: Whether \( p \) is in \( S \).
  - Learns: Nothing, not even this PMT result.
Design :: PMT Building Block I

Multiplicatively Homomorphic Encryption (MHE)

\[ 3 \times \text{encryption} \rightarrow 3 \]

\[ 5 \times \text{encryption} \rightarrow 5 \]

\[ x_{pk} : \text{homomorphic multiplication (only } pk \text{ is needed)} \]

\[ pk : \text{public key (or “encryption key”)} \]

\[ sk : \text{private key (or “decryption key”)} \]
Multiplicatively Homomorphic Encryption (MHE)

\[ 3 \times 5 = 15 \]

- \( x_{pk} \): homomorphic multiplication (only \( pk \) is needed)
- \( pk \): public key (or “encryption key”)
- \( sk \): private key (or “decryption key”)

Design :: PMT Building Block I

Multiplicatively Homomorphic Encryption (MHE)

\[ 3 \] \[ 5 \] \[ 3 \times 5 = 15 \]

- \( x_{pk} \): homomorphic multiplication (only \( pk \) is needed)
- \( pk \): public key (or “encryption key”)
- \( sk \): private key (or “decryption key”)
Multiplicatively Homomorphic Encryption (MHE)

 Plaintexts are elements of a multiplicative group $G$

\[ \text{x} \times \text{pk} : \text{homomorphic multiplication (only pk is needed)} \]

\[ \text{pk} : \text{public key (or “encryption key”) } \]

\[ \text{sk} : \text{private key (or “decryption key”) } \]
Bloom Filter (BF)

- Space-efficient
- Allows adding element and testing membership in constant time
- Membership test can produce false positives

\[
\{"bigbang", "domingo"}\]

\[
\begin{array}{cccccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Bloom Filter (BF)

- Space-efficient
- Allows adding element and testing membership in constant time
- Membership test can produce false positives

{"bigbang", "domingo"}

0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 0 0

"bigbang"
Design :: PMT Building Block II

Bloom Filter (BF)

- Space-efficient
- Allows adding element and testing membership in constant time
- Membership test can produce false positives

\[
\begin{array}{c}
\text{"bigbang", "domingo"} \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}
\]

"BIGBANG"

Design :: PMT Design (I)

**Requester**
(Password: "bigbang", pk, sk)

**Responder**
(Set: $S = \{"bigbang", "BIGBANG"\})

Password candidate set $S$ generated by some algorithm (e.g., password guessers)
Design :: PMT Design (II)

Requester
(Password: “bigbang”, pk, sk)

Responser
(Set: $S$)

Password candidate set $S = \{ \text{“bigbang”}, \text{“BIGBANG”} \}$
generated by some algorithm
(e.g., password guessers)

Bloom Filter

“bigbang”

1 0 1 0 0 1 0 1

Design :: PMT Design (III)

Requester
(Password: “bigbang”, pk, sk)

Responser
(Set: $S$)

Password candidate set $S = \{ \text{“bigbang”}, \text{“BIGBANG”} \}$
generated by some algorithm
(e.g., password guessers)

$1_g$ is group identity;
$S_i$ is random group elmt

1 0 1 0 0 1 0 1

$S_1$, $S_2$, $S_3$, $S_4$, $S_5$.

# PP
Design :: PMT Design (IV)

Encryption using MHE

Requester
(Password: "bigbang", pk, sk)

Responder
(Set: S)

Password candidate set $S = \{ "bigbang", "BIGBANG" \}$ generated by some algorithm (e.g., password guessers)

Design :: PMT Design (V)

Requester
(Password: "bigbang", pk, sk)

Responder
(Set: S)

Password candidate set $S = \{ "bigbang", "BIGBANG" \}$ generated by some algorithm (e.g., password guessers)
Design :: PMT Design (VI)

Requester
(Password: "bigbang", pk, sk)

Comparer
(Set: $S$)

Password candidate set $S = \{"bigbang", "BIGBANG"\}$ generated by some algorithm (e.g., password guessers)

$C_0, C_7, pk$

Requester
(Password: "bigbang", pk, sk)

Response
Multiplicative homomorphism is used here

$r$ is a random residue mod group order

$S = \{"bigbang", "BIGBANG"\}$

From the requester

Design :: PMT Design (VII)
Design :: PMT Design (VIII)

Requester
(Password: "bigbang", pk, sk)

Responder
(Set: S)

Password candidate set $S = \{"bigbang", "BIGBANG"\}$ generated by some algorithm (e.g., password guessers)

$S = \{"bigbang", "BIGBANG"\}$

After decryption using sk:
If $1_G$: "true" (membership holds); otherwise: "false".

Design :: PMT Design (IX)

Requester
(Password: "bigbang", pk, sk)

Responder
(Set: S)

Password candidate set $S = \{"bigbang", "BIGBANG"\}$ generated by some algorithm (e.g., password guessers)

$S = \{"bigbang", "BIGBANG"\}$
Our PMT Protocol

- One round of interaction

Requester
(element, \( p \))

Responder
(set: \( S \))
Our PMT Protocol

- **One** round of interaction
- **One ciphertext per response message**

Requester: (element, \( p \))

Responder: (set: \( S \))

One ciphertext only

- **Information leakage limited to one bit against malicious parties.**
Our PMT Protocol

- **One** round of interaction
- **One** ciphertext per response message
- Information leakage limited to **one** bit against malicious parties.
  - Requester obtains up to 1 bit
  - Responder obtains up to 1 bit

**Query 1 for Alice**

- Responder’s guess: the response for Query 1 must be positive
- Up to 1 bit
Our PMT Protocol

- One round of interaction
- One ciphertext per response message
- Information leakage limited to one bit against malicious parties.
  - Requester obtains up to 1 bit
  - Responder obtains up to 1 bit
  - “probabilistic fake query”

Design :: PMT Security for the Responder

What if a malicious requester tries to learn more about the password set $S$ at the responder?

- Requester (password, $p$)
- Responder (Set: $S$)

Learned: whether $p$ is in $S$.
Security: $p$ remains private.
Design :: PMT Security for the Responder

What if a malicious requester tries to learn more about the password set $S$ at the responder?

- Provably impossible for a malicious requester to learn more about the responder’s $S$ (information-theoretic security)

<table>
<thead>
<tr>
<th>Requester (password, $p$)</th>
<th>Responder (Set: $S$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query message</td>
<td>Response message</td>
</tr>
<tr>
<td>Learned:</td>
<td>Security:</td>
</tr>
<tr>
<td>whether $p$ is in $S$.</td>
<td>$S$ remains private.</td>
</tr>
<tr>
<td>Security: $p$ remains private.</td>
<td></td>
</tr>
</tbody>
</table>

Design :: PMT Security for the Requester

What if a malicious responder tries to figure out the password $p$ at the requester?

Responder’s guess: Query 2 will show up if previous response is “false” (the user would try a new password)
Design :: PMT Security for the Requester

What if a malicious responder tries to figure out the password $p$ at the requester?

- Adopted ElGamal encryption over a multiplicative prime-order group
- Proved in the Generic Group Model that a malicious responder learns nothing more than the result of the query
- When the generic group is instantiated, this becomes a cryptographic assumption

Directory

User
(Alice)

Requester
(Website A)

Directory
(3rd party)

Responders
(Websites where Alice already has accounts)
Directory

User (Alice)  Requester (Website A)  Directory (3rd party)

<table>
<thead>
<tr>
<th>User ID</th>
<th>Responder Address</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:alice@xxx.com">alice@xxx.com</a></td>
<td>RespAddr 1</td>
</tr>
<tr>
<td></td>
<td>RespAddr 2</td>
</tr>
<tr>
<td></td>
<td>RespAddr 3</td>
</tr>
</tbody>
</table>

Responders (Websites where Alice already has accounts)

- Forwarding messages
- Not involved with password storage or any cryptographic operations for PMTs
Framework Design

When Alice tries to register a new account or to change her password at the requester:

User ID, Password

Requester
(Website A)

PMT query

Directory
(3rd party)

Responders
(Websites where Alice already has accounts)

Set: $S_r$

Set: $S_m$
Framework Design

When Alice tries to register a new account or to change her password at the requester:

User ID, Password

Requester (Website A)

PMT query

Directory (3rd party)

PMT query

PMT response

Responders (Websites where Alice already has accounts)

Set: $S_1$

Set: $S_m$
Framework Design

When Alice tries to register a new account or to change her password at the requester:

User (Alice)

Requester (Website A)

Directory (3rd party)

Responders (Websites where Alice already has accounts)

User ID, Password

Accept/Reject

PMT query

PMT Responses

PMT query

PMT response

PMT query

PMT response

Set: S₁

Set: Sₘ

Account Location Privacy

(Websites where Alice already has accounts)
Framework Design

User ID, Password

User (Alice)

Requester (Website A)

PMT query

PMT Responses

Directory (3rd party)

PMT query

PMT response

Responders (Websites where Alice already has accounts)

User ID, Password

User ID Responder Address

alice@xxx.com

xxxxx.edu

pseudo address 1

pseudo address 2

When Directory is trusted for account location privacy

When Directory is untrusted for account location privacy
Anonymous Communication

Tor (The Onion Router) network enables anonymous communication, which can hide the identities of the requester and responders when the directory is untrusted for account location privacy.

A customized Tor network for our prototype system, across 8 different datacenters in Europe and North America.

Design :: Security for the Requester

Response 1

Query 2 ($p'$)

"password already exists"

Responder: got 1 bit of information about the bloom filter of $p$. But $p$ is rejected and no longer used by the user.
Design :: Security for the Requester

- It is hard for malicious responders to exploit choice of a similar password since the user would then be required to choose a different password to register.

- If multiple responders exist for a user, no single responder knows which responder determines the result of the current run of the protocol.
Design :: Security for Each Responder

Requester (malicious) \quad \leftrightarrow \quad Directory (3rd party) \quad \rightarrow \quad \text{Responders} (\text{Websites where the user already has accounts})

Set: \(S_f\) \quad \rightarrow \quad Set: \(S_m\)

Hard to invert Bloom filters to get plaintext passwords if hash functions are costly to compute
Design :: Security for Each Responder

Requester (malicious) \[\rightarrow\] Directory (3rd party)

Set: \(S_f\)

Respnders (Websites where the user already has accounts)

Directory (3rd party) \[\rightarrow\] WS

Set: \(S_m\)

Hard to determine the identities of responders since account location privacy is guaranteed.

Add honey passwords into responder's sets.
Design :: Security for Each Responder

Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user's confirmation to proceed with the protocol.

Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user’s confirmation to proceed with the protocol.
Design :: Security for Each Responder

Requester (malicious) → Directory (3rd party) → Responders (Websites where the user already has accounts)

Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user’s confirmation to proceed with the protocol.
Design :: Security for Each Responder

Requester (malicious) \rightarrow \text{Directory (3rd party)} \rightarrow \text{Responders (Websites where the user already has accounts)}

Set: $S_r$

Set: $S_m$

Directory can send the user a confirmation URL upon receiving queries from the requester and requires the user’s confirmation to proceed with the protocol.

Probabilistic Model Checking

Adversary \rightarrow \text{Responders (Websites where Alice already has accounts)}
Probabilistic Model Checking

Adversary
(Markov Decision Process)

Responders
(Websites where Alice already has accounts)

Prior knowledge about Alice's passwords (the "dictionary")

Adversary
(Markov Decision Process)

Responders
(Websites where Alice already has accounts)
Probabilistic Model Checking

Prior knowledge about Alice’s passwords (the “dictionary”)

Adversary
(Markov Decision Process)

PMT
(with budgets)

Login
(with budgets)

Responders
(Websites where Alice already has accounts)

Adversary’s goal: to compromise at least one account on those responders.
### Probabilistic Model Checking

- **Max. Prob. of the adversary's success** (at least one account takeover)

### Difficulty of guessing a user's passwords, given different levels of prior knowledge

- **Dictionary size (bits)**
- **# of websites: 12**
- **Login budgets: 9**

---

### Probabilistic Model Checking

- **Max. Prob. of the adversary's success** (at least one account takeover)

### Difficulty of guessing a user's passwords, given different levels of prior knowledge

- **Dictionary size (bits)**
- **# of websites: 12**
- **Login budgets: 9**
**Probabilistic Model Checking**

Max. Prob. of the adversary’s success (at least one account takeover)

![Graph showing the maximum probability of the adversary’s success against dictionary size (bits) for different login budgets.](image)

- PMT budget = 9
- PMT budget = 6
- PMT budget = 3
- PMT budget = 0

# of websites: 12
Login budgets: 9

Dictionary size (bits)

---

**Probabilistic Model Checking**

Max. Prob. of the adversary’s success (at least one account takeover)

![Graph showing the maximum probability of the adversary’s success against dictionary size (bits) for different login budgets.](image)

- PMT budget = 9
- PMT budget = 6
- PMT budget = 3
- PMT budget = 0

# of websites: 12
Login budgets: 9

Dictionary size (bits)
Probabilistic Model Checking

Max. Prob. of the adversary's success (at least one account takeover)

Dictionary size (bits)

- PMT budget = 9
- PMT budget = 6
- PMT budget = 3
- PMT budget = 0

# of websites: 12
Login budgets: 9

Decent amount of prior knowledge
Max. Prob. of the adversary’s success (at least one account takeover)

# of websites: 12
Login budgets: 9

Not too guessable

a password with only 3 randomly generated characters (a-z, A-Z, 0-9).

Implementation & Performance
Implementation :: Prototype

- **Language**
  - C language for crypto implementation
  - Go language for other parts of the prototype system

- **Cryptosystem Implementation**
  - Elliptic-curve ElGamal cryptosystem
    - Elliptic curves: secp160r1, secp192r1 *(default)*, secp224r1, secp256r1
    - Key length: 160 bits, 192 bits, 224 bits, 256 bits.

- **Bloom Filter parameter**
  - Bloom Filters with $2^{20}$ false positive rate

Implementation :: Experimental Setup

- **Requester x 1, Directory x 1** *(@Our department):*
  - 2.67GHz * 8 physical cores, 72GiB RAM, 1Gb/s network

- **Responders x 1, 32, 64, 96, 128** *(@Our department):*
  - All responders were split evenly across 4 hosts.
  - 2.3GHz * 32 physical cores, 128GiB RAM, 1Gb/s network

- **Tor nodes** *(@ Amazon AWS “m4.large” instances, spread across datacenters in Europe and North America)*

- **Default parameters:**
  - Number of responders: 64
  - Number of similar passwords: 1000
  - Elliptic curves: secp192r1
Performance :: Response Time

Measured at the requester

m: Number of responders

Number of similar passwords

Trusted directory Untrusted directory (Tor circuits)
Performance :: Response Time

Number of similar passwords

Trusted directory Untrusted directory (for circuits)

How the number of participating responders would impact the response time
Performance :: Response Time

Number of similar passwords

Response Time (s)

Trusted directory

Untrusted directory (Tor circuits)

Does a responder actually needs to deploy a password set of this large?

Detection Rate & Scalability :: Empirical Study

Prior Empirical Study about Password Reuse [1]

- An estimate of probability of detecting password reuse as a function of the number of similar passwords.
- Detection rate increases sharply when set of similar passwords is small. Adding to similar-password set doesn’t improve detection much, but it does increase overhead.

Detection Rate & Scalability :: Empirical Study

Prior Empirical Study about Password Reuse [1]

- An estimate of probability of detecting password reuse as a function of the number of similar passwords.
- Detection rate increases sharply when set of similar passwords is small. Adding to similar-password set doesn’t improve detection much, but it does increase overhead.
Detection Rate & Scalability :: Results

**True detection rate maximization**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate

<table>
<thead>
<tr>
<th>( n )</th>
<th>( t_{\text{goal}} )</th>
<th>( m )</th>
<th>( n )</th>
<th>( t_{\text{tdr}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>( .343 )</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>( .805 )</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>( n )</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>( n )</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>( n )</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>( n )</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>( n )</td>
</tr>
</tbody>
</table>

**Max qualifying responses per sec.**

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>( 1 )</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>( 12 )</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>( 19 )</td>
</tr>
</tbody>
</table>

Untrusted directory

(Qualifying response: <= 8s)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>( 40 )</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>( 40 )</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>( 40 )</td>
</tr>
</tbody>
</table>

Trusted directory

(Qualifying response: <= 5s)

118

---

Detection Rate & Scalability :: Results

**True detection rate maximization**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate

<table>
<thead>
<tr>
<th>( n )</th>
<th>( t_{\text{goal}} )</th>
<th>( m )</th>
<th>( n )</th>
<th>( t_{\text{tdr}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>( .343 )</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>( .805 )</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>( n )</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>( n )</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>( n )</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>( n )</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>( n )</td>
</tr>
</tbody>
</table>

**Max qualifying responses per sec.**

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>( 1 )</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>( 12 )</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>( 19 )</td>
</tr>
</tbody>
</table>

Untrusted directory

(Qualifying response: <= 8s)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>( 40 )</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>( 40 )</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>( 40 )</td>
</tr>
</tbody>
</table>

Trusted directory

(Qualifying response: <= 5s)

118

---

Detection Rate & Scalability :: Results

**True detection rate maximization**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate

<table>
<thead>
<tr>
<th>( n )</th>
<th>( t_{\text{goal}} )</th>
<th>( m )</th>
<th>( n )</th>
<th>( t_{\text{tdr}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>( .343 )</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>( .805 )</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>( n )</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>( n )</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>( n )</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>( n )</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>( n )</td>
</tr>
</tbody>
</table>

**Max qualifying responses per sec.**

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>( 1 )</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>( 12 )</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>( 19 )</td>
</tr>
</tbody>
</table>

Untrusted directory

(Qualifying response: <= 8s)

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>( w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td>( 40 )</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>( 40 )</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>( 40 )</td>
</tr>
</tbody>
</table>

Trusted directory

(Qualifying response: <= 5s)

118

---
### Detection Rate & Scalability :: Results

**True detection rate maximization**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

<table>
<thead>
<tr>
<th>$t_{\text{goal}}$</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
<th>0.08</th>
<th>0.09</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$m$</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>$t_{\text{dtr}}$</td>
<td>0.003</td>
<td>0.006</td>
<td>0.008</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

**Max qualifying responses per sec.**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

<table>
<thead>
<tr>
<th>Max qualifying responses per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{goal}}$</td>
</tr>
<tr>
<td>$m$</td>
</tr>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$t_{\text{dtr}}$</td>
</tr>
</tbody>
</table>

**Trusted directory** (Qualifying response: $\leq 5s$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>95</td>
<td>61</td>
<td>47</td>
<td>33</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

**Untrusted directory** (Qualifying response: $\leq 8s$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>105</td>
<td>65</td>
<td>42</td>
<td>32</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

**Number of similar passwords (or set size) at responders**

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>95</td>
<td>61</td>
<td>47</td>
<td>33</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

---

### Detection Rate & Scalability :: Results

**True detection rate maximization**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

<table>
<thead>
<tr>
<th>$t_{\text{goal}}$</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
<th>0.08</th>
<th>0.09</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$m$</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>$t_{\text{dtr}}$</td>
<td>0.003</td>
<td>0.006</td>
<td>0.008</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

**Max qualifying responses per sec.**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

<table>
<thead>
<tr>
<th>Max qualifying responses per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{goal}}$</td>
</tr>
<tr>
<td>$m$</td>
</tr>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$t_{\text{dtr}}$</td>
</tr>
</tbody>
</table>

**Trusted directory** (Qualifying response: $\leq 5s$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>95</td>
<td>61</td>
<td>47</td>
<td>33</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

**Untrusted directory** (Qualifying response: $\leq 8s$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>105</td>
<td>65</td>
<td>42</td>
<td>32</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

**Number of similar passwords (or set size) at responders**

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>95</td>
<td>61</td>
<td>47</td>
<td>33</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

---

### Detection Rate & Scalability :: Results

**True detection rate maximization**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

<table>
<thead>
<tr>
<th>$t_{\text{goal}}$</th>
<th>0.01</th>
<th>0.02</th>
<th>0.03</th>
<th>0.04</th>
<th>0.05</th>
<th>0.06</th>
<th>0.07</th>
<th>0.08</th>
<th>0.09</th>
<th>0.10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$m$</td>
<td>1</td>
<td>10</td>
<td>17</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td>$t_{\text{dtr}}$</td>
<td>0.003</td>
<td>0.006</td>
<td>0.008</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

**Max qualifying responses per sec.**

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

<table>
<thead>
<tr>
<th>Max qualifying responses per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{goal}}$</td>
</tr>
<tr>
<td>$m$</td>
</tr>
<tr>
<td>$n$</td>
</tr>
<tr>
<td>$t_{\text{dtr}}$</td>
</tr>
</tbody>
</table>

**Trusted directory** (Qualifying response: $\leq 5s$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>95</td>
<td>61</td>
<td>47</td>
<td>33</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

**Untrusted directory** (Qualifying response: $\leq 8s$)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>105</td>
<td>65</td>
<td>42</td>
<td>32</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

**Number of similar passwords (or set size) at responders**

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>95</td>
<td>61</td>
<td>47</td>
<td>33</td>
<td>27</td>
<td>22</td>
</tr>
</tbody>
</table>

---
Detection Rate & Scalability :: Results

True detection rate maximization

Given a target response time constraint, how to choose number of similar passwords and number of participating responders to maximize true detection rate.

Max qualifying responses per sec.

<table>
<thead>
<tr>
<th>#</th>
<th>1</th>
<th>6</th>
<th>11</th>
<th>16</th>
<th>21</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>tdr</td>
<td>1.60</td>
<td>1.62</td>
<td>1.64</td>
<td>1.66</td>
<td>1.68</td>
<td>1.70</td>
</tr>
<tr>
<td>m</td>
<td>1.60</td>
<td>1.62</td>
<td>1.64</td>
<td>1.66</td>
<td>1.68</td>
<td>1.70</td>
</tr>
<tr>
<td>n</td>
<td>1.60</td>
<td>1.62</td>
<td>1.64</td>
<td>1.66</td>
<td>1.68</td>
<td>1.70</td>
</tr>
<tr>
<td>td</td>
<td>1.60</td>
<td>1.62</td>
<td>1.64</td>
<td>1.66</td>
<td>1.68</td>
<td>1.70</td>
</tr>
<tr>
<td>t</td>
<td>1.60</td>
<td>1.62</td>
<td>1.64</td>
<td>1.66</td>
<td>1.68</td>
<td>1.70</td>
</tr>
</tbody>
</table>

Trusted directory

Untrusted directory

A quick estimate:
A throughput of 50 qualifying responses per second is enough to enable each of the about 3x10^8 Internet users in the U.S. to setup or change passwords on more than 5 accounts per year.

Conclusion

- Password reuse is prevalent
Conclusion

- Password reuse is prevalent
- Password reuse makes credential stuffing effective and has been a top threat to web security

- We proposed a PMT protocol and then a framework to interfere with password reuse, to ensure:
  - Account security
  - Account location privacy
Conclusion

- Password reuse is prevalent
- Password reuse makes credential stuffing effective and has been a top threat to web security
- We proposed a PMT protocol and then a framework to interfere with password reuse, to ensure:
  - Account security
  - Account location privacy
- To our knowledge, our work is the first to actively interfere with password reuse by the same user at multiple websites

We believe even modest adoption of our framework would break the culture of password reuse.
Thanks!

Q&A