How to End Password Reuse On the Web

Ke Coby Wang          Michael K. Reiter

University of North Carolina at Chapel Hill
Password Reuse

same user,
same or similar password,
multiple websites.
Password Reuse

According to studies in past twenty years, most of users reuse same/similar passwords across multiple websites.

same user,
same or similar password,
multiple websites.

According to studies in past twenty years, most of users reuse same/similar passwords across multiple websites.
Leaked Passwords

Announced data breaches in the past **two months**:

- **Photography site 500px resets 14.8 million passwords after data breach**
  - 15 FEB 2019

- **Coinmama suffers a data breach of 450,000 emails and hashed passwords**
  - Coinmama, a crypto broker that specializes in letting users buy cryptocurrencies with credit cards, **announced Friday** that it suffered a data breach of 450,000 emails and hashed passwords. Security researchers said that the leaked data is an extension of a much larger breach that occurred last year. Coinmama also said that users who had their email addresses and hashed passwords compromised should reset their passwords.

- **The 773 Million Record "Collection #1" Data Breach**

- **Houzz discloses data breach, asks some users to reset passwords**
  - Citing an ongoing investigation, the company wouldn't say how or when the incident occurred.
Credential Stuffing

* via database breaches, phishing, malware, social engineering, etc.
The reuse of passwords is the No. 1 cause of harm on the internet.

--- Alex Stamos (former CSO, Facebook)

99% of compromised user accounts come from password reuse.

--- Patrick Heim (Head of Trust & Security, Dropbox)

Credential stuffing is enormously effective due to the password reuse problem.

--- Troy Hunt (Regional Director, Microsoft)
Existing Works

**Pre-attack**
- Leaked passwords
- Password reuse

**Attack**
- Credential stuffing

**Post-attack**
- Account takeovers

Detecting & crosschecking leaked passwords

Honey accounts, account activity monitoring, etc.

Two-factor, multi-factor authentication, etc.
Existing Works

Detecting & crosschecking leaked passwords

Leaked passwords

Password reuse

Pre-attack

Credential stuffing

Attack

Post-attack

Account takeovers

Two-factor, multi-factor authentication, etc.

Honey accounts, account activity monitoring, etc.
Existing Works

Detecting & crosschecking leaked passwords

Pre-attack

Leaked passwords

Password reuse

Credential stuffing

Post-attack

Account takeovers

Two-factor, multi-factor authentication, etc.

Honey accounts, account activity monitoring, etc.
Existing Works

Detecting & crosschecking leaked passwords

Pre-attack

Leaked passwords

Password reuse

Detecting password reuse on the client side

Attack

Credential stuffing

Two-factor, multi-factor authentication, etc.

Post-attack

Account takeovers

Honey accounts, account activity monitoring, etc.

Pre-attack ➔ Attack ➔ Post-attack
Our Work

Detecting & crosschecking leaked passwords

Pre-attack

Leaked passwords

Password reuse

Our work: server side

Attack

Credential stuffing

Two-factor, multi-factor authentication, etc.

Post-attack

Account takeovers

Honey accounts, account activity monitoring, etc.
Goals
Functionality
Functionality

User
(Alice)

(Re)set password

Requester
(Website A)
Functionality

User (Alice) (Re)set password

Requester (Website A)

Same/similar password exists for Alice?

Responders (Websites where Alice already has accounts)
Functionality

User (Alice)

(Re)set password

Accept/Reject

Requester (Website A)

Same/similar password exists for Alice?

Responders (Websites where Alice already has accounts)
Security and Privacy Goals
Security and Privacy Goals

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

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

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

![Diagram showing the process of (re)setting a password and checking for same/similar passwords across different websites](diagram.png)

**User** (Alice)  
**Requester** (Website A)  
**Responders** (Websites where Alice already has accounts)
Security and Privacy Goals

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

![Diagram showing the process of password reset and the interaction between the user, requester, and responders.](image)
Design
Key Elements in Framework Design

- Private Membership Test (PMT) protocol
  - A building block
Key Elements in Framework Design

- Private Membership Test (PMT) protocol
  - A building block

- Directory
  - A 3rd party
Key Elements in Framework Design

- Private Membership Test (PMT) protocol
  - A building block

- Directory
  - A 3\textsuperscript{rd} party

- Techniques for account location privacy
Key Elements in Framework Design

- Private Membership Test (PMT) protocol
  - A building block

- Directory
  - A 3\textsuperscript{rd} party

- Techniques for account location privacy

- Countermeasures for information leakage
Private Membership Test (PMT)

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

- **Requester** (element, \( p \))
- **Responder** (set: \( S \))

**Query message**

**Response message**

- \( p \) remains private.

**Learned only:**

- **whether \( p \) is in \( S \).**

- \( S \) remains private.

**Learned:**

- **Nothing, not even this PMT result.**
PMT Application

User
(Alice)

Requester
(Website A)

Responders
(Websites where Alice already has accounts)

Set: $S_1 = \{\text{Hash(bigbang)}, \text{Hash(bigbang!)} \ldots\}$

Set: $S_m = \{\text{Hash(domingo)}, \text{Hash(domingo!)} \ldots\}$
PMT Application

User (Alice) → “bigbang” → Requester (Website A)

Set: $S_1 = \{\text{Hash}(\text{bigbang}), \text{Hash}(\text{bigbang}!) \ldots\}$

Set: $S_m = \{\text{Hash}(\text{domingo}), \text{Hash}(\text{domingo}!) \ldots\}$

Responders (Websites where Alice already has accounts)
PMT Application

User (Alice)

"bigbang"

Reject

Requester (Website A)

Hash(bigbang) in S₁?

Hash(bigbang) in Sᵱ?

Set: S₁ = {Hash(bigbang), Hash(bigbang!) ...}

Set: Sᵱ = {Hash(domingo), Hash(domingo!) ...}

Responders (Websites where Alice already has accounts)

PMTs
Our PMT Protocol
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
(\text{element}, \ p)

Responder
(set: \ S)

One ciphertext only
Our PMT Protocol

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

**Requester** (element, $p$)

**Responder** (set: $S$)

- Query 1 for Alice
- Query 2 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”
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)

Responders (Websites where Alice already has accounts)

<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>
Directory

- **User (Alice)**
- **Requester (Website A)**
- **Directory (3rd party)**
- **Responders (Websites where Alice already has accounts)**

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

User
(Alice)

Requester
(Website A)

Directory
(3rd party)

Responders
(Websites where Alice already has accounts)
Framework Design

User (Alice)

Requester (Website A)

Directory (3rd party)

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) -> Set: $S_1$

Responders (Websites where Alice already has accounts) -> 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)

Set: $S_1$

Respnders
(Websites where Alice already has accounts)

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

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 ID, Password → Requester (Website A) → Directory (3rd party)

PMT query → PMT response → Set: S₁

PMT query → PMT response → Set: Sₘ

Responders
(Websites where Alice already has accounts)
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)

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)

USER ID, PASSWORD

Accept/Reject

Requester (Website A)

PMT query

PMT Responses

Directory (3rd party)

PMT query

PMT response

PMT query

PMT response

Set: $S_1$

Set: $S_m$

Responders

(Websites where Alice already has accounts)
Framework Design

User (Alice) -> Requester (Website A) -> Directory (3rd party) -> PMT query -> PMT response

Set: $S_1$

Responders (Websites where Alice already has accounts)

Set: $S_m$

Account Location Privacy

User ID, Password | Accept/Reject | PMT query | PMT Responses | PMT query | PMT response
Framework Design

User ID, Password

Set: \( S_1 \)

Requester (Website A)

Directory (3rd party)

User (Alice)

Responder

(Websites where Alice already has accounts)

User ID, Password

Accept/Reject

PMT query

PMT Responses

PMT query

PMT response

PMT query

PMT response

User ID

Responder Address

alice@xxx.com

xxxxx.edu

pseudo address 1

pseudo address 2

When Directory is trusted for account location privacy
Framework Design

**User** (Alice)

- User ID, Password
- Accept/Reject

**Requester** (Website A)

- PMT query
- PMT Responses

**Directory** (3rd party)

- PMT query
- PMT response

**Responders**

- (Websites where Alice already has accounts)

<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>xxxxx.edu</td>
</tr>
<tr>
<td></td>
<td>pseudo address 1</td>
</tr>
<tr>
<td></td>
<td>pseudo address 2</td>
</tr>
</tbody>
</table>

When Directory is untrusted for account location privacy

Set: $S_1$

Set: $S_m$
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.
Security
Against Malicious Requester

Requester
(malicious)

Directory
(3rd party)

Set: $S_1$

Set: $S_m$

Responders
(Websites where the user already has accounts)
Account location privacy makes it more difficult to determine the identities of responders

Responders
(Websites where the user already has accounts)
Against Malicious Requester

Requester
(malicious)

Directory
(3rd party)

Set: $S_1$

Set: $S_m$

Responders
(Websites where the user already has accounts)

Directory requires users’ confirmation to proceed with the protocol
User Confirmation :: Example

Requester (malicious) → Directory (3rd party)

Directory requires users’ confirmation to proceed with the protocol

Set: $S_1$

Responders
(Websites where the user already has accounts)

Set: $S_m$
User Confirmation :: Example

Requester
(malicious)

Confirmation?

Directory
(3rd party)

Set: $S_1$

Responders
(Websites where the user already has accounts)

Set: $S_m$

Directory requires users’ confirmation to proceed with the protocol
User Confirmation :: Example

Directory requires users’ confirmation to proceed with the protocol

Requester (malicious)

Directory (3rd party)

Responders
(Websites where the user already has accounts)

Set: $S_1$

Set: $S_m$
User Confirmation :: Example

Directory requires users’ confirmation to proceed with the protocol

Responders
(Websites where the user already has accounts)

Directory (3rd party)

Requester

Confirmation

Set: $S_1$

Set: $S_m$
Probabilistic Model Checking

Adversary

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

**Adversary**
(Markov Decision Process)

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

Prior knowledge about Alice’s passwords

Adversary
(Markov Decision Process)

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

**Adversary** (Markov Decision Process)

Prior knowledge about Alice’s passwords

PMT (with budgets)

Login (with budgets)

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

Adversary's goal: wins if she is able to compromise at least one account on those responders.

Prior knowledge about Alice’s passwords

Adversary
(Markov Decision Process)

PMT
(with budgets)

Login
(with budgets)

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

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

Dictionary size (bits)

# of websites: 12
Login budgets: 9

PMT budget = 9
PMT budget = 6
PMT budget = 3
PMT budget = 0
Probabilistic Model Checking

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

# of websites: 12
Login budgets: 9

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

Dictionary size (bits)
Probabilistic Model Checking

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

Baseline

Dictionary size (bits)

# of websites: 12
Login budgets: 9

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

# 68
Probabilistic Model Checking

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

# 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)

# of websites: 12
Login budgets: 9
Probabilistic Model Checking

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

Dictionary size (bits)

Decent amount of prior knowledge

# of websites: 12
Login budgets: 9
Probabilistic Model Checking

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

# of websites: 12
Login budgets: 9

Dictionary size (bits)

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

Not too guessable
Scalability
## Scalability Evaluation

Max qualifying responses per sec.

<table>
<thead>
<tr>
<th>number of responders</th>
<th>set size at responders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4304</td>
</tr>
<tr>
<td>10</td>
<td>2415</td>
</tr>
<tr>
<td>20</td>
<td>1478</td>
</tr>
<tr>
<td>30</td>
<td>1076</td>
</tr>
<tr>
<td>40</td>
<td>788</td>
</tr>
<tr>
<td>50</td>
<td>683</td>
</tr>
<tr>
<td>60</td>
<td>611</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>number of responders</th>
<th>set size at responders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>95</td>
</tr>
<tr>
<td>10</td>
<td>87</td>
</tr>
<tr>
<td>20</td>
<td>78</td>
</tr>
<tr>
<td>30</td>
<td>71</td>
</tr>
<tr>
<td>40</td>
<td>62</td>
</tr>
<tr>
<td>50</td>
<td>53</td>
</tr>
<tr>
<td>60</td>
<td>42</td>
</tr>
</tbody>
</table>

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

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

“Roundtrip” time measured at the requester
Scalability Evaluation

Max qualifying responses per sec.

<table>
<thead>
<tr>
<th>number of responders</th>
<th>set size at responders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>4304</td>
</tr>
<tr>
<td>10</td>
<td>2415</td>
</tr>
<tr>
<td>20</td>
<td>1478</td>
</tr>
<tr>
<td>30</td>
<td>1076</td>
</tr>
<tr>
<td>40</td>
<td>788</td>
</tr>
<tr>
<td>50</td>
<td>683</td>
</tr>
<tr>
<td>60</td>
<td>611</td>
</tr>
</tbody>
</table>

**Trusted directory**
(Qualifying response: <= 5s)

<table>
<thead>
<tr>
<th>number of responders</th>
<th>set size at responders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>95</td>
</tr>
<tr>
<td>10</td>
<td>87</td>
</tr>
<tr>
<td>20</td>
<td>78</td>
</tr>
<tr>
<td>30</td>
<td>71</td>
</tr>
<tr>
<td>40</td>
<td>62</td>
</tr>
<tr>
<td>50</td>
<td>53</td>
</tr>
<tr>
<td>60</td>
<td>42</td>
</tr>
</tbody>
</table>

**Untrusted directory**
(Qualifying response: <= 8s)
Scalability Evaluation

Max qualifying responses per sec.

<table>
<thead>
<tr>
<th>number of responders</th>
<th>set size at responders</th>
<th>number of responders</th>
<th>set size at responders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>4304</td>
<td>1013</td>
<td>492</td>
</tr>
<tr>
<td>10</td>
<td>2415</td>
<td>549</td>
<td>277</td>
</tr>
<tr>
<td>20</td>
<td>1478</td>
<td>336</td>
<td>182</td>
</tr>
<tr>
<td>30</td>
<td>1076</td>
<td>243</td>
<td>124</td>
</tr>
<tr>
<td>40</td>
<td>788</td>
<td>187</td>
<td>94</td>
</tr>
<tr>
<td>50</td>
<td>683</td>
<td>159</td>
<td>76</td>
</tr>
<tr>
<td>60</td>
<td>611</td>
<td>132</td>
<td>63</td>
</tr>
</tbody>
</table>

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

**Untrusted** directory
(Qualifying response: \( \leq 8s \))
### Scalability Evaluation

#### Max qualifying responses per sec.

<table>
<thead>
<tr>
<th>Responders</th>
<th>Set size at responders</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 0</td>
</tr>
<tr>
<td>2</td>
<td>1 6</td>
</tr>
</tbody>
</table>

A conservative 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 set up or change passwords on **more than 5 accounts per year**.
Summary
Summary

- A framework to detect password reuse:
  - Account security
  - Account location privacy
Summary

- A framework to detect password reuse:
  - Account security
  - Account location privacy
- A novel PMT protocol
Summary

- A framework to detect password reuse:
  - Account security
  - Account location privacy
- A novel PMT protocol
- First to actively interfere with password reuse on the server side
Summary

- A framework to detect password reuse:
  - Account security
  - Account location privacy
- A novel PMT protocol
- First to actively interfere with password reuse on the server side
- We believe even modest adoption of our framework would break the culture of password reuse and improve users’ account security on the web