SeifPass: a Secure Password Manager

MIT PRIMES
By: Cristian Gutu
Computer Science
Mentor: Albert Kwon
Problem

- Passwords: common, but vulnerable to offline attacks

User | Password
-----|---------
Alice | password123
Bob   | MyPetsName
...   | ...
## Hash

<table>
<thead>
<tr>
<th>User</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>hash(password123)</td>
</tr>
<tr>
<td>Bob</td>
<td>hash(MyPetsName)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Web Server

- pw db

### Attacker

- pw db

### Compute hash for common passwords

<table>
<thead>
<tr>
<th>Common passwords</th>
<th>Computed hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>password123</td>
<td>hash(password123)</td>
</tr>
<tr>
<td>MyPetsName</td>
<td>hash(MyPetsName)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Salted Hash

<table>
<thead>
<tr>
<th>User</th>
<th>Hashed Password, Salt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>hash(password123 + 4abcd@!#), 4abcd@!#</td>
</tr>
<tr>
<td>Bob</td>
<td>hash(MyPetsName + abcd%), abcd%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Web Server

Attacker

<table>
<thead>
<tr>
<th>Common passwords</th>
<th>Computed hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>password123</td>
<td>hash(password123 + 4abcd@!#)</td>
</tr>
<tr>
<td>MyPetsName</td>
<td>hash(MyPetsName + abcd%)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Solution

- Architecture for password hardening using a remote cryptographic service

Benefits:
- No offline attacks

Hardening: encrypting password with secret key stored only on crypto service
Distributed Trust

User -> hash(pw) -> Web Server

Attacker

Web Server

hash(pw) -> Crypto Service

pw db

hash(hash(pw), $secret) -> Crypto Service

$harden(pw_0)$

$harden(pw_1)$

Hard to detect online attacks

How do we rotate $secret$?
Pythia

Cryptographic erasure

User

hash(pw)

Web Server

hash(pw)

hash(hash(pw), $secret)

Attack

Detects online attacks

Key rotation

Pythia

$secret

pw db

pw db

$secret
**Pythia - New User**

**User**
- user, pw

**Web Server**
- pw db
- t = random()
- x = randomize(pw)

**Pythia**
- w: web server id
- t: user id
- x: randomized password
- k = keytable[w]
- y = F(k, t, x)
- z = unrandomize(y)
- store: (user, t, z)
Compromise Recovery

No need for original password
User password does not change

Original pw db is useless

Key:
z_i : passwords
w: web server id
t: user id
x: randomized password
k: pythia instance for this web server
Limitations of Pythia

- May be hard to merge in large code base
- No guarantee applications will implement it
SeifPass Overview

User does not have to depend on application to secure password
Seif - Node.js Pythia Wrapper

- Pythia Service in Node.js
- Easy to use
- [https://github.com/naitsirc/seif](https://github.com/naitsirc/seif)

Dependencies:

- Relic Crypto Library
- pyrelic (python wrapper for Relic)
- Node.js
- Express.js (web application framework)
- [mongoDB](https://www.mongodb.com/)
Scalability Study

New Password Latency vs Number of Users

Latency (ms)

1000 2000 3000 4000 5000 6000

Number of Users
SeifPass - Sign up

Username
Cristian

Email address
Email

Password
Password

Sign up

Sign in
SeifPass - Sign in

Sign in

Username
Cristian

Password
Password

Sign in

Create account
SeifPass - New Password
SeifPass - Password Manager

Google

Password: Gt Element Not Normalized x:5ED5BF854962C807 DF5DC269E9DB0FAD C7ACA210C6ECD5CB 9F510E2042C7F13E y:2A259E8E2990BC62 7D2A2274A4191683 8965A966C9146D4D 3FC1FD2B87DD2EAD z:2A68E8E841B0894A 231DA57475C6E166 384762067B11FF131 98F68B6EC6F7916

Facebook

Password: Gt Element Not Normalized x:180919B0C9A28AE BBCFEEBC96C8FF13 F8FD826752191F27 5DB44DB4522B04DC y:22355E8593B3A388 6957FDE1268CE850 5FBDBA4B906283F 3230B2770F77953E z:51A8D5CB500601AD 8F8645C5EA551DA6 5B54E2573C2079E1 998B6F8C00095ED6
SeifPass - Delete Password

Delete password

Account
Facebook

Warning: once you delete a password, there is no going back!

Delete
Close
Acknowledgments

Thank you to Albert Kwon for mentoring and project suggestion!

Thank you Srini Devadas for the Computer Science Program!

Thank you MIT PRIMES for the opportunity!

Thank you to my parents for the support!