Attacking Authentication

Professor Larry Heimann
Web Application Security
Information Systems
Challenge from last class

• Importance of checking every step of the process

• Simple ways to defend against this attack
Data from an analysis of 320 million passwords recovered from rockyou.com in 2009

<table>
<thead>
<tr>
<th>Rank</th>
<th>Password</th>
<th>Number of Users with Password (absolute)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123456</td>
<td>290731</td>
</tr>
<tr>
<td>2</td>
<td>12345</td>
<td>79078</td>
</tr>
<tr>
<td>3</td>
<td>123456789</td>
<td>76790</td>
</tr>
<tr>
<td>4</td>
<td>Password</td>
<td>61958</td>
</tr>
<tr>
<td>5</td>
<td>iloveyou</td>
<td>51622</td>
</tr>
<tr>
<td>6</td>
<td>princess</td>
<td>35231</td>
</tr>
<tr>
<td>7</td>
<td>rockyou</td>
<td>22588</td>
</tr>
<tr>
<td>8</td>
<td>1234567</td>
<td>21726</td>
</tr>
<tr>
<td>9</td>
<td>12345678</td>
<td>20553</td>
</tr>
<tr>
<td>10</td>
<td>abc123</td>
<td>17542</td>
</tr>
<tr>
<td>11</td>
<td>Nicole</td>
<td>17168</td>
</tr>
<tr>
<td>12</td>
<td>Daniel</td>
<td>16409</td>
</tr>
<tr>
<td>13</td>
<td>babygirl</td>
<td>16094</td>
</tr>
<tr>
<td>14</td>
<td>monkey</td>
<td>15294</td>
</tr>
<tr>
<td>15</td>
<td>Jessica</td>
<td>15162</td>
</tr>
<tr>
<td>16</td>
<td>Lovely</td>
<td>14950</td>
</tr>
<tr>
<td>17</td>
<td>michael</td>
<td>14898</td>
</tr>
<tr>
<td>18</td>
<td>Ashley</td>
<td>14329</td>
</tr>
<tr>
<td>19</td>
<td>654321</td>
<td>13984</td>
</tr>
<tr>
<td>20</td>
<td>Qwerty</td>
<td>13856</td>
</tr>
</tbody>
</table>
Authentication Technologies

- Various technologies are used, often in combination:
  - HTML forms-based
  - Multi-factor (passwords & tokens, etc)
  - Client SSL certificates & smartcards
  - HTTP basic / digest authentication
  - Windows-integrated authentication
  - Authentication services (e.g. MS Passport)

- The majority of Internet applications use simple forms-based authentication.

- Most authentication flaws can arise with any technology.
The obvious stuff

- Weak passwords
- Ability to enumerate usernames
- Ability to brute force the login
More subtle variations

- The application may require strong passwords but not validate them fully (e.g. case-insensitive check).

- Login failure messages may be the same on-screen, but contain subtle differences in the HTML source.

- Password guessing may be blocked in the browser but still possible using a scripted attack, due to reliance on client-side controls, logic flaws, etc.
Exploiting common login defects

- Experiment to determine what password quality rules are enforced.

- Check whether credentials are being validated in full.

- Review every detail of failed login messages to find username enumeration bugs. Check the page source, HTTP headers, and response times.

- Experiment to identify any account lockout defenses.

- Identify every possible target for mounting a brute force attack.

- Perform password guessing attacks breadth-first not depth-first – that is, work through a list of common passwords trying each password with every username in turn. Start with the most obvious and common passwords.
Other authentication functions

- Most applications contain other functionality to support the primary login, which can often be used to attack the overall mechanism:
  - User registration
  - Password change
  - Account recovery
  - “Remember me”
Other authentication functions

• User registration functions very often contain username enumeration flaws, because the application indicates whether a chosen username is already registered.

• Password change functions may allow username enumeration and brute force password guessing even if these are blocked in the main login function.

• “Remember me” functions often contain logic flaws or access control defects:

  Set-Cookie: RememberUser=edgruberman

  Set-Cookie: autologin=true
Other authentication functions

- Account recovery functions often involve a secondary challenge which presents a considerably lower bar than the main login function (e.g. “Do I own a pet?”).

  **Forgot Your Password or User ID?**

  User Id: Tim

  When you registered your User Id, you provided a secret question.

  Your secret question, provided during registration, is:

  what street did you live on in sierra vista

  Enter the answer to your secret question:

  [Input field]

  [Continue button]

- Users assume that only they will see their challenge.

- An attacker can harvest a large number of challenges and choose the easy ones.

- Username enumeration and brute force password guessing may be possible even if these are blocked in the main login function.
Other authentication functions

• Instead of a secondary challenge, account recovery often uses a password “hint”.

• An attacker can harvest large numbers of hints and then start guessing.

• Following successful completion of the account recovery challenge, the application often lets you:
  • Jump straight into an authentication session.
  • Recover the existing password.
  • Set a new password directly.
  • Receive a recovery URL to an arbitrary email address you specify.
Class Demonstration

Dogbert’s Password Recovery Service for Morons

I don’t remember my password.

Is it “123”? 

That’s just spooky.

© Scott Adams, Inc./Dist. by UFS, Inc.
Securing authentication

**Use strong credentials**

- Rules for minimum length, appearance of different character types, upper and lower case, avoidance of dictionary words, etc.

- Ensure any system-generated values are unpredictable. Handle credentials secretively

- Use SSL for all authentication functions (both loading and submission of forms).

- Only transmit credentials using POST requests, and never pass them back to the client.

- Store credentials using salted one-way hashes.

- “Remember me” functions should only remember usernames.

- Implement a password change function.
Securing authentication

Validate credentials properly

• Validate in full, case-sensitively.

• Defend aggressively against unexpected events during login processing (catch all exceptions and immediately invalidate the session).

• Implement proper access control over user impersonation functions.
Securing authentication

Prevent information leakage

• Remember every piece of functionality where credentials are validated.

• Use a single code component to handle all failed login attempts, and return a generic message.

• 2 ways self-registration functions can be designed to prevent username enumeration:
  • The application can generate its own usernames in an unpredictable way, avoiding the need to disclose that a selected username already exists.
  • The application can use email addresses as usernames. The first stage of registration involves entering an email address, and the application sends an email containing a one-time registration URL or an indication that the address is already registered.
Securing authentication

**Prevent brute force attacks**

- Suspend accounts after a small number of failed logins (e.g. three). Optionally, reinstate accounts after a short period (e.g. 30 minutes).

- To prevent information leakage, do not identify that any specific account has been suspended – after a failed login, simply state that accounts are suspended after a small number of failures.

- Do not disclose the metrics of the suspension policy.

- If an account is suspended, reject login attempts without checking the credentials, and records an additional failed login.

- Per-account measures will not prevent a stealthy breadth-first attack (for example, targeting every username with a small number of weak passwords).

- To defend against these attacks, controls like CAPTCHAs can be used.
Securing authentication

Defend the password change function

• Allow access to authenticated users only.

• Do not allow users to specify a username (either on-screen or in a hidden request parameter).

• Require the existing password to be supplied.

• Defend against password guessing and information leakage.

• Notify the user via email that their password has been changed.
Securing authentication

**Defend the account recovery function**

- Do not use password “hints”

- To enable account recovery, send a one-time URL to the email address which the user provided during registration. Visiting the URL should allow the user simply to specify a new password.

- A secondary challenge may also be used before the one-time URL is sent:
  - It should use the same question (or set of questions) for all users, rather than user-specified questions.
  - Responses should contain reasonable entropy (e.g. name of first school is preferable to favorite color).
  - Defend against username enumeration and brute force attacks.
Next Class:

Lab 1 on Authentication, Simple Attacks

You will need the following installed on a laptop before next class:

1. Git (at least 1.7.x; 1.8.x preferred)
2. Rails (3.2.13)
3. Gems -- rake (10.1.0), faker (1.2.0), thin (1.5.1), will_paginate (3.0.4), and sqlite3 (1.3.8)
4. Burp Suite (free version from http://portswigger.net/burp/download.html is fine)
5. Firefox or Chrome with appropriate extensions, tools for carrying out simple attacks