AuthScan: Automatic Extraction of Web Authentication Protocols from Implementations

Guangdong Bai\textsuperscript{1}
Guozhu Meng\textsuperscript{1}
Prateek Saxena\textsuperscript{1}
Yang Liu\textsuperscript{3}

Jike Lei\textsuperscript{1}
Sai Sathyanarayan Venkatraman\textsuperscript{1}
Jun Sun\textsuperscript{2}
Jin Song Dong\textsuperscript{1}

\textsuperscript{1}National University of Singapore
\textsuperscript{2}Singapore University of Technology and Design
\textsuperscript{3}Nanyang Technological University
Web Authentication Schemes & Single Sign-On

- **Web Authentication**

- **Single Sign-On (SSO)**
  - BrowserID (Mozilla)
  - Facebook Connect
    - 250+ Million users, 2,000,000 websites
  - OpenID
    - one billion users, 50,000 websites
  - ...

Alice

**Identity Provider (IDP)**

- e.g., 
  - Facebook
  - Google
  - Windows

**Service Provider (SP)**

- e.g., 
  - eBay
  - Classifieds
Implementations Can have Bugs!!

- Web Authentication
  - Password Guessing
  - Session/Cookie Stealing
  - ...

- Harder in SSO implementations
  - Vulnerabilities [BlackHat’07, Oakland’12, CCS’12, USENIX Security’12]
Is Manual Analysis Possible?

- Manual analysis is impractical
  - Closed source
  - Numerous implementations
    - OAuth 1.0 & 2.0: 47 implementations

OAuth (RFC 5849 & RFC 6749)
Can’t We Verify the Web Authentication?

- Previous protocol verification: design-level protocol specifications

It is the IMPLEMENTATION that security relies on!!

Implementation == Specification?

[CSNT’11] [SOFSEM’11]

[Oakland’12] [CCS’12]
Our Solution & Contributions

- **#1** Automatically extract protocols from implementations
- **#2** Checking extracted protocols for vulnerabilities

- **Automatic extraction techniques** to extract protocol specifications
- **AuthScan**: an end-to-end framework
- Find 7 security flaws in the **real-world** implementations
Examples #1: Freshness Problem in BrowserID Imp

• Missing Nonce
  – May lead to replay attacks
Example #2: Logic Flaw in Using Windows Live ID

- Using Publicly-Known Values as Tokens
  - Keep **constant** across multiple login sessions and the values are **publicly-known**
  - e.g., email, publicly-known id, hash(email), etc.

- Flaw found in credential cookies in Sina Weibo

```
GET http://www.weibo.com/msn/bind.php
HTTP/1.1
User-Agent: Mozilla/5.0
Host: www.weibo.com
Cookie: msn_cid=412ee98792885346
Connection: Keep-Alive

msn_id can be retrieved from profile page on MSN space !!!
```
Many More Vulnerability Examples

• Guessable Token
• Unchecked Referrer
  – Leading to CSRF attack
• Secret Token Leakage
• Short-length Token

Is there a generalized method to detect all these vulnerabilities?
Our Approach
AuthScan: Overview

Protocol Extraction
Protocol Extraction & Challenge

**Extraction**: to infer protocol from these available code and messages exchanged
- Protocol steps
- **Semantics of data element** exchanged in each step
  - Signature, cipher text, nonce, etc.

**Challenge**: Partially available implementation
- Partial code (client-side JavaScript code)
- HTTP messages exchanged

**Insight**: Hybrid Inference
- Whitebox Program Analysis
- Blackbox Differential Fuzzing
Whitebox Program Execution Analysis

Alice & ksad381s...nx89Ds

<table>
<thead>
<tr>
<th>Code Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>window.addEventListener('message',function(event) {</td>
</tr>
<tr>
<td>var id=extractUser(event.data); (u)</td>
</tr>
<tr>
<td>var idpSign=extractSign(event.data); (X)</td>
</tr>
<tr>
<td>var data=id; (u)</td>
</tr>
<tr>
<td>var idpPubKey=loadPubKey(); (K)</td>
</tr>
<tr>
<td>if(verify(data, idpSign, idpPubKey)){</td>
</tr>
<tr>
<td>{...}});</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>{...},false);</td>
</tr>
<tr>
<td>(X={u})^{K^{-1}}</td>
</tr>
</tbody>
</table>
Blackbox Differential Fuzzing

• To identify the relations between HTTP data

  HTTP GET  uname= bob@idp.com
  HTTP 200  token= 7aAod5...as09uA

  assoc(uname, token)

• To identify the relations between HTTP data and participants

  HTTP 200  c_user = 10299987
  ......
  HTTP 200  c_user = 10299987

  assoc(uid, c_user)

• To eliminate the redundant messages and data
• To identity long-lived and short-length token
AuthScan: Overview

Protocol Extraction

Protocol Verification

Back-end Verification Tool

Counter Example

SP
Server
SP
Client
IDP
Client
IDP
Server
Token
U_ID, pwd
Token
Token
SP_ID

Security Analyst

User-Agent

IDP

SP

15
Attacker Models, Properties & Assumptions

- Attacker models considered in AuthScan
  - Network Attacker
  - Web Attacker
    - Same-origin policy, Referrer, postMessage

- Properties
  - Authentication
    - Correspondence [oakland’ 93]
  - Secrecy

- Assumptions
  - Correct Cryptographic Algorithms
  - Knowledge of Participants
    - Each one knows the others’ identifiers
AuthScan: Overview

Protocol Extraction

Protocol Verification

Back-end Verification Tool

Attack Confirmation

Counterexample

Security Analyst

User-Agent

IDP

SP

SP_ID

U_ID, pwd

Token

Token

Token
Evaluation
AuthScan Evaluation

• Implementation
  – Implemented as a Firefox add-on
  – Uses ProVerif as the back-end

• Evaluation Subjects
  – BrowserID (three websites)
  – Facebook Connect (two websites)
  – Windows Live ID
  – Standalone websites

• Setup
  – Test harness
    • pre-registered user accounts
  – Protocol principals & public keys
  – Cryptographic functions
    • Mozilla jwcrypto used in BrowserID

**Millions of Users are Impacted!**
## Vulnerabilities Found

- 7 real-world vulnerabilities
  - 6 previously unknown

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Deployed SSO</th>
<th>#Flaws*</th>
<th>Flaw Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>myfavoritebeer.com</td>
<td>BrowserID</td>
<td>2(T1, T2)</td>
<td>T1 Missing nonce in BrowserID</td>
</tr>
<tr>
<td>openphoto.me</td>
<td></td>
<td>2(T1, T2)</td>
<td>T2 Unchecked Referrer in SPs (leading to CSRF attack)</td>
</tr>
<tr>
<td>developer.mozilla.org</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>ebayclassifieds.com</td>
<td>Facebook Connect</td>
<td>2(T3, T4)</td>
<td>T3 Secret token leak in FB connect</td>
</tr>
<tr>
<td>familybuilder.com</td>
<td></td>
<td>1(T3)</td>
<td>T4 Secret token leak in SP</td>
</tr>
<tr>
<td>Weibo.com</td>
<td>Windows Live ID</td>
<td>1(T5)</td>
<td>T5 Using Publicly-Known Values as Tokens</td>
</tr>
<tr>
<td>iyermatrimony.com</td>
<td>---</td>
<td>1(T6)</td>
<td>T6 Guessable Token</td>
</tr>
<tr>
<td>meetingmillionaires.com</td>
<td>---</td>
<td>1(T7)</td>
<td>T7 Short-Length Token</td>
</tr>
</tbody>
</table>

* With Overlapping
Example #3: Secret Token Leakage in FB Connect

- Secret Token Leakage
  - Secret tokens are transmitted through **unencrypted** channels

- Flaw found in secret cookie in Facebook Connect

![Diagram showing the transmission of Uname/pwd, c_user/xs, OAuth token, and www.facebook.com/login.php and www.facebook.com/dialog/permissions.request through HTTP and HTTPS channels.]

www.facebook.com/login.php

www.facebook.com/dialog/permissions.request
Example #4: Guessable Token

- Guessable Token

http://www.iyermatrimony.com/login/intermediatelogin.php?
  sds=QdR.j/ZJEX./A&
  sdss=Tf/GpQptuzuEs&
  sde=U1ZsU01UZ3dOVE01

- Keep constant
  First 14 characters: keep constant

- Incremented by one across accounts whose IDs are consecutive
Example #5: Short-Length Token

- Short-Length Token

http://app.icontact.com/icp/mmail-mprofile.pl?
r=36958596&l=2601&m=318326&c=752641&s=21DS

User ID  Constant among different users’ sessions  Alpha-numeric string

(10 + 26)^4 Possible Values
Attacker: 500 “probes”/ min
## Scalability

<table>
<thead>
<tr>
<th>Web Sites</th>
<th>Time(s) (Excluding Verification Time)</th>
<th>Verification Time</th>
<th>Fuzzing Round</th>
</tr>
</thead>
<tbody>
<tr>
<td>myfavoritebeer.com</td>
<td>113</td>
<td>3.0</td>
<td>20</td>
</tr>
<tr>
<td>openphoto.me</td>
<td>72</td>
<td>3.0</td>
<td>22</td>
</tr>
<tr>
<td>developer.mozilla.org</td>
<td>96</td>
<td>3.0</td>
<td>28</td>
</tr>
<tr>
<td>ebayclassifieds.com</td>
<td>127</td>
<td>58.7</td>
<td>107</td>
</tr>
<tr>
<td>familybuilder.com</td>
<td>110</td>
<td>58.7</td>
<td>77</td>
</tr>
<tr>
<td>Weibo.com</td>
<td>30</td>
<td>0.03</td>
<td>78</td>
</tr>
<tr>
<td>iyermatrimony.com</td>
<td>5.33</td>
<td>0.04</td>
<td>51</td>
</tr>
<tr>
<td>meetingmillionaires.com</td>
<td>4.72</td>
<td>0.04</td>
<td>30</td>
</tr>
</tbody>
</table>
Conclusion & Take-away

• AuthScan: an end-to-end framework to extract web authentication protocols from their implementations
  – Hybrid inference techniques for protocol extraction
  – Found 7 vulnerabilities in real-world web-sites

• The devil is in the details!
Reference

• **[Oakland’12]** R. Wang, S. Chen, and X. Wang.

• **[CCS’ 12]** S.T. Sun, K. Beznosov.
  The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems.

  On Breaking SAML: Be Whoever You Want to Be.

• **[BlackHat’ 07]** E. Tsyrklevich and V. Tsyrklevich.
  Single Sign-On for the Internet: A Security Story

• **[CSNT’11]** S. Pai, Y. Sharma, S. Kumar, R. M. Pai, and S. Singh.
  Formal verification of OAuth 2.0 using Alloy framework.

• **[SOFSEM’11]** M. Miculan, C. Urban
  Formal analysis of Facebook Connect single sign-on authentication protocol.

• **[Oakland’ 93]** T. Y. C. Woo and S. S. Lam.
  A Semantic Model for Authentication Protocols.
Thank you!

We are hiring!!
Phd & Post-doc in NUS, NTU and SUTD!
Contact: yangliu@ntu.edu.sg