The Web SSO Standard
OpenID Connect

In-Depth Formal Security Analysis
and Security Guidelines

Daniel Fett, Ralf Küsters, Guido Schmitz

Institute of Information Security
University of Stuttgart
Our Contributions

OpenID Connect 1.0 with Discovery and Dynamic Registration:

- Developed formal model of the standard
  - Based on most comprehensive model of the web to date (extension of S&P 2014).

- Formalized central security properties
  - Authentication
  - Authorization
  - Session Integrity

- Proved security for (fixed) standard (see security guidelines)

Paper to appear at CSF 2017
All details: TR available at https://sec.informatik.uni-stuttgart.de
Formal Analysis of Web Applications and Standards

- Many flaws and attacks in web applications
- Increasing complexity of web sites & systems
- Interaction of different components

Formal methods required to ...

- precisely specify security properties
- carry out security proofs
Specifications for the web are spread across many sources with mutual dependencies:

- Standards and RFCs
  - HTTP/1.1 and HTTP/2 Standards
  - W3C HTML5
  - W3C Web Storage
  - WHATWG Fetch
  - W3C Cross-Origin Resource Sharing
  - RFCs (6265, 6797, 6454, 2616, …)

- Browser implementations
  - Google Chrome
  - Mozilla Firefox
  - …

Model provides coherent view of core aspects of the web
Web Model

Network

Browser

Dolev-Yao Process

WebAttacker

NetworkAttacker

Web Server

DNS

S1

Sj

B1

Bk
Web Browser Model

Including ...

- DNS, HTTP, HTTPS
- window & document structure
- scripts
- attacker scripts
- web storage & cookies
- web messaging & XHR
- message headers
- redirections
- security policies
- dynamic corruption
- ...

Origin: https://example.com
Web Model

Browser → $B_1$ → Dolev-Yao Process

WebAttacker → NetworkAttacker

NetworkAttacker → Network

Network → DNS$_1$ → $S_1$

Network → DNS$_1$ → $S_j$

Network → Web Server
Limitations

- No language details
- No user interface details
- No byte-level attacks (e.g., buffer overflows)
- Abstract view on cryptography and TLS
Previous Work

[SP 2014, ESORICS 2015, CCS 2015, CCS 2016]

● Formal analysis of Mozilla’s BrowserID
  Main design goal: privacy
  - Found severe attacks
  - Proposed fixes for authentication and proved security
  - Privacy broken beyond repair

● Designed our own new SSO system: SPRESSO
  Provably provides strong authentication and privacy properties.

● Analysis of OAuth 2.0
  - Found attacks
  - Proposed fixes and proved security
Our Contributions

**OpenID Connect 1.0 with Discovery and Dynamic Registration:**

- Developed formal model of the standard
  - Based on most comprehensive model of the web to date (extension of S&P 2014).

- Formalized central security properties
  - Authentication
  - Authorization
  - Session Integrity

- Proved security for (fixed) standard (see security guidelines)

Paper to appear at CSF 2017

All details: TR available at https://sec.informatik.uni-stuttgart.de
OpenID Connect

- OpenID Connect 1.0 (OIDC): Web SSO
  - Goal: identity provider (IdP) assures relying party (RP) of user’s identity
  - Based on the OAuth 2.0 authorization framework (RFC 6749)
  - Introduces ID token as one-time proof of user’s identity
  - Several modes, options, and extensions:
    - 3 different modes of interaction
    - discovery
    - dynamic registration
    - …
Why Another Security Analysis?

• Subtle differences between OAuth 2.0 and OIDC 1.0
  - OAuth is not for authentication (although proprietary extensions exist)
  - New hybrid mode
  - Discovery and dynamic registration
    - ID Token
• (Also: more modular proof)
ID Token

• **One-time proof** of the user’s identity
• Issued by IdP, consumed by RP.
• Signed by IdP
• Contains following claims:
  - user identifier (unique at respective IdP)
  - issuer (IdP)
  - audience (RP)
  - ...
  - unique global user identifier
Implicit Mode

1. "Login with idp.com."

2. user authentication

3. redirect to rp.com with ID Token $IT$, Access Token $AT$

4. access URI (w/o token)

5. send $IT$, $AT$

6. logged in

7. retrieve data using $AT$
1. "Login with idp.com."

2. user authentication

3. Redirect to rp.com with Authorization Code AC in URI

4. Request URI with AC

6. logged in

5. retrieve IT, AT using AC

7. retrieve data using AT
Hybrid Mode

1. "Login with idp.com."
2. user authentication
3. Redirect to rp.com with Authorization Code \( IT, AT, AC \) in URI
4. access URI (w/o token)
5. Request URI with \( AC \)
6. retrieve \( IT', AT' \) using \( AC \)
7. logged in
8. retrieve data using \( AT' \)
Discovery and Dynamic Registration

1. user/idp identifier

For example: email address or URL

2. WebFinger

3. OIDC configuration

4. Dynamic Registration

OIDC core flow
Our Contributions

OpenID Connect 1.0 with Discovery and Dynamic Registration:

- Developed formal model of the standard
  - Based on most comprehensive model of the web to date (extension of S&P 2014).

- Formalized central security properties
  - Authentication
  - Authorization
  - Session Integrity

- Proved security for (fixed) standard (see security guidelines)

Paper to appear at CSF 2017
All details: TR available at https://sec.informatik.uni-stuttgart.de
Authentication Property

1. "Login with idp.com."
2. user authentication
3. redirect to rp.com with ID Token $IT$, Access Token $AT$
4. access URI (w/o token)
5. send $IT$, $AT$

logged in
Authorization Property

1. "Login with idp.com."

2. user authentication

3. redirect to rp.com with ID Token \textit{IT}, Access Token \textit{AT}

4. access URI (w/o token)

5. send \textit{IT}, \textit{AT}

6. retrieve data using \textit{AT}

7. retrieve data using \textit{AT}
Session Integrity

1. "Login with idp.com."
2. User authentication
3. Redirect to rp.com IT, AT
4./5. retr. URI, send IT, AT
5. "Logged in"
6. "User logged in (authn) or the user's data are accessed (authz) only if the user expressed her wish to log in before."
Our Contributions

OpenID Connect 1.0 with Discovery and Dynamic Registration:

- Developed formal model of the standard
  - Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  - Authentication
  - Authorization
  - Session Integrity
- Proved security for (fixed) standard (see security guidelines)

Paper to appear at CSF 2017

All details: TR available at https://sec.informatik.uni-stuttgart.de
Formal Model of OIDC

- Formal description of OIDC RP and OIDC IdP (with discovery and dynamic registration)
- Implements best practices and follows security guidelines
- Unbounded number of users (browsers), RPs, and IdPs
- Network attacker for authentication and authorization properties
- (Unbounded number of) web attackers for session integrity properties
- Browsers, RPs, and IdPs can become corrupted
Example: RP Checks an ID Token

Algorithm 20  Relying Party $R'$: Check id token.

1: \textbf{function} CHECK\_ID\_TOKEN(sessionId, id\_token, $s'$) \quad \rightarrow \text{Check id token validity and create service session.}
2: \quad \textbf{let} sessionId := $s'$\_sessions[sessionId] \quad \rightarrow \text{Retrieve session data.}
3: \quad \textbf{let} identity := sessionId[identity]
4: \quad \textbf{let} issuer := $s'$\_issuerCache[identity] \quad \rightarrow \text{Retrieve issuer.}
5: \quad \textbf{let} oидcConfig := $s'$\_oidcConfigCache[issuer] \quad \rightarrow \text{Retrieve OIDC configuration for that issuer.}
6: \quad \textbf{let} credentials := $s'$\_clientCredentialsCache[issuer] \quad \rightarrow \text{Retrieve OIDC credentials for issuer.}
7: \quad \textbf{let} jwks := $s'$\_jwksCache[issuer] \quad \rightarrow \text{Retrieve signing keys for issuer.}
8: \quad \textbf{let} data := extractmsg(id\_token) \quad \rightarrow \text{Extract contents of signed id token.}
9: \quad \textbf{if} data[iss] \neq \text{issuer} \quad \textbf{then}
10: \quad \quad \textbf{stop} \quad \rightarrow \text{Check the issuer.}
11: \quad \quad \textbf{if} data[aud] \neq \text{credentials[client\_id]} \quad \textbf{then}
12: \quad \quad \quad \textbf{stop} \quad \rightarrow \text{Check the audience against own client id.}
13: \quad \quad \textbf{if} checksig(id\_token, jwks) \neq \top \quad \textbf{then}
14: \quad \quad \quad \textbf{stop} \quad \rightarrow \text{Check the signature of the id token.}
15: \quad \quad \textbf{if} nonce \in \text{sessionId} \land data[nonce] \neq \text{session[nonce]} \quad \textbf{then}
16: \quad \quad \quad \textbf{stop} \quad \rightarrow \text{If a nonce was used, check its value.}
17: \quad \quad \textbf{let} $s'$\_sessions[sessionId][\text{loggedInAs}] := (issuer, data[sub]) \quad \rightarrow \text{User is now logged in. Store user identity and issuer.}
18: \quad \quad \textbf{let} $s'$\_sessions[sessionId][\text{serviceSessionId}] := \nu_4 \quad \rightarrow \text{Choose a new service session id.}
19: \quad \quad \textbf{let} request := sessionId[redirectEpRequest] \quad \rightarrow \text{Retrieve stored meta data of the request from the browser to the redirect endpoint in order to respond to it now. The request's meta data was stored in PROCESS\_HTTPS\_REQUEST (Algorithm 17).}
20: \quad \textbf{let} headers := [\text{RefererPolicy:origin}]
21: \quad \textbf{let} headers[Set-Cookie] := [\text{serviceSessionId}:(\nu_4, T, T, T)] \quad \rightarrow \text{Create a cookie containing the service session id.}
22: \quad \textbf{let} m' := \text{enc}_5((\text{HTTPResp, request[message].nonce, 200, headers, ok}, request[key])) \quad \rightarrow \text{Respond to browser's request to the redirect endpoint.}
23: \quad \textbf{stop} \langle [\text{request[sender], request[receiver], m'}], s' \rangle
Security Guidelines

- Mix-Up attack mitigation (send issuer identifier along with tokens)
- Fresh nonce for state (each time a new flow is stated)
- Referrer Policy to avoid code, token, and state leakage via Referer header
- Explicit user intention tracking (i.e., RP stores user’s choice of IdP)
- HTTP redirects with 303 (not 307)
- No open redirectors
- CSRF protection
- No third-party resources on endpoints
- TLS everywhere
- Sessions follow best practices, separate sessions before and after login
Our Contributions

OpenID Connect 1.0 with Discovery and Dynamic Registration:

- Developed formal model of the standard
  - Based on most comprehensive model of the web to date (extension of S&P 2014).

- Formalized central security properties
  - Authentication
  - Authorization
  - Session Integrity

- Proved security for (fixed) standard (see security guidelines)

Paper to appear at CSF 2017

All details: TR available at https://sec.informatik.uni-stuttgart.de
Theorem: OIDC fulfills security properties

- Authentication
- Authorization
- Session Integrity

Proof:

- Secondary security properties
  - discovery and dynamic registration are sane
  - id tokens and state do not leak
- Proof of theorem for each property separately by contradiction
Our Contributions

OpenID Connect 1.0 with Discovery and Dynamic Registration:

- Developed formal model of the standard
  - Based on most comprehensive model of the web to date (extension of S&P 2014).

- Formalized central security properties
  - Authentication
  - Authorization
  - Session Integrity

- Proved security for (fixed) standard (see security guidelines)

Thank you!

Paper to appear at CSF 2017
All details: TR available at https://sec.informatik.uni-stuttgart.de