OpenID Security Analysis and Evaluation

San-Tsai Sun, Kirstie Hawkey, Konstantin Beznosov

Laboratory for Education and Research in Secure Systems Engineering (LERSSE)
University of British Columbia
summary

• CSRF attacks
  – single-sign-on CSRF (force victim to login) (70%)
  – account profile CSRF (50%)
  – login CSRF (login as attacker) (73%)

• authentication response interception
  – impersonation (67%)
  – replay attack (6%)
agenda

• background
• approach and evaluation result
• countermeasures
OpenID

• open and user-centric Web single sign-on protocol

• OpenID Foundation (2007) [1]
  • Microsoft, Google, IBM, Yahoo, VeriSign, Facebook, PayPal, PingIdentity

• over **one billion** OpenID enabled user accounts provided by Google, Yahoo, AOL...[1]
how OpenID works

Identity Provider

user name: alice.myopenid.com
password: xxxxxxxxx
agenda

• background of OpenID
• approach and evaluation results
• countermeasures
methodology

1. vulnerability identification

- OpenID Spec
- Protocol Interpretation/Implementation
- sequence diagram
- known weakness
- RP_v1

2. vulnerability evaluation

- real RPs
- RP Evaluation
- exploits
- MLP_v1
- exploit Design
- attack vectors
- Model Checking
- attack traces
- Threat Model

3. countermeasure evaluation

- countermeasure evaluation results
- Countermeasure Evaluation
- RP_v2
- Countermeasure Implementation
- countermeasure
- Countermeasure Design
- Model Checking
- HLPSL
- Protocol Encoding
- HLPSL
OpenID sequence diagram

**Browser**
- login request: $I$ (identity url)
- IdP discovery

**ID server**
- $p, g, g^a \mod p$
- compute $k$
- store $h$
- redirect to IdP: $I, RP, [h]$

**RP**
- authentication request
- $g^b \mod p, h, H(g^{ab} \mod p) \otimes k$
- find $c$
- $I, RP, [h], [c]$
- user authentication (if $c$ is missing)

**IdP**
- check: $h, c$
- user authentication
- $n, c, \text{HMAC}_k(I, RP, h, n)$
- save $c$
- authentication response
- redirect to RP: $I, RP, h, n, \text{HMAC}_k(I, RP, h, n), c$

**IdP discovery**
- association
- gen: $b, h, k$

**User authentication**
- $g^b \mod p, h, H(g^{ab} \mod p) \otimes k$
- sign verification

**Identity**
- signature verification
- $I, RP, h, n, \text{HMAC}_k(I, RP, h, n)$
- valid/invalid

- $I$: identity url
- $h$: session handle
- $k$: session key
- $n$: nonce
- $c$: IdP cookie
- $p$: DH modulus
- $g$: DH generator
known weakness

Browser

ID server

RP

IdP

login request: I

I, RP, [h], [c]

I, RP, h, n, HMAC_k(I,RP,h,n)

check: h, c

check: credential

gen: n, c, HMAC_k(I,RP,h,n)

verify HMAC_k(I,RP,h,n) if h exists

or

I, RP, h, n, HMAC_k(I,RP,h,n)

valid/invalid

• I: identity url
• h: session handle
• k: session key
• p: DH modulus
• n: nonce
• c: IdP cookie

• RP: RP return_to url
• IdP: IdP endpoint url
• g: DH generator

find c

save c

redirect to IdP: I, RP, [h]

redirect to RP: I, RP, h, n, HMAC_k(I,RP,h,n), c

compute k

I, RP, [h], [c]

(authentication request)

user credential

user authentication

I, RP, [h], [c]

(authentication response)
threat model

• adversary: non-RP or IdP associated attackers
• adversary goal: unauthorized access/modification to users’ data hosted on RP
• Web poster: post comments
• Web attacker:
  – setup a malicious website
  – send malicious links via spam
  – deliver malicious content via Ads network
  – exploit web vulnerabilities (i.e., XSS) of benign websites
• network attacker:
  – setup an wireless access point
  – compromise client DNS resolution
threat assumptions

• RP, IdP, user machine, and browser are not compromised
• RP, IdP are not malicious
• user credentials on IdPs are secured
• cookies in the browser are secured (integrity and confidentiality)
non-considered threats

• availability threat
  – DoS by sending massive concurrent auth requests to an IdP
  – DoS by sending massive concurrent auth responses to an RP

• identity spoofing
  – phishing attacks by RP
  – exploits vulnerabilities on IdP

• integrity of IdP discovery process
  – altering discovery information
  – compromise RP DNS resolution
found weakness

authentication response acts as an one-time access token to an RP, but

• authentication response is not bound to a specific authentication request (non-associate)
• authentication request is not bound to a specific login request
• login request is not bound to the browser session
attack vectors

• CSRF
  – single sign-on (SSO) CSRF (force victim to login)
    • HTTP GET Auth Request CSRF [Web poster, Web attacker]
    • HTTP POST Login CSRF [Web attacker]
    • HTTP GET Login CSRF [Web poster, Web attacker]
  – account profile CSRF [Web poster, Web attacker]
  – login CSRF (login as attacker) [Web attacker]

• authentication response interception
  – impersonation [Network attacker]
  – replay attack [Network attacker]
SSO CSRF: HTTP GET Auth Request

Browser → ID server → RP → IdP

Authentication Request:

- Request: \( I, RP, c \) (authentication request)
- User credential

Authentication Response:

- Redirect to RP: \( I, RP, h, n, \text{HMAC}_{k}(I,RP,h,n) \)
- Check: \( c \) gen: \( n, h \)
- \( \text{HMAC}_{k}(I,RP,h,n) \)
- Check authentication

valid/invalid

A nonce associated with the auth request can stop the attack.
SSO CSRF: HTTP POST/GET login

Browser

ID server

RP

IdP

login request: I

(fetch request)

login request: I

compute k

redirect to IdP: I, RP, [h]

find c

save c

I, RP, h, c

(authentication request)

I, RP, h, c

I, RP, h, n, HMAC_k(I,RP,h,n)

(authentication response)

redirect to RP: I, RP, h, n, HMAC_k(I,RP,h,n), c

check: h,c

gen: n, HMAC_k(I,RP,h,n)

g^b mod p, h, H(g^{ab} mod p) \otimes k

verify HMAC_k(I,RP,h,n) if h exists

or

check_authentication

I, RP, h, n, HMAC_k(I,RP,h,n)

valid/invalid

associate the login request with the browser session can stop the attack

associate the login request with the browser session can stop the attack
login CSRF: login as the attacker

RP

IdP

login request

attacker user name and password

authentication request

authentication response

authentication response

associate the authentication response with the browser session can stop the attack

<img src="auth response" style="display:none">
impersonation and replay attack

Browser

ID server

RP

IdP

login request: I

(login request)

fetch I

IdP

compute k

redirect to IdP: I, RP, [h]

I, RP, [h], [c] (authentication request)

user authentication

user credential

find c

save

I, RP, h, n, HMAC_k(I,RP,h,n)

authentication response

redirect to RP: I, RP, h, n, HMAC_k(I,RP,h,n), c

user authentication (if c is missing)

check: h, c

gen: b, h, k

check: credential

gen: n, c,

HMAC_k(I,RP,h,n)

associate the authentication response with the browser session can stop the attack

I, RP, h, n, HMAC_k(I,RP,h,n)

verify HMAC_k(I,RP,h,n)

or

certify

check: credential

gen: n, c,
attack summary

• CSRF attacks
  – single-sign-on CSRF (force victim to login) (70%)
    • HTTP GET Auth Request (25%)
    • HTTP POST Login (50%)
    • HTTP GET Login (35%)
  – account profile CSRF (50%)
  – login CSRF (login as attacker) (73%)

• authentication response interception
  – impersonation (67%)
  – replay attack (6%)
agenda

• background of OpenID
• approach and evaluation result
• countermeasures
countermeasure

• when a new browser session initialized
  – RP generates a nonce $N = \text{HMAC}(\text{browser session id })$
  – issues a new cookie $C_N = N$
  – append a parameter $P_N=N$ to the OpenID login form

• when receive a login request
  – check if $P_N = C_N$ and $C_N = \text{HMAC}(\text{browser session id })$
  – initiate a new authentication request
  – append a parameter $R_N=N$ to the **return_to** URL

• when receive an authentication response
  – check if $R_N = C_N$ and $C_N = \text{HMAC}(\text{browser session id })$
characteristics of countermeasure

• compatible with existing OpenID
• do not require any additional storage on RP
• would not reveal browser session id
• protect from cookie overwrite
future work

• evaluate more RPs
• apply our methodology to other Web single sign-on protocol
  – Facebook connect
  – Microsoft Live ID
OpenID Security Analysis and Evaluation

san-tsai sun <santsais@ece.ubc.ca>

Department of Electrical and Computer Engineering
Laboratory for Education and Research in Secure Systems Engineering (LERSSE)