Analysing the Security of Google's implementation of OpenID Connect

Wanpeng Li & Chris J Mitchell

Information Security Group
Royal Holloway, University of London

Wanpeng.Li.2013@live.rhul.ac.uk

July 8, 2016

Overview

Why OpenID Connect?

Back to 2007 [Dinei, 2007]:

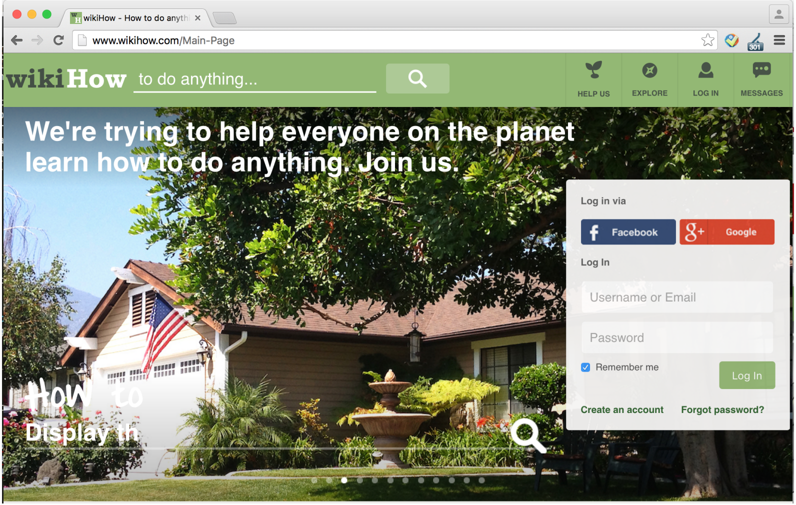
- the web service user had an average of 6.5 passwords
- every web user had around 25 accounts protected by passwords
- typed roughly 8 passwords every day

Why OpenID Connect?

Ease the Burden of Password Management

- Password Manager
- Identity Management Systems supporting Single-Sign-On (SSO) (e.g OpenID, OAuth 2.0, OpenID Connect)

What is OpenID Connect?



wikiHow - How to do anythi ×

www.wikihow.com/Main-Page

We're trying to help everyone on the planet learn how to do anything. Join us.

Log in via

Log In

Username or Email

Password

☑ Remember me

Create an account Forgot password?

Wikihow login page

What is OpenID Connect?

Entities in OpenID Connect

- User Agent (UA), typically a web browser

What is OpenID Connect?

Entities in OpenID Connect

- User Agent (UA), typically a web browser
- OpenID Provider (OP), provides methods to authenticate an end user and generates assertions regarding the authentication event and the attributes of the end user

What is OpenID Connect?

Entities in OpenID Connect

- User Agent (UA), typically a web browser
- OpenID Provider (OP), provides methods to authenticate an end user and generates assertions regarding the authentication event and the attributes of the end user
- Relying Party (RP), provides protected on-line services and consumes the identity assertion generated by the OP

What is OpenID Connect?

Entities in OpenID Connect

- User Agent (UA), typically a web browser
- OpenID Provider (OP), provides methods to authenticate an end user and generates assertions regarding the authentication event and the attributes of the end user
- Relying Party (RP), provides protected on-line services and consumes the identity assertion generated by the OP
- End User (U), who accesses on-line services of the RP

What is OpenID Connect?

Tokens in OpenID Connect

- *code*, is an one-time opaque value, has limited validity period, RP can use *code* to exchange an *access_token* with OP
- *access_token*, has limited validity period, RP can use it to retrieve user attributes from OP
- *id_token*, contains claims about the authentication of an end user by an OP together with any other claims requested by the RP.

What is OpenID Connect?

Feature of OpenID Connect

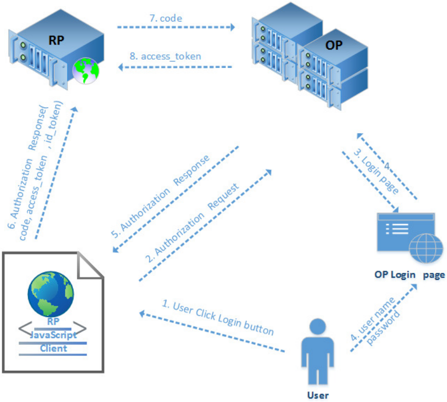
- builds on top of OAuth 2.0 (finalised in 2012)
- enable RPs to verify an end user identity

What is OpenID Connect?

Authentication Flows in OpenID Connect

- Hybrid Server-side Flow
- Authorization Code Flow
- Client-side Flow

What is OpenID Connect?



Google's Hybrid Server-side Flow

What is OpenID Connect?

Related Work

- Vladislav et. al [Vladislav, 2015] looked at the security of the OpenID Connect Discovery and Dynamic Registration extensions.

Our Contribution

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Our Work:

- We report on the first field study of the security properties of Google's implementation of OpenID Connect.

Our Contribution

Our Work:

- We report on the first field study of the security properties of Google's implementation of OpenID Connect.
- We examined the security of all 103 of the RPs supporting the Google Sign-in service from the GTMetrix list of the Top 1000 Sites.

Our Contribution

Our Work:

- We report on the first field study of the security properties of Google's implementation of OpenID Connect.
- We examined the security of all 103 of the RPs supporting the Google Sign-in service from the GTMetrix list of the Top 1000 Sites.
- We discovered a number of vulnerabilities which allow an attack to log in to the RP as a victim user, we reported our findings to the most serious affected websites and Google, and helped these RPs fix the identified problems.

Our Contribution

Our Work:

- We report on the first field study of the security properties of Google's implementation of OpenID Connect.
- We examined the security of all 103 of the RPs supporting the Google Sign-in service from the GTMetrix list of the Top 1000 Sites.
- We discovered a number of vulnerabilities which allow an attack to log in to the RP as a victim user, we reported our findings to the most serious affected websites and Google, and helped these RPs fix the identified problems.
- We propose practical improvements which can be adopted by OpenID Connect RPs and OPs that address the identified problems.

Adversary Model

Adversary Model:

- **A Web Attacker** can share malicious links or post comments containing malicious content (e.g. stylesheets or images) on a benign website; and/or exploit vulnerabilities in an RP website. Malicious content forged by a web attack might trigger the UA to send HTTP(S) requests to an RP and OP using GET or POST methods, or execute attacker JavaScripts. For example, a web attacker could operate an RP website to collect *access_tokens*.
- **A Passive Network Attacker** can intercept unencrypted data sent between an RP and a UA (e.g. by monitoring an open Wi-Fi network).

Study the Security of Google's OpenID Connect

Our Study:

- examined 103 RPs supporting Google sign-in
- 33 (32%) use the Hybrid Server-side Flow
- 69 (67%) adopt the Authorization Code Flow
- just 1 use the Client-side Flow.

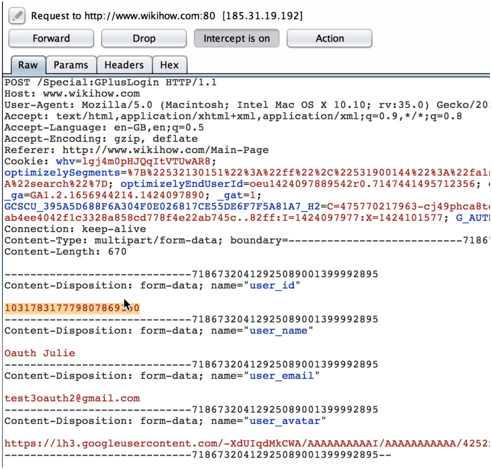
Studying the Hybrid Server-side Flow

Authentication by Google ID

- 6 RPs (out of 33) submit user's Google ID to their Google sign-in endpoint
- 3 RPs rely on Google ID as authentication
- Google ID value is public (e.g. https://plus.google.com/u/0/115722834054889887046/posts)

Authentication by Google ID

Wikihow

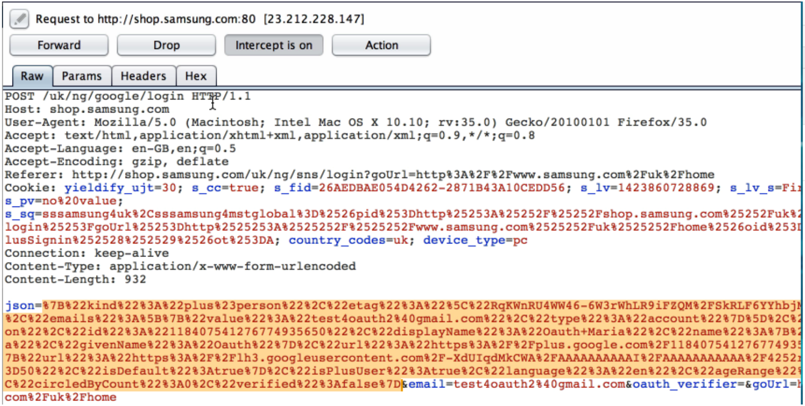


```
POST /Special:GPlusLogin HTTP/1.1
Host: www.wikihow.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:35.0) Gecko/20
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.wikihow.com/Main-Page
Cookie: whv=lgj4m0pHJQqItVTUwAR8;
optimizelySegments=%7B%22532130151%22%3A%22ff%22%2C%22531900144%22%3A%22fal
A%22search%22%7D; optimizelyEndUserId=oeu1424097889542r0.7147441495712356;
_ga=GA1.2.1656944214.1424097890; _gat=1;
GCSCU_395A5D688F6A304F0E026817CE55DE6F7F5A81A7_H2=C=475770217963-cj49phca8t
ab4ee4042f1c3328a858cd778f4e22ab745c..82ff:I=1424097977:X=1424101577; G_AUT
Connection: keep-alive
Content-Type: multipart/form-data; boundary=---------------------------7186
Content-Length: 670

-----------------------------718673204129250890013999992895
Content-Disposition: form-data; name="user_id"

103178317779807869150
-----------------------------718673204129250890013999992895
Content-Disposition: form-data; name="user_name"

Oauth Julie
-----------------------------718673204129250890013999992895
Content-Disposition: form-data; name="user_email"

test3oauth2@gmail.com
-----------------------------718673204129250890013999992895
Content-Disposition: form-data; name="user_avatar"

https://lh3.googleusercontent.com/-XdUIqdMkCWA/AAAAAAAAAAI/AAAAAAAAAAA/4252
-----------------------------718673204129250890013999992895--
```

Wikihow's Google sign-in endpoint

Authentication by Google ID

Samsung UK

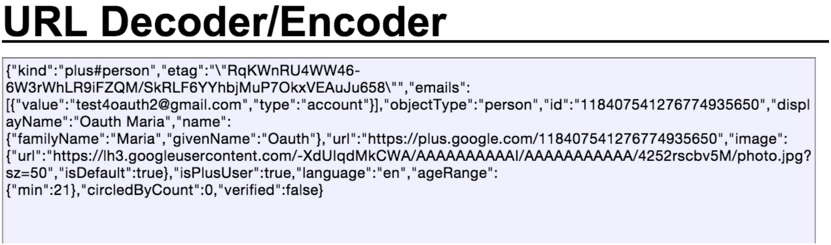


```
POST /uk/ng/google/login HTTP/1.1
Host: shop.samsung.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:35.0) Gecko/20100101 Firefox/35.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://shop.samsung.com/uk/ng/sns/login?goUrl=http%3A%2F%2Fwww.samsung.com%2Fuk%2Fhome
Cookie: yieldify_ujt=30; s_cc=true; s_fid=26AEDBAE054D4262-2871B43A10CEDD56; s_lv=1423860728869; s_lv_s=Fi
s_pv=no%20value;
s_sq=sssamsung4uk%2Csssamsung4mstglobal%3D%2526pid%253Dhttp%25253A%25252F%25252Fshop.samsung.com%25252Fuk%
login%25253FgoUrl%25253Dhttp%2525253A%2525252F%2525252Fwww.samsung.com%2525252Fuk%2525252Fhome%2526oid%253
lusSignin%252528%252529%2526ot%253DA; country_codes=uk; device_type=pc
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 932

json=%7B%22kind%22%3A%22plus%23person%22%2C%22etag%22%3A%22%5C%22RqKWnRU4WW46-6W3rWhLR9iFZQM%2FSkRLF6YYhbj
%2C%22emails%22%3A%5B%7B%22value%22%3A%22test4oauth2%40gmail.com%22%2C%22type%22%3A%22account%22%7D%5D%2C%
on%22%2C%22id%22%3A%22118407541276774935650%22%2C%22displayName%22%3A%22Oauth+Maria%22%2C%22name%22%3A%7B%
a%22%2C%22givenName%22%3A%22Oauth%22%7D%2C%22url%22%3A%22https%3A%2F%2Fplus.google.com%2F11840754127677493
7B%22url%22%3A%22https%3A%2F%2Flh3.googleusercontent.com%2F-XdUIqdMkCWA%2FAAAAAAAAAAI%2FAAAAAAAAAAA%2F4252
3D50%22%2C%22isDefault%22%3Atrue%7D%2C%22isPlusUser%22%3Atrue%2C%22language%22%3A%22en%22%2C%22ageRange%22
C%22circledByCount%22%3A0%2C%22verified%22%3Afalse%7D&email=test4oauth2%40gmail.com&oauth_verifier=&goUrl=
com%2Fuk%2Fhome
```

Samsung UK's Google sign-in endpoint

Authentication by Google ID

Samsung UK



URL Decoder/Encoder

```
{"kind":"plus#person","etag":"\"RqKWnRU4WW46-
6W3rWhLR9iFZQM/SkRLF6YYhbjMuP7OkxVEAuJu658\"","emails":
[{"value":"test4oauth2@gmail.com","type":"account"}],"objectType":"person","id":"118407541276774935650","displ
ayName":"Oauth Maria","name":
{"familyName":"Maria","givenName":"Oauth"},"url":"https://plus.google.com/118407541276774935650","image":
{"url":"https://lh3.googleusercontent.com/-XdUIqdMkCWA/AAAAAAAAAAI/AAAAAAAAAAA/4252rscbv5M/photo.jpg?
sz=50","isDefault":true},"isPlusUser":true,"language":"en","ageRange":
{"min":21},"circledByCount":0,"verified":false}
```

Decoded JSON value

Studying the Hybrid Server-side Flow

Using the Wrong Token

- *access_token* is a bearer token
- 58% of RPs (19 out of 33) using Hybrid Server-side Flow submit an *access_token*, back to their Google sign-in endpoint
- 45% (15 of these 19) use the *access_token* to authenticate the user
- 39% of the RPs (13 of 33) are vulnerable to impersonation attack.

Using the Wrong Token

Impersonation Attack

- an attacker is able to log in to the user account by submitting an *access_token* from other RP to the RPs who use use the *access_token* to authenticate the user
- 39% of the RPs (13 of 33) are vulnerable to impersonation attack.

Studying the Hybrid Server-side Flow

Intercepting an *access_token*

- 58% of RPs (19 out of 33) using Hybrid Server-side Flow submit an *access_token*, back to their Google sign-in endpoint
- 12% (4 of these 33) send the *access_token* unprotected

Intercepting an *access_token*

TheFreeDictonary

- SSL is enabled to protect its Google sign-in endpoint
- store *access_token* to the cookie
- homepage of TheFreeDictionary is not protected by SSL

Studying the Hybrid Server-side Flow

Privacy Issues

- the RPJC running on the users browser sends user information, the id token or the *access_token* back to its Google sign-in endpoint without SSL protection (4 out of 33)
- the RP Google sign-in endpoint sends the user information directly to the users browser without SSL protection (2 out of 33)
- the RP uses SSL to protect the link to the Google sign-in endpoint, but changes to http when sending user information back to the UA. (1 out of 33)
- user privacy cannot be guaranteed for 21% (7 out of 33) of the RPs using Hybrid Server-side Flow

Studying the Hybrid Server-side Flow

Session Swapping (Cross Site Request Forgery)

- The attacker first logs in to the RP website using his/her own account and intercepts the Google-generated tokens
- The attacker constructs a request to the RPs Google sign-in endpoint, including the attackers own tokens.
- The attacker inserts the request in an HTML document (e.g. in the src attribute of a img or iframe tag) made available via an HTTP server.
- The victim user is now, by some means, induced to visit the website offering the attackers page

Session Swapping

- 73% of RPs using Hybrid Server-side Flow (i.e. 24 of 33) are vulnerable.
- 8 submit a *code* to their Google sign-in endpoint
- 16 submit an *access_token* or the users Google ID to the Google sign-in endpoint,

Session Swapping

The Google Hybride Server-side Flow Sample Code

```
function signInCallback(authResult)
  if (authResult[’code’])
    $.ajax(
      type: ’POST’,
      url: ’http://example.com/storeauthcode’,
      contentType: ’application/octet-stream;charset=utf-8’,
      success: function(result)  ... ,
      processData: false,
      data: authResult[’code’]
    );
  ...
```

Studying the Authorization Code Flow

Intercepting an *access_token*

- a *code* is returned to the RP's Google sign-in endpoint
- 6% of their Google sign-in endpoints (4 out of 69) return an *access_token* to the users browser without SSL protection.

Studying the Authorization Code Flow

Stealing an *access_token* via Cross-site Scripting

- **automatic authorization granting**, generates an authorization response automatically if a user has a session with Google and previously granted permission for the RP concerned
- an attacker is able to steal a user *access_token* by exploiting an XSS vulnerability in the RP or UA

Studying the Authorization Code Flow

Privacy Issues

- no *access_token* and *id_token* are transmitted during authorisation
- user privacy cannot be guaranteed for 16% (11 out of 69) of the RPs using Authorization Code Flow

Studying the Authorization Code Flow

Cross Site Request Forgery

- 35% of the RPs using the Authorization Code Flow (24 out of 69) are vulnerable to session swapping attack
- an attacker can force a user log in on 35% of the RPs using the Authorization Code Flow via a CSRF attack

Security Concerns

Security Concerns over Google's implementation of OpenID Connect

- Giving RPs the Ability to Customise the Hybrid-Server-side Flow
- No CSRF Countermeasures in the Hybrid-Server-side Flow
- Automatic Authorization Granting

Recommendations

Recommendations for RPs

- Do not customise the Hybrid Server-side Flow
- Deploy countermeasures against CSRF attacks
- Do not use a constant or predictable *state* value

Recommendations

Recommendations for OPs

- Remove the *token* from the authorization request in the Hybrid Server Flow
- Add a *state* value to the sample code
- Allow the RP to specify the *state* value in the Hybrid Server Flow

Conclusion

Our Work:

- We report on the first field study of the security properties of Google's implementation of OpenID Connect.
- We examined the security of all 103 of the RPs supporting the Google OpenID Connect service from the GTMetrix list of the Top 1000 Sites.
- We discovered a number of vulnerabilities which allow an attack to log in to the RP as a victim user, we reported our findings to the most serious affected websites and Google, and helped these RPs fix the identified problems.
- We propose practical improvements which can be adopted by OpenID Connect RPs and OPs that address the identified problems.

Possible Future Work

Future Work

- Analyse the security of other SSO systems, e.g. Shibboleth, OAuth 2.0
- Look at other security issues faced by the OpenID Connect, e.g. phishing, DDOS.

References

Vladislav Mladenov and Christian Mainka and Julian Krautwald and Florian Feldmann and Jörg Schwenk
On the security of modern Single Sign-On Protocols: OpenID Connect 1.0
arXiv preprint arXiv:1508.04324, 2015

Dinei Florencio and Cormac Herley (2007)
A largea-scale study of web password habits
Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007

Thanks for listening

Thank you for listening!
Questions?