OpenID

Single Sign-On for the Internet: A Security Story

Eugene Tsyrklevich eugene@tsyrklevich.name Vlad Tsyrklevich vlad902@gmail.com

BlackHat USA, Las Vegas 2007

Introduction

With the explosion of Web 2.0 technology, the number of individual sites requiring registration has dramatically increased... and it is becoming apparent that the current authentication situation is unsustainable. To deal with the dozens of individual logins and passwords required by different sites, users are being forced to write down their logins or reuse the same username and password for every website. This is clearly undesirable as it creates multiple points of attacks and a single hacked site (the weakest link) has the potential to completely compromise a user's digital identity on the Internet today.

OpenID

OpenID is a Single Sign-On protocol that solves the problem of having an individual login and password for every web site. With OpenID, a user can register once with an Identity Provider (IdP) of their choice and then use that login on all OpenID-enabled sites. As OpenID is a decentralized system, a user can register with any identity provider. In the same way, any site can allow users to login using their OpenID login. This is in contrast with systems such as Microsoft Passport that are controlled by a single central entity.

An OpenID login is simply a URL such as <u>http://john.doe.name/</u> or <u>http://jonny.myopenid.com/</u>, that contains a set of HTML tags that identify a user's Identity Provider:

```
<html>
<head>
<link rel="openid.server" href="http://www.myopenid.com/server">
</head>
<body></body>
</html>
```

For example, in the case above, the Identity Provider (IdP) is <u>www.myopenid.com</u>.

When an Identity Provider successfully authenticates a user, the IdP makes a basic assertion that a user owns a given URL. Thus the identity provider answers the *who* question – e.g. is this user john.doe.name? The IdP does NOT deal with other questions such as the *what* (authorization) – e.g. is john.doe.name allowed to access a site?

Using OpenID

Figure 1 demonstrates a sample login on an OpenID-enabled web site (also called Relying Party).

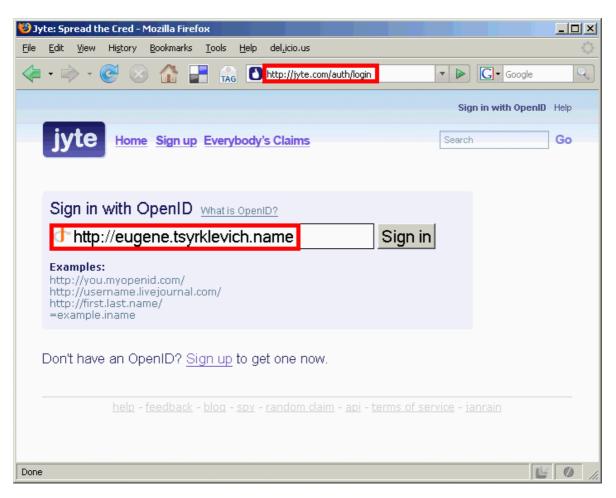


Figure 1: Logging in

After providing their login, a user is redirected to their identity provider for authentication – Figure 2.

🥹 Sign In - Mozilla Firefox	
Eile Edit View History Bookmarks Iools Help del _u icio.us	¢
< 🔹 🛸 < 🕑 🛞 🏠 📑 🚓 🖻 https://www.myopenid.com/signin_password?tid=92bcdb19	Gr Google
SIGN IN	YOUR PERSONAL ICON
	Home
About this page	Sign Up
If you have a MyOpenID account, sign in here. <u>Sign up</u> if you don't yet have an account.	Recover Account
I Notice	OpenID Site Directory
You must sign in to authenticate to http://jyte.com/ as as http://eugene.tsyrklevich.name/.	FreeYourlD.com
Help Feedback Privacy MyOpenID ^{tes} and	Biog About Us © 2007 JanRain, Inc. d the MyOpenID [™] website are trademarks of JanRain, Inc.
Dure	j www.myopenia.com 🔤 j 💽 j 💋

Figure 2: User Authentication

After a user enters their password, IdP verifies that the user is disclosing their information to the site that they originally thought they were logging into (Figure 3).

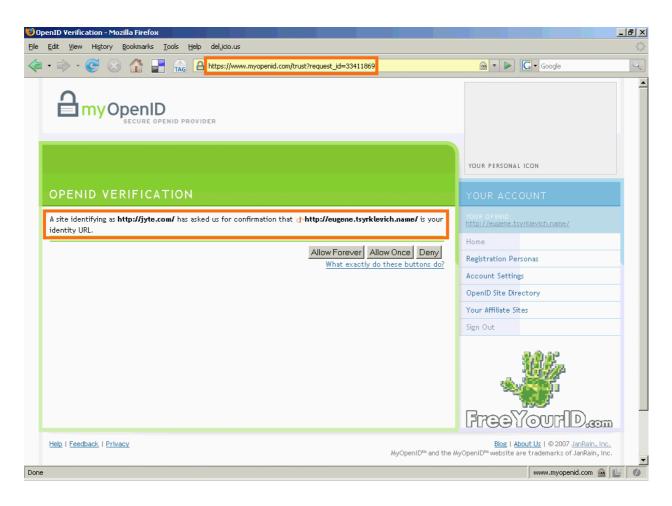


Figure 3: Relying Party Verification

Finally, the user is redirected back to the original site (Figure 4).

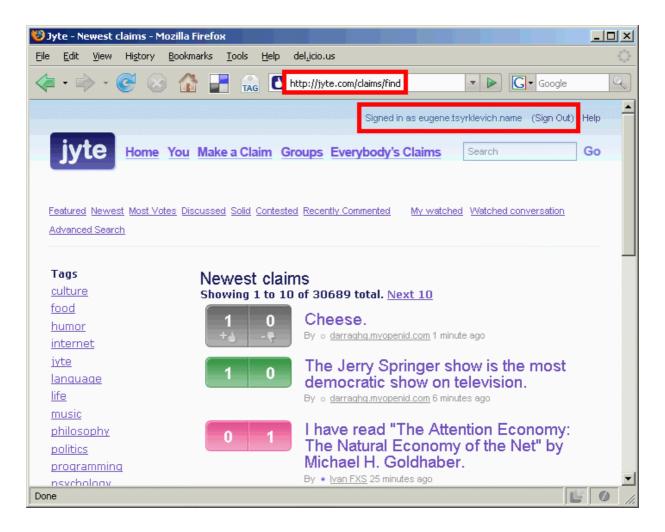


Figure 4: Logged in user

OpenID Protocol

Whilst the screenshots above might give you an impression that OpenID is a trivial protocol, there are a number of transactions that actually take place in the background.

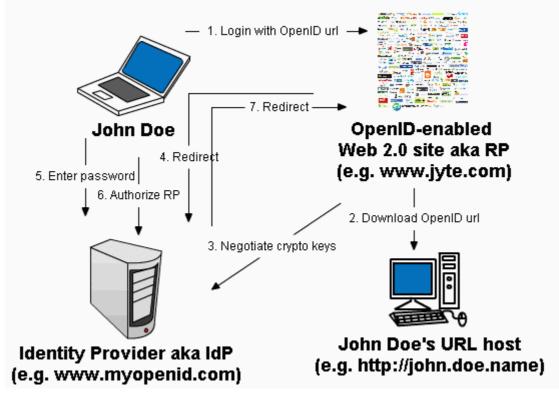


Figure 5: OpenID protocol

There are seven separate steps that have to take place in order to allow an OpenID user to login. Each step represents a security risk that is described in detail in the following sections.

Step 2: Downloading an OpenID URL

As previously mentioned, an OpenID login is simply a URL. Thus when a user logs into a site and provides their login, the site needs to download the URL and extract the Identity Provider address to continue with the protocol. As you can imagine, downloading data from arbitrary hosts on the Internet is extremely risky. Here are some of malicious URLs that any OpenID-enabled site needs to consider and to be able to protect against:

- <u>http://www.nsa.gov:1/, http://www.nsa.gov:2/, http://www.nsa.gov:3/,</u> ... It is trivial to specify an arbitrary port and cause the Relying Party to port scan any host on the Internet. Even if no harm is actually done, it is the OpenID site that seems to carry out the scan and not the malicious user.
- <u>https://192.168.1.15/internal/auth?ip=1.1.1.1</u>
 In addition to connecting to any host on the Internet, the Relying Party can be tricked into connecting to an arbitrary internal host, which an external attacker would not normally have access to. This can be exploited to access internal scripts, for example, that can carry out unauthorized actions.

• <u>http://localhost:8080/</u>

Similarly, the RP has to protect itself from a malicious user who might be trying to bypass the firewall by forcing the web server to connect to its firewalled off ports using the loopback interface.

- <u>http://www.youtube.com/largemovie.flv</u>
- <u>http://www.tarpit.com/cgi-bin/hang.pl</u> Relying Parties also have to protect themselves from Denial of Service attacks and limit the amount of data and time a single request is allowed to consume
- <u>file:///dev/null</u> Finally, an RP should limit any downloads to http or https protocols. Other protocols such as file, ftp, etc should be explicitly disallowed.

The above examples are not meant to be a comprehensive list. They merely point out some of the attacks that an OpenID-enabled web site needs to be able to protect itself against. As you can see, a simple task of downloading a user-controlled URL can be quite treacherous.

Step 3: Negotiating crypto keys

To guarantee the integrity of the exchanged data (provided by HMAC), an Identity Provider and a Relying party need to agree on a shared cryptographic key. To achieve this, an IdP and an RP use the Diffie-Hellman (DH) algorithm to come up with a shared symmetric key, to be used for a predetermined amount of time (see Figure 6).

POST /server HTTP/1.0 Host: www.myopenid.com Content-type: application/x-www-form-urlencoded Content-length: 504

```
openid.mode=associate&openid.assoc_type=HMAC-SHA1&
openid.session_type=DH-SHA1&
openid.dh_consumer_public=QVlovpfJ31QmJPmvHFs6gIHE63FliBDAPH%2F693U
k3arIaMvhVcLx4kQyQhy5G00OOJvVNwsVciFVxN0%2BiuKnFtmPn7gjgq3yjKzf5
NT5iDz35QZx1Z7WGwVgGRYiYyrNne1RgKJu3EvpMHDhL7KxwCyq%2B2gIsO
jwezBD8KVyuxM%3D&openid.dh_modulus=ANz5OguIOXLsDhmYmsWizjEOH
Tdxfo2Vcbt2I3MYZuYe91ouJ4mLBX%2BYkcLiemOcPym2CBRYHNOyyjmG0mg
3BVd9RcLn5S3IHHoXGHb1zqdLFEi%2F368Ygo79JRnxTkXjgmY0rx1J5bU1zIKaS
DuKdil%2BXUkKJX8Fvf8W8vsixYOr&openid.dh_gen=Ag%3D%3D
```

Figure 6: Diffie-Hellman exchange

Unfortunately, DH is vulnerable to a man-in-the-middle attack. To address this problem, the OpenID spec suggests that the above HTTP exchange should be executed over HTTPS to avoid any potential attacks. However if HTTPS is being used between the two parties, the DH key generation is no longer required, as a random string can be securely exchanged over the established SSL connection. With

HTTPS required to guarantee the security of the transaction, the Diffie-Hellman exchange represents an unnecessary complexity that can be eliminated.

Step 4: Relying Party to Identity Provider redirect

Once the IdP has been identified and the crypto keys have been exchanged, the web site (Relying Party) redirects the user to the Identity Provider for the actual authentication. The user is redirected to their IdP server using a simple HTTP redirect construct:

Location: http://www.myopenid.com/server? openid.assoc_handle=%7BHMAC-SHA1%7D%7B4..& openid.identity=http%3A%2F%john.doe.name%2F& openid.mode=checkid_setup& openid.return_to=http%3A%2F%2www.somesite.com%2F& openid.trust_root=http%3A%2F%2www.somesite.com%2F

Unfortunately, the IdP server address is specified by the web site itself. Thus a malicious Relying Party can easily redirect a user to a malicious, yet identically looking, provider that can be used to steal a user's password. Today, "phishing" is the most well known attack against OpenID protocol and remains unsolved.

It should also be noted that the same attack can be carried out by a malicious URL host without the Relying Party realizing that such an attack was even taking place. For example, a malicious <u>http://john.doe.name/</u> host could return

<link rel="openid.server" href="http://www.myprecious.com/server">

instead of

<link rel="openid.server" href="http://www.myopenid.com/server">

causing the Relying Party to send the user to the wrong IdP! The end result would be the same – a user's identity would be stolen, potentially without them even realizing that they had just been phish'ed.

Steps 5 & 6: Identity Provider authentication

Once the user is redirected to the Identity Provider (IdP) server, they log in and authorize the Relying Party. Once they are logged into one site, they can be automatically logged into other sites. Whilst of great convenience to the user, the IdP becomes a central clearing place for all user logins. Thus a malicious IdP can easily spy on a user's activity on the Internet. Apart from using multiple OpenID logins, there is little that a user can do today to avoid disclosing their surfing habits, when using OpenID enabled sites, to their Identity Provider.

Step 7: Identity Provider to Relying Party redirect

Once a user is authenticated, the Identity Provider redirects them back to the original web site:

```
Location: http://www.somesite.com/finish_auth.php?
openid.assoc_handle=%7BHMAC-HA1%7D%7B47bb..&
openid.identity=http%3A%2F%2Fjohn.doe.name%2F&
openid.mode=id_res&
openid.return_to=http%3A%2F%2www.somesite.com&
openid.sig=vbUyND6n39Ss8IkpKl19RT83O%2F4%3D&
openid.signed=mode%2Cidentity%2Creturn_to&
nonce=wVso75KH
```

The problem with this redirect is the fact that anyone who can obtain this URL (e.g. by sniffing the wire) can replay it and get logged into the site as the victim user. Some of the Identity Providers use nonces (number used once) to allow a user to log into the site once and fail all the consecutive attempts. The nonce solution works if the user is the first one to use the URL. However a fast attacker who is sniffing the wire can obtain the URL and immediately reset a user's TCP connection (as an attacker is sniffing the wire and knows the required TCP sequence numbers) and then execute the replay attack as described above. Thus nonces only protect against passive attackers but cannot prevent active attackers from executing the replay attack.

Other attacks

Once a user is logged in, there are a number of other attacks that can be executed against the user. One of the attacks is a cross-site request forgery:

```
<html><body>
<iframe id="login"
src="http://bank.com/login?openid_url=john.doe.name" width="0"
height="0"></iframe>
```

```
<iframe id="transfer"
src="http://bank.com/transfer_money?amount=100&to=attacker"
width="0" height="0"></iframe>
</body></html>
```

The above HTML code contains 2 hidden iframes that silently log a user into their banking site (assuming the site is trusted and a user has already logged into another OpenID site) and then transfer the money to an attacker. The above attack succeeds because the Identity Provider, and not the Relying Party, dictates the user login security policy. Thus OpenID Relying Parties have little say over how users should login because they do not control their login security policy.

Security Benefits

Besides the obvious benefit of having a single login and password that do not need to be written down anywhere to be remembered, OpenID also has another important security advantage. As the majority of users will only use one OpenID login, the authentication of that login can be made extremely secure. At present, it is expensive for individual web sites to provide their users with extra security features such as client-side certificates, smartcards or SecurIDs. However, with OpenID an Identity Provider might be able to afford to put the time and money into securing a single "front gate" instead of having dozens of sites securing their individual "front gates". This has the potential to greatly increase the security of our everyday logins.

Conclusion

Whilst this paper has presented a number of attacks against OpenID, it still remains the only viable option for the Internet-wide SSO system. Some of the attacks presented are either partially solved already or can be solved with relative ease. Other attacks such phishing and the redirect attack require further thought. However, it is our belief that OpenID can be made secure.