OpenID analysis. Can we trust software found in the street?

ROGER LARSEN
Essay - IMT4581 Network Security
Gjøvik University College 2011

May 18, 2011

Abstract

Open source software have existed as long as computer technology. The last 10 years it has really exploded in many successfully projects. OpenID is one of these successful stories of open source software. OpenID is a single sign-on (SSO) system that due to date have over billions of accounts enabled and thousands of web-sites accepting it [1]. The dilemma may be open source software and security solutions. Is this a good combination? Can we build security systems on open source software? With some precaution - open source software have in general no limitations regarding security.

I start this article describing the evolution of web-pages concerning users interaction. Then I describe OpenID more technical and analyse some security issues. Further I discuss open versus closed source software. Finally I conclude my article and raise some question for further discussion.

1 Introduction

1.1 The Challenge

Internet have evolved exponential the last decade. We have changed from small band to broad band in our needs for interactive multimedia Internet service. Static web-pages produces by companies are changed to web-pages where we all program/interact with our social networks activity. Online users are producing comments/links/evaluation/”Likes”/”Twitters” et.al. every second on web-pages (articles/blogs/newspapers).

Many professionals in computer industry use the term ”**Web 2.0**” to describe this new way web-pages makes us interact (read more on this on `oreilly.com` [12]. To give You an example of the number of the popular social network; the recent web-page cited regarding ”Web 2.0” gave us 52 possible so-called ”**ShareThis**” choices. Figure 1 shows some very common buttons found on most web-pages.

Figure 1: ShareThis typical buttons.

In the struggle for users statistics and -interaction; the challenge for content publishers was obviously... they needed to offer users personal login on web-pages. Soon most every web-pages offered user to register for an user account which forced users to give some personal information. This way the content publishers had much statistics and got interactions (comments/links e.g) from many of their users.

Internet browsers often have default installation with the options ”Save Passwords?” and ”AutoFill” mechanisms active. These mechanisms gives You the chance to store every UID and password for your web-pages which are seemingly very convenient. This behaviour of the Internet browser from practical and security point of view are as follows:

[1] Numbers found in OpenID Fundation’s introduction document [11]

- The Internet browsers have obviously to save authentication information on the local computer. In security thinking this is a unwanted behaviour of the Internet browser.
- If the user uses an other computer s/he have to log in for every new web-page s/he are visiting.
- Users tends to get very sloppy regarding his/hers complexity of UID's and password's when s/he gets use to these mechanisms.
- A computer that are used by several users in the same local computer account (very common installation mode in Microsoft Windows) gives all the chance to use each others user accounts on web-pages.
- A re-installation and/or cleanup process of the computer's operating system often removes the actual saved UID's and belonging passwords.

The number of social networks on Internet gave births to new web-pages almost every day [10]. This interactivity between publisher and reader of content (articles/blogs/newspapers) demanded new technique in authentication . Soon users had a lot of users accounts/UID and passwords to manage. This made often UID and passwords being reused on several web-pages with poor passwords (easier to remember and bad security), but in worst case the users did not register. The need for a efficiently authentication system was long overdue. This was solved by Single Sign-On (SSO) solutions suck as OpenID.

1.2 Limitation

In order to match the scope of this essay I can not analyse the software OpenID in details, of course. I have structured known security vulnerabilities and discuss weakness in OpenID's basic functions. The analysis mentioned in the title are not a scientific approach to analysis, but more a collection of information from good articles, technical reports and master thesis's.

2 OpenID - an Open source Single Sign-On solution

OpenID is a open source community driven for SSO solution. Almost every large company running social networks have embraced OpenID (e.g. Amazon, Aol, Facebook, Flickr, Google, Microsoft (Myspace) and Yahoo (in alphabetic order)).

2.1 Brief History

Here is a brief timeline of milestones in OpenID's history:

- In May 2005 OpenID was invented by Brad Fitzpatrick at Six Apart Ltd. (US) [18].
- In June 2007 **OpenID Foundation** was formally established. The OpenID Foundation is an international non-profit organization of individuals and companies that supports and developes OpenID framework.
- In December 2007 the last final version (OpenID 2.0) was released.

Since 2005 this SSO system have become very popular - due to date there are over billions of accounts enabled and thousands of web-sites accepting it. Source: OpenID Fundation. [11].

2.2 Technical description of OpenID

OpenID is a decentralised/user-centric SSO solution using several protocols. This includes among others Yadis [22], Diffie-Hellman [16], HTTP/HTTPS [15].

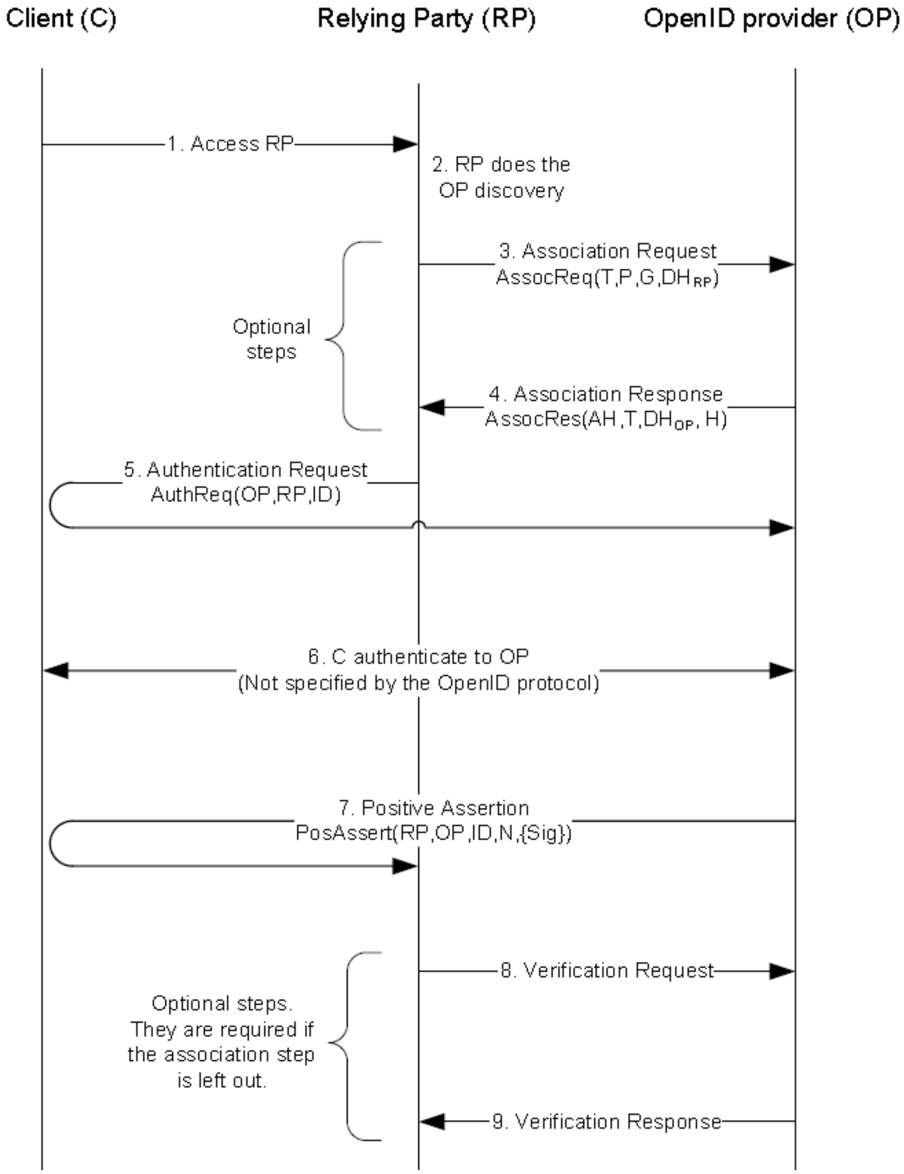
Client (C)
Relying Party (RP)
OpenID provider (OP)
1. Access RP
2. RP does the OP discovery
3. Association Request AssocReq(T,P,G,DH_RP)
Optional steps
4. Association Response AssocRes(AH,T,DH_OP, H)
5. Authentication Request AuthReq(OP,RP,ID)
6. C authenticate to OP (Not specified by the OpenID protocol)
7. Positive Assertion PosAssert(RP,OP,ID,N,{Sig})
8. Verification Request
Optional steps. They are required if the association step is left out.
9. Verification Response

Figure 2: OpenID Protocol description [9].

2.2.1 Illustration of OpenID process

.

2.2.2 Abbreviations used

T = Type of algorithm, **P** = Modulus Prime number, **G** = Diffie-Hellman Generator , $\mathbf{DH}_{RP}$ = The RP's Public Key, **AH** = Identifier of the association, $\mathbf{DH}_{OP}$ = the OpenID Provider's public key, **H** = the hash value of the created key XOR with the MAC key to transport the secret MAC key encrypted, **OP** = OpenID Provider, **ID** = the user's identity, **N** = a random value (nonce), **{Sig}** = a generated signature by hashing the concatenation of the RP, OP, ID, N and the key K, **K** = a session key between RP and OP.

2.2.3 Description of OpenID login process

1. A user access a website offering OpenID login (Relying Party (RP)).

2. The RP performs a *Discovery Process* to find a proper OpenID Provider (OP). *Yadis* protocol

are used in this process [22].

3. (Optional step) The RP performs a *Association Request* to the OP.
4. (Optional step) The OP answer with a *Association Response* to the RP.
5. The RP *redirects* the users session to the OP for an *Authentication Request.*
6. The user *authenticates* using his/her credentials in interaction with the OP.
7. The user are redirected to the RP (initial website) and its services that demanded OpenID login (assumed that the authentication was a success).
8. (Optional step) The RP sends a *Verification Request* to the OP (if performed: a verification on the outcome of previously steps).
9. (Optional step) The OP answers with a *Verification Response* to the RP.

Note! One of the steps 3-4 or 8-9 must be done! The association steps are done to generate the session key **K**.

2.3 OpenID Security Issues

The current OpenID version 2.0 still have some security weakness.

2.3.1 Phishing

OpenID login process are highly vulnerable for phishing attacks. There are two basic kinds of phishing attacks: (i) a malicious/compromised RP can forward the user to a OP that is controlled by the same attacker, (ii) the URL from the user can be malicious/compromised. This makes (again) the RP to forward the user to a malicious/compromised OP.

Countermeasure regarding phishing attacks are in general (i) the use of strong authentication and (ii) use trusted OP's.

2.3.2 Other Security Issues

Here is a list of challenges using OpenID [9], [2], [6], [19], [1], [20].

- The initial challenge may be how strict the installation of the OpenID is done. OpenID is very flexible and may be "sloppy" installed in contrast to high security.
- The OpenID is user centric and an compromised computer can hijack the OpenID identity.
- The user should have the chance to choose among OpenID Providers when the need for authentication arise (e.g. the RP must have several OP's to offer). This will also help the user to be logged on using OpenID even if some OP's are down/inaccessible.
- The user should be able to chose between OpenID and the web-page's own registration.
- The user should have the chance to connect the OpenID to federal SSO solutions to complete proper identification.
- OpenID have a Privacy Problem. The system is leaking OpenID user identities to third parties (for advertisement and traffic analysis). This is not a bug, but the design [20].

2.4 Alternatives to OpenID

There are several alternatives to OpenID. In table 1 the most common SSO solutions are compared [2].

Protocol	**Description**
Electronic ID	A X.509 certificate based authentication and authorization handler that solve the reliable end-user problem. **Pro.** = Based on the well scrutinized X.509 certificate standard. Widely used. Supports Reliable end-user identification. **Con.** Expensive **Intended used** = X.509 certificate based identity with a java based client side web application for authenticaion.
Kerberos	Computer network authentication protocol mainly developed in MIT in the 1980's. **Pro.** = Widely used and well scrutinized. **Cons.** = Requires synchronisation of clocks between involved computers/hosts. **Intended used** = In Local Area Networks (LAN's internal networks) [7].
Microsoft Live ID	Microsoft's own SSO alternative. **Pro.** = Widely used in Microsoft systems, Web based infrastructure. **Cons.** = Bad reputation regarding security vulnerability. **Intended used** = As a SSO solution.
OpenID	Open source SSO solution. Focus mainly on solving SSO. **Pro.** = Easy to learn. Web based infrastructure, through XML, and HTTP. **Con.** = Relatively new. **Intended used** = Web based single sign-on for web-services [5].
SAML	Security Assertion Markup Language (SAML) in a open standard OSS solution from Organization for the Advancement of Structured Information Standards (OASIS), U.S. . SAML is encoded/written in XML. **Pro.** = Widely used. Web based infrastructure, through XML, SOAP and HTTP. Supports Federated identities. **Con.** = Complex. **Intended used** = Exchanging authentication and authorization data between business partners. [4] [3]

Table 1: Single Sign-On competitors to OpenID (alphabetic order).

3 Open versus Closed Source Software

Can we trust software found in the street?
Stephen Wolthusen [21].

I have quoted professor Stephen Wolthusen in my title of this essay [2]. This quote rise an important question. Especially in security software we must have control. Open source software plays a major role in software development all over the world nowadays.

3.1 From Add-On's to Main Business Focus

Software was in early days of computer industry some *"add-on"* we got in additional to the very expensive computer equipment we bought. We just had to adapt to the often not so logic way the software worked.

The last decade we have seen a that software is the main focus for many companies. An common business model is that the software is initial free, but all service and support (especially service level agreements (SLA's)) costs a lot of money. Even large software companies with traditionally very closed code development have joined the open source approach (e.g. Microsoft, Oracle and Sun).

[2] Lecture Notes from Stephen Wolthusen, IMT4651-Applied Information Security, Gjøvik University College, 2010

3.2 Human approach to Free Software

We all enjoy freebies. This is a basic human characteristic.

I have in several occasions found myself wondering after having installed free software. Was I a bit to relaxed regarding security issues here? Am I getting to exited by the features and just want to try it out? Where was my normal paranoid and sceptical way of thinking this moment? Was all my experience and studying of computer security wasted?

In computer security the biggest treat are us humans. This is mainly because; (i) we believe we do the right action, but are either too self-confident or lack proper training, (ii) we get stressed and starts to cut corners in our struggle to manage our goals, (iii) we have for some reason a hidden agenda for our actions and misuses our trust et.al.

3.3 Common Challenges

We love open software!
Jean Paoli, General Manager, Interoperability Strategy, Microsoft [8].

Linux is a "cancer" that are threatening Microsoft's intellectual property.
Steve Ballmer, CEO, Microsoft [8].

The Open Source Software (OSS) versus Closed Source Software (CSS) is a never ending story. I will in this section discuss the common challenges between OSS and CSS. Software licensing is a huge topic, I focus here only on the general differences between OSS and CSS. I am well known of the fact that this may be to much of a simplification so please bare over with me here.

3.3.1 Price as main focus in the short run?

The price of implementing new software systems have two sides; the short run and the long run. OSS v. CSS price is difficult to measure. We often use more time in initial phase with OSS, but then we do not pay for license fee or support fee (as for CSS). But - time is money! If the project find a good match with ready made CSS - the CSS wins. In larger projects the local adaptation/adjustment are easier with the use of all knowledge in the OSS community. *Conclusion: Even race!*

3.3.2 Price as main focus in the long run?

The price of using OSS v. CSS are depending on the following key topics: (i) the amount of users, (ii) the stability of the software, (iii) the lifetime of this actual software. The chance of paying a lot of money in licensing fee is in general high. Stability may be a definition and can often be controlled. Lifetime is a difficult parameter, new technology/new leaders/laws and politics may remove fully working software in short time. *Conclusion: Credit to OSS!*

3.3.3 I need help! HelpDesk, Support and Documentation?

CSS demands licensing fee and helpdesk/ support fee - but this is normally very professional. OSS have the community power and often many more users that are willing to help in every minute of the day for free. The OSS community will often have a broader knowledge because of the large amount of users but we can not call anyone and we do need some understanding of the challenge/situation when we ask the community. Nevertheless, we can always read the FAQ pages in the OSS community to help us with the trivial questions. Documentation especially are often much better on CSS, but all the Quality Assurance (QA) in private companies often delay this delivery. *Conclusion = Even race!*

3.3.4 What do You actually Buy?

CSS have very much legal/juridical text attached to the usage of this CSS. How many of You have reads the legal/juridical text we have to accept before using the software? OSS are owned by us/none! *Conclusion: Credit to OSS!*

3.3.5 Can we trust OSS?

There is no guaranties in either open or closed source software regarding errors and/or vulnerabilities. People make mistakes! An empirical research regarding vulnerabilities and vendors patching behavior showed no significantly difference between the two categories [17]. This can also be explained with the number large of eyeballs/users in the OSS communities compared to the QA processes and heavy bureaucracy/procedures in CSS companies. *Conclusion: Even race!*

3.3.6 Software Quality Assurance

CSS lead on regarding QA. In general CSS companies use a lot of money to test and retest their software before it enters the marked. This is in general a good policy - but high QA slows down the version finalisation process. OSS utilise their large number of community users/members and can archive a high level of quality in this way. There are several OSS QA projects. Two popular examples are: OSSTMM - Open Source Security Testing Methodology Manual from the Institute for Security and Open Methodologies (ISECOM). Est. Jan 2001[13].
QualOSS - Second generation software quality assessment model. This is an European Project. The first model was finished in Sept 2007 [14]. *Conclusion: Credit to CSS!*

4 Conclusion

The perfect is the enemy of the good!
Voltaire.

4.1 OpenID

OpenID is very good example of the power of OSS community development. OpenID 2.0 have still some unwanted security issues but is in general a very flexible and powerful SSO solution. We are indeed looking forward for the next release 3.0.

4.2 Further work/discussions

Who owns my user ID/OpenID? Who can assure me that my OpenID are properly deleted? How is my privacy protected with the huge amount of systems involved?

4.3 OSS versus CSS?

We may have asked the wrong question. The question should have been asked in a more modern way: How can we ensure our software needs? I actually asked OpenID Foundation regarding their QA policy/methods. I got an prompt answer with a new email address to contact. The latter gave me no answer despite two reminders. OSS projects have a job to do regarding information and QA methods/policy. CSS are in general more available and service focused (often for some money).

If our initial choice are between OSS or CSS - we are excluding to much! If we start with a detailed requirement specification, a clear time and financial plan for our needs - we MUST seek both OSS and CSS to get what's most appropriate in the actual case. To exclude OSS may be a great failure. My conclusion is; Yes, we can trust OSS software with a good plan including QA methods!

References

[1] David Chadwick and University of Edinburgh Sandy Shaw. Openid study. URL, November 2008.

[2] Max Charas. An overview of openid from a security perspective. Master's thesis in computer science, Royal Institute of Technology, Stockholm, Sweden, Royal Institute of Technology School of Computer Science and Communication KTH CSC SE-100 44 Stockholm, Sweden, 2009.

[3] Robin Cover. The SAML Cover Pages. http://xml.coverpages.org/saml.html URL, 2000.

[4] Robin Cover. The XML Cover Pages. http://www.oasis-open.org/cover/xml.html URL, 2000.

[5] Encyclopaedia Britannica. Open source. http://www.britannica.com/EBchecked/topic/1017825/open-source URL, May 2011.

[6] Sebastian Feld and Norbert Pohlmann. Security analysis of openid, followed by a reference implementation of an npa-based openid provider. In Norbert Pohlmann, Helmut Reimer, and Wolfgang Schneider, editors, *ISSE 2010 Securing Electronic Business Processes*, pages 13–25. Vieweg+Teubner, 2011.

[7] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC 1510 (Proposed Standard), September 1993. Obsoleted by RFC 4120.

[8] Jon Brodkin. NetworkWorld. PCWorld Communications Inc. Microsoft: We love open source. http://www.pcworld.com/businesscenter/article/203923/microsoft_we_love_open_source.html URL, August 2010.

[9] Alexander Lindholm. Security evaluation of the openid protocol. Master's thesis, Royal Institute of Technology, Stockholm, Sweden, Royal Institute of Technology School of Computer Science and Communication KTH CSC SE-100 44 Stockholm, Sweden, 2009.

[10] Daniel Nations. What is social networking? http://webtrends.about.com/od/socialnetworking/a/social-network.htm URL, April 2011.

[11] OpenID Fundation. Introduction to OIDF Presentation. http://openid.net/wordpress-content/uploads/2011/03/Introduction-to-OpenID-Foundation-March-2011.pdf URL, March 2011.

[12] Tim O'Reilly. What is web 2.0. http://oreilly.com/web2/archive/what-is-web-20.html URL, March 2005.

[13] Pete Herzog. The Institute for Security and Open Methodologies (ISECOM). Osstmm - open source security testing methodology manual. http://www.isecom.org/osstmm/ URL, May 2011.

[14] QUALOSS. Qualoss summary. http://www.qualoss.org/about/summary/qualoss-summary URL, May 2011.

[15] E. Rescorla. HTTP Over TLS. RFC 2818 (Informational), May 2000. Updated by RFC 5785.

[16] RSA Labratories. EMC Corp. What is diffie-hellman? http://www.rsa.com/rsalabs/node.asp?id=2248 URL, May 2011.

[17] G. Schryen and E. Rich. Increasing software security through open source or closed source development? empirics suggest that we have asked the wrong question. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1 –10, January 2010.

[18] Six Apart Ltd. OpenID. http://www.sixapart.com/labs/openid/ URL, May 2011.

[19] Pavol Sovis, Florian Kohlar, and Jorg Schwenk. Security analysis of openid. In *Sicherheit 10*, pages 329–340, 2010.

[20] Uruena, Manuel and Busquiel, Christian. Analysis of a privacy vulnerability in the openid authentication protocol. IEEE Multimedia Communications, Services and Security (MCSS2010). http://www.it.uc3m.es/muruenya/papers/MCSS10OpenID.pdf URL, March 2010.

[21] Stephen Wolthusen. Lecture notes. applied information security. fall 2010, November 2010. Lecture Notes. Applied Information Security. Fall 2010. Gjoevik University College, Norway.

[22] Yadis.org. Yadis protocol. http://yadis.org/wiki/Main_Page URL, May 2011.