Microsoft .NET Passport, a solution of single sign on

Zheng Liu

Department of Computer Science
University of Auckland

zliu025@ec.auckland.ac.nz

Abstract:
As the World Wide Web grows rapidly, accessing web-based services or resources has become a part of people's everyday life. But most of these services or resources will require an authentication. It follows that people need to memorize many sets of user name and password. As a result of that it appears that there is a high demand of single sign on. Microsoft .Net Passport is one of the single sign on solutions. In this paper, it is aimed to describe the architecture of .Net Passport, and to discuss the security related appreciations and criticisms of this product from perspective of single sign on.

1. Introduction

.Net Passport is a single sign on services provided by Microsoft. To be more specific, it provides the single sign on solution to the companies offering web-based services and resources to users. The main objective of .NET Passport is to allow a user to have only one credential, and present his credential only once in order to access the services and resources from many different web sites. It also aims to provide the easy implementation of the services from company's aspect. As said by Microsoft, "Microsoft's goal is to make implementing Passport as easy as possible" [3]. And also it provides the great usability from the user's aspect, since .NET Passport using the common web technologies such as http, ssl, cookies, etc [1]. Those technologies are supported by almost all of web browsers. So there is no need for extra software, and to use the sign in services there are only few click on the web age.

In this page, we will only focus on the details of architecture and security issues of .NET Passport regarding to single sign on.

2. Overview of the architecture of .NET Passport

Towards single sign on, facilities provided by .NET Passport includes authentication authority, credential database, credential management, user registration authority and the user interfaces of signing in and registration.

The authentication authority, credential database, credential management and user registration authority reside on www.passport.com. And user interfaces of signing in and registration reside on www.passport.net. (Two domains are indicated by Microsoft [3])

To achieve the objective, .NET Passport first forms a federation among all the web sites that want to receive the service. The purpose of such federation is to build an environment. This federation is called passport federation by Jan De Clercq [2]. Within this federation, all the web sites using .NET Passport will use the same format of credentials for their users, and also use the same authentication protocol. With such an environment as the basis, .Net Passport then implements the “simple single sign on architecture” defined by Jan De Clercq [2] to perform the single sign on tasks.

Before going to the detail, let’s declare some terminologies that will be used in this paper:

The passport authentication authority, passport credential database, passport credential management, passport registration authority refer to the authentication authority, credential database, credential management, user registration authority that are provided by .NET Passport.

Participating web site refer to the web site operated by the company using .NET Passport. And the network formed by these participating web sites will be referred to as passport network.

Now we will go to detail of the architecture of .NET Passport from two aspects, Passport federation (in section 3) and implementation of “simple single sign on architecture” in .NET Passport (in section 4).

3. Passport federation in .NET Passport

Passport federation can be seen as an approach to extend the single sign on scope to be able to cover different companies. Since .NET Passport intends to provide the single sign on ability among all the participating web sites and they are operated by different companies. By using passport federation, all those web sites can be merged into a virtual organization. The expression “virtual organization” has been used for the reason that is there is no such an organization, but by establishing the trust relationship between passport authentication authority and the participating web sites, all participating web sites agree to obtain the same format of credential and authentication protocol.

Within Passport federation, trust relationships between authentication authorities are essential. With these trust relationships, the identity of certain user will be accepted by all the participating web sites once that identity has been authenticated with the passport authentication authority.

.NET Passport implements passport federation by using Service Agreement and

Passport User ID.

Service Agreement is a precondition for participating web sites. Once they want to use .NET Passport, they need to sign up this agreement with .NET Passport. With the signing up the agreement, a trust relationship is established between the participation web sites and the passport authentication authority. Also the Service Agreement unifies the format of the credentials and the authentication protocols among the participating web sites.

Format of the credential is the user name and password pair where user name is an email address and the password is a string consisted of at least 6 characters. The authentication protocol used is that all participating web site delegate the authentication processes to the passport authentication authority, and passport authentication authority will inform the user's identity to that participating web site by using Passport User ID.

Passport User ID (PUID) is a unique 64-bit number associated with each user, it can be seen as the identity of a user. And this identity will be agreed upon by all the participating web sites. In additional, it can also be used in the authorization tasks of user within the participating web sites.

Assignment of PUID is performed by passport registration authority while registering users. But users are usually registered from a participating web site. So it is necessary to anatomize the registration process within the passport federation.

3.1. Passport User ID assignment

When users want to register a participating web site, there are two choices that participating web sites can choose to register them. The first choice is to register the user only with that participating web site. By doing so, that participating web site needs its own facilities to register and authenticate users, and more importantly, the users are not registered with passport registration authority. So there are no PUIDs associated with them, then after they are authenticated by this participating web site, other participating web sites will not accept the identities of those users. The second choice is to ask the user to register through the passport registration authority. Since choosing first choice will result in the loss of the ability of single sign on, we will only focus on the detail of the second choice.

When a participating web site wants to register with it through passport registration authority, the processes are as the following: The participation web site will redirect the user to a page hosted on the participating web site, but the passport registration page is embedded within it. And the user will be asked to enter an e-mail address as user name and a password into the embedded passport registration page. Once the user pressed submit button, the e-mail address and password and the information about the participating web site are passed into passport registration authority. On the reception of the e-mail address and password, passport registration authority will first store them into the passport credential database, and generate the PUID for the user. The mapping between this credential and the PUID will be stored as well. After these done, passport registration authority will send the PUID back to that participating web site using the information of that participating web site that passed in along with the e-mail address and password. Then the participating web site stores the returned PUID to its own database for the use of authorization tasks.

But at the time when a participating web site starting to use the .NET Passport, it may already have many users registered with it. In this case, a participating web site needs to ask those existing user to register with passport registration authority in order to obtain a PUID associated. And later on those users can use the new credential to sign in.

4. “Simple single sign on architecture” [2] in .NET Passport

Since the passport federation unifies the credential format and the authentication protocol used by different participating web site, it brings the simplicity on the phase of implementing the single sign on architecture. In fact, .NET Passport implements the “Simple single sign on architecture” [2] on top of the passport federation. First, we will go over the blueprint of such architecture before looking at the actual implementation in.NET Passport.

4.1. “Simple single sign on architecture”

“Simple single sign on architecture” is defined by Jan De Clercq [2] as shown in figure 1, and this architecture consists of two main components. The first component is the authentication authority. The second component is a set of resource server. There is a trust relationship between the authentication authority and each resource server.

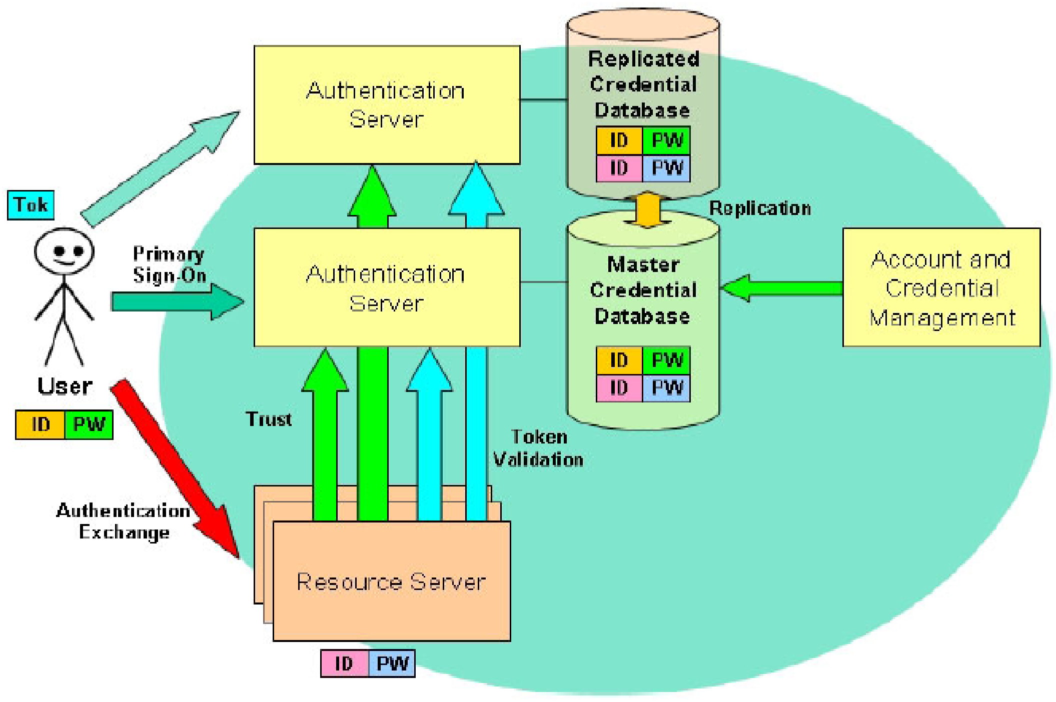

Figure 1 Simple single sign on architecture (after [2])

Within this architecture, when users want to access the resources hosted on the resource servers, they need to authenticate themselves with the authentication authority first by sending their credential to the authentication authority. On the reception of the credential, the authentication authority looks up its credential database. If there is a match between the credential from the user and the one in its credential database, the user’s identity is successfully authenticated. After this, a token is generated by the authentication authority, and the content of the token will be encrypted. Authentication authority returns the token to that user and it will be stored on the user’s machine. Afterwards, users present the token obtained from authentication authority to that resource server as the proof of the authentication. Then resource server will ask the authentication authority to validate this token, if authentication authority confirms that the token is valid, the users’ identities will be accepted by the resource server.

Next we will probe into the details of the implementation of it in .NET Passport.

4.2 Implementation of “Simple single sign on architecture” in .NET Passport

Within the implementation of “Simple single sign on architecture” [2], .NET Passport keeps the two main components, authentication authority and the set of resource servers. And the passport authentication authority is assigned the role of the authentication authority of the architecture, and the web servers of the participating web sites are assigned the role of the resource server.

Credentials in the implementation are an e-mail address and password pair as mentioned before. All the credentials are stored in the passport credential database. On the participating web sites, there are no credentials stored (without considering the exception that extra authentication authority is deployed on the participating web site).

The Tokens in the implementation are two cookies for each user. One is issued by

passport authentication authority, and it is encrypted that include information about the corresponding user's authentication, such as the time authenticated.
Another one is issued by the web server of participating web site.

Each participating authentication authority needs to establish a trust relationship based on the passport federation. With the trust relationship, a secret key is allocated between each participating web site and passport authentication authority. It is used to encrypt the messages between them to enhance the security.

Also in .NET Passport, the implementation has a difference to the blueprint of "Simple single sign on architecture" [2]. That is the process of token validation. Since the token consists of two parts, one is the cookie issued by participating web site and one is issued by passport authentication authority. So the token validation actually is carried out by both participating web site and authentication authority. And the details will be cover in later section.

Let's first explore the functionalities of passport authentication authority, and the web servers of the participating web sites regarding the roles they assigned.

4.2.1 Functionality of passport authentication authority

In accordance with the role of the authentication authority in "simple single sign on architecture" [2], the Passport authentication authority performs three tasks to authenticate a user from a participating web site. These tasks are: token validation, credential validation and authentication exchange. When a user arrived at passport.com for authentication, passport authentication authority will first perform the token validation by trying to retrieve the token issued by passport authentication authority from the user. As mentioned above, the token is an encrypted cookie stored on user's browser. If the cookie can be found, and also is valid, then the user will be treated as authenticated. The word "valid" in this case means that the cookie has not been expired, and also been correctly encrypted. Otherwise the user will be redirected to the sign in page which resides on login.passport.net. Then user needs to submit his e-mail address and the password to the passport authentication authority. After the credential passed in, credential validation is preformed. Passport authentication authority will check that passed in credential against those stored in the passport credential database. If there is a match found, the passport authentication authority will issue the encrypted token to the user. Also for those of the users arrived at the passport authentication authority by being redirected from a participating web site, they will be redirected back to the web site. During the redirection, the authentication exchange performed by embedding a secret message to the participating authentications authority. Passport authentication authority uses the secret key shared with the participating web site to encrypt the message. The content of this message includes information such as PUID and time of authentication proceeded

4.2.2. Functionality of the web servers on the participating web sites

With the role of resource server, the web server on a participating web site now will not only respond to host the resource but also need to check the identity claimed by each user want to access the resource on its site. There are two tasks need to be performed by the web server of participating web site, token validation and authentication redirection. Whenever a user requests resources, it first performs the token validation by looking up the cookie issued by itself from user’s browser. If the cookie can be found and also not expired, it will treat the user as having been authenticated. Otherwise the web server will perform the authentication redirection by redirecting the user to the passport authentication authority with the information about web site that is embedded in the URL. Once the user has been redirected back, this web server will use the information encrypted in the query string to issue the cookie in the user’s browser.

Now the functionalities of the passport authentication authority and web servers on participating web sites are clear. Let’s take a look at how those two components cooperate with each other to handle the processes to authenticate a user.

4.2.3 Handle authentications inside .NET Passport architecture

Within the implementation of “simple single sign on architecture” [2], the authentication process can be divided three steps, token validation on participating web site, token validation on passport authentication authority and the credential validation on passport authentication authority.

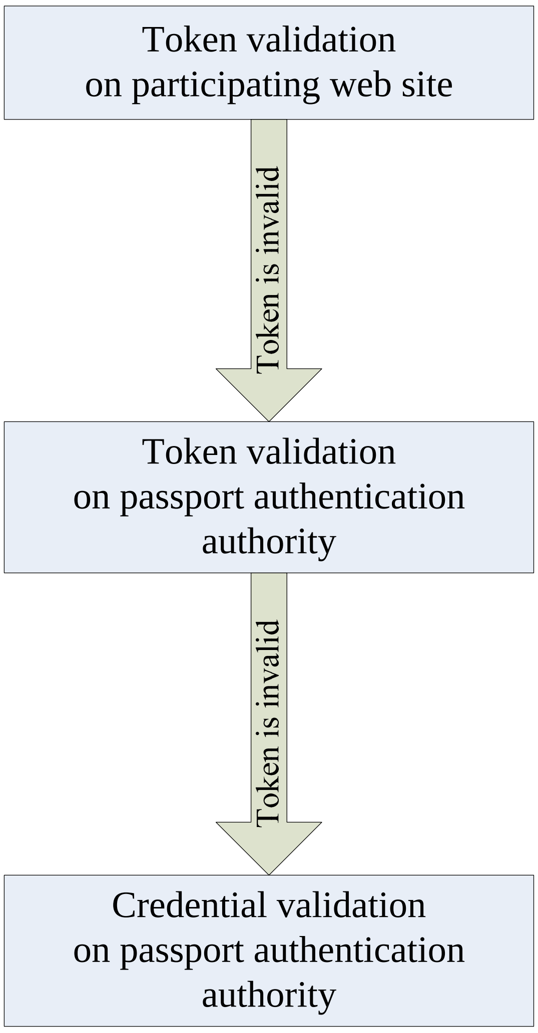
Token validation on participating web site
Token is invalid
Token validation on passport authentication authority
Token is invalid
Credential validation on passport authentication authority

Figure 2 Authentication steps

As shown in Figure 2, three steps are lined up in an order. That is the “token validation on participating web site” applied first. If the token issued by the web server of participating web site is invalid, then the “token validation on passport authentication authority” will be applied. If then token issued by passport authentication authority is invalid, then “credential validation on passport authentication authority” will be applied.

To be more detailed, from the perspective of a participating web site, there are three situations when an authentication process occurred:

<u>The first situation</u>: that user has not authenticated with passport authentication authority before.

<u>The second situation</u>: that user has authenticated with the passport authentication authority but not authenticated with this participating web site before.

<u>The third situation</u>: that user has authenticated with passport authentication authority and this participating web site before.

In the first situation, web server of the participating web site will perform the token validation first. In this situation, the cookie issued by this web server will not be found. Then it will continue with the authentication redirection. From here, that user will be redirected to passport authentication authority along with the information of the participating web site that embedded into the URL. When the user arrived, passport authentication authority will perform the token validation first. Since the user has not authenticated before, passport authentication authority will not find any token. Then it will redirect the user to the login page on login.passport.net. At this page, user enters his user name and password on the page, and submitted to passport authentication authority. And the credential validation will be carried out. If this task is success, two actions will occur. Firstly, passport authentication authority issues the token as a cookie on the user’s browser. Secondly, Passport authentication authority performs the authentication exchange by embedding the secret message in the query string while redirecting the user back to the participating web site. Once the user is back to the web site, the web server of the participating web site retrieves the encrypted message in the query string and decrypts the message. Using the content of the message from passport authentication authority, the web server of the participating web site will then be able to authenticate the user. Then it will issue another cookie to user’s browser. This avoids the duplicated authentication processes when user revisits the web site in the future.

In the second situation, it is similar to the first situation at the beginning. The web server of the participating web site performs the validation of token, but it is failed. And after user been redirected at passport authentication authority, the token validation will be applied. Since the user has been authenticated by passport authentication authority before, the token issued by passport authentication authority will be found. If the token is not expired, the passport authentication authority will skip the credential validation step, and carry out the authentication exchange directly. Otherwise passport authentication authority will continue with the credential validation.

In the third situation, since the user has been authenticated by this participating authentication authority before, the cookie issued by this participating authentication authority can be found in the user’s browser. If it is not expired, the token validation on participating web site will be success, and there will be no need for further steps. But if then cookie is already expired, for instance, the age of the cookie is exceeded the maximum age specified by the web site. Then further steps will be required.

5. Security in .NET Passport

.NET Passport, as a product of single sign on, brings about security advantages. But on the other hand, there are also some potential security related risks. Since the single sign on system is not a perfect secured system, both security advantage and disadvantage exist in the system. In this section, we will discuss the security issues in .NET Passport. First of all, let’s look at the problems existing in the generic single sign on system from the security aspect.

5.1. Problems of single sign on

Within the single sign on system, the main critical issue is the “key to the kingdom” mentioned in a paper written by Jan De Clercq [2]. That is the compromising of that single credential will make the attack gain all the access right of the illegal user. Additionally, since the authentication process is centralized, the primary authentication authority will become the bottle neck of the system. In the case of those systems based on the “simple single sign on” architecture, there is only one authentication authority, and all the users have to have their identities to be authenticated by that authentication authority first. If the authentication servers are down caused by attack, none of the users can have his identity to be authenticated. And the resource server will reject the request from the uses. This will affect the availability of authentication service.

Now we will focus on those concerns mentioned in the implementation of single sign on in .NET Passport. And within .NET Passport, some of the concerns are solved, but some of them are still remained. Also there is other concern which is raised because of the implementation .NET Passport chose. So we will discuss the security issues of .NET Passport in two directions, appreciation and criticism.

5.2. Security related appreciation in .NET Passport

In .NET Passport, authentication processes are centralized by providing the passport authentication authority. Both companies and users gains benefit from that, but it has also become an important security issue. By using .NET passport, participating web site will delegate all the authentication process to the passport authentication authority. The downtime of the authentication servers within the passport authentication authority will make the resources and services provided by participating web sites unavailable to the users. As has been mentioned in the description of the implementation of “single sign on architecture” [2] in .NET Passport, the passport authentication authority resides a location of www.passport.com and the registration and login pages resides on www.passport.net. The user interfaces of the services are separated with the passport authentication authority. During the authentication process, users will be redirected between pages on www.passport.com and www.passport.net. But the time interval is very small; the user will not even notice this redirection. The actual location of the servers of the passport authentication authority is then transparent to users. With such consideration in the implementation, .NET Passport reduces the possibility of the attack to the passport authentication authority.

5.3. Security related criticism in .NET Passport

Within the problem of “key to the kingdom”, the format or type of the credential appears to be vital. In .NET Passport the credential is a password and e-mail address pair. Using this type of credential is not suggested by Jan De Clercq [2]. As he said, using biometric-(such as fingerprint) or possession-based (such as smart card) will reduce the risk. However, in real life, there may be only a few people who want to use smart card or scan their hands to read their emails in hotmail. But using this type of credentials, .NET Passport needs an algorithm to check the weak passport during the registration phase and an unsuccessful login limit at the login phase. On the surface, both of them are implemented in .NET Passport. But the password check is actually very weak in .NET Passport. It only checks whether the length of the password is less than 6 and whether the password chosen is a portion of the e-mail address user entered. Other weak password such as repeated numbers and those words that can be found in the dictionary will not be checked. Even the unsuccessful login limit is implemented. .NET Passport should not allow the users to choose a weak password in the first place.

Besides “Key of the kingdom”, there is another problem within the single sign on implementation in .NET Passport. It is called “Bogus merchant”, which is pointed out by David P. Kormann and Aviel D. Rubin in the paper of “Risks of the Passport Single Signon protocol” [1]. As we talked about before, when users want to sign in, their credentials will be passed to passport authentication authority. But how can those uses know that their credentials are really passed to passport authentication authority. For an ordinary user, the only way to tell is the .NET Passport looking login page. In fact this page is not difficult to fake. If a malicious web site faked a .NET Passport login page, then when users try to login from this web site, their credentials will be collect by this malicious web site. Such problem is directly referred to the fundamental design of the implementation. As the provider of the sign on sign on services, .NET Passport needs to inform users about the existence of such problem and ask user to inspect the URL of the login page carefully in order to protect their credentials again those malicious web sites. But there are no such actions from .NET Passport.

6. Conclusion

With .NET Passport, both users and companies will gain many benefits. From users’ perspective, only one credential needs to be memorized. It reduces the possibility of compromise of credential, since when users have to have multiple credentials, they will either try to use some weak password or just write them down. Also there is only one sign on needed, it will improve the users’ online experience. From companies’ perspective, .NET Passport reduces the cost and workload of authentication process and users’ credential administration. It can allow companies to focus more on other phases of the business to improve the quality of the services.

As one of the single sign on solution, .NET Passport gives the users of participating web sites a convenience while accessing resources and services, and makes the life easier for the administrator and developer of participating web sites.

7. Reference

[1] David P. Kormann and Aviel D. Rubin,
"Risks of the Passport Single Signon Protocol"
Computer Networks, Elsevier Science Press, volume 33, pages 51-58, 2000

[2] Jan De Clercq.
"Single sign-on architectures"
In George I. Davida, Yair Frankel, and Owen Rees, editors, Infrastructure Security, International Conference, InfraSec 2002.

[3] Microsoft.
".Net Passport Review Guide"
Microsoft, January 2004; Available from:
http://www.microsoft.com/net/services/passport/review_guide.asp
Accessed 14 Oct. 04