Trusted Identity and Session Management Using Secure Cookies

Joon S. Park and Harish S. Krishnan

Laboratory for Applied Information Security Technology (LAIST),
Syracuse University, Syracuse, NY 13244-4100
{jspark, hskrishn}@syr.edu

Abstract. The concept of federated identity management is increasingly coming to use in order to bring Service Providers closer to customers. Users are being provided an enriched experience while carrying out business on the Web at reduced overhead and improved customer service. The idea of maintaining a single profile and gaining access to multiple services has been accepted well by the customers. However, the benefits of breaking through just one set of credentials to gain access to multiple services has made the concept of Federated Identity Management of high interest to malicious users. In this paper, we analyze the structure of a generic Federated Identity Management System and explore the .NET Passport framework in depth. We explore the current security mechanisms adopted by the .NET Passport and identify potential security weaknesses. We then propose our new approaches to enhance the security services in .NET Passport by using Secure Cookies. Our approaches are transparent to and compatible with the current .NET Passport server. Finally, we prove the feasibility by implementing our ideas in a real system.

Keywords: Cookies, Identity Management, .NET Passport.

1 Introduction

In the world of ever growing businesses it has become important to provide resources and services ubiquitously. This has lead to making businesses available through the Web. No doubt that the enabling of services or resources on the Web has opened up a gallery of opportunities, it has also brought along a wide range of security concerns. Providing services through the Web has enriched the experience of conducting business for the customers. However, this enriched experience is packaged with fears of compromising sensitive information to malicious subjects who intend to break through week security perimeters and gain access to unauthorized resources. One of the main security concerns while making businesses Web-enabled is Identity Management (IM [GC02, NRC02]). By Identity Management we mean capturing and storing of User identities, managing the identities of Users, and authenticating Users based on their identities. Once authenticated, the User has gained access into the system, but access control policies determine what parts of the system the User has access to. A strong and

S. Jajodia and D. Wijesekera (Eds.): Data and Applications Security 2005, LNCS 3654, pp. 310–324, 2005.

effective access control mechanism should provide fine grained and scalable services [KPF01, PCZG04, PKF01]. Such mechanism can protect sensitive identity information from reaching malicious hands but still continue to provide Users with an enjoyable experience of conducting business on the Web [AAM, Cla94]. In this paper, we are first going to explore the ongoing efforts in Federated Identity Management. We examine the security strengths and potential weaknesses in one of the most popular IM models, namely .NET Passport [MNP04, Pass05]. We then go on to propose a possible solution to strengthen the current mechanism by extending our previous work, Secure Cookies [PSG99, PS00], within the .NET Passport framework. Our work is both transparent to and compatible with the current .NET Passport framework.

2 Related Work

In this section we discuss a generic identity management framework and explore the .NET Passport framework [MNP04, Pass05] and Liberty Alliance Project [ILAIA03, ISLSI03] in detail. In a later section we are going investigate the .NET Passport Service and explore the current security features of the framework. We go on to suggest mechanisms to improve on the exiting security features of .NET Passport. Although we have been working with the .NET Passport frame in this paper, we believe that our work can be applied to other Identity-Management systems including Liberty Alliance.

2.1 .NET Passport

Passport is a Web-based authentication service that helps Users and service provides use the Internet faster and easier. One of the largest online authentication systems in the world,.NET Passport provides Users with Single-Sign-in (SSI) services, reducing the amount of information the User needs to remember or resubmit to various sites. For businesses, Passport helps make their websites easier for visitors and customers to use and also helps reduce the costs associated with resetting forgotten User passwords. By helping Users to connect easily to websites, Passport also makes it easy for businesses to recognize their customers and deliver consistent, valuable services no matter how or where customers are connecting. Passport can help companies allow customers to easily identify themselves across all applications offered by a website, so that Users can go to the company website, interact with all of their account information, pay bills, and get the kind of experience they want. In this paper we first investigate the identity management capabilities of Passport and discuss the authentication mechanism used by it. We then investigate the impending threats in such form of authentication and propose feasible solutions.

2.2 Liberty Alliance

The Liberty Alliance Project aims at reducing the differences between various businesses of the networked world and providing a framework for different Web-

enabled services and resources to interact across boundaries in way that respects privacy of the participating entities and enables security of shared identity information. The Liberty Alliance divides any Web-enabled business that requires authentication into 3 basic entities, namely - Identity Provider, Service Provider, and Principal. The Identity Provider is a Liberty-enabled entity that creates, maintains, and manages identity information or Users and provides Principal (User) Authentication to other Service Providers within a circle of trust. A Service Provider is an entity that provides different services to the Users. Finally, the Principal is an entity that can acquire a federated identity that is capable of making decisions and to which authenticated actions are done on its behalf. The Liberty-Alliance proposes the construction of a Circle-of-Trust between Service Providers (SP) and Identity Providers (IP), such that all authentications carried by the IP for a Principal is readily accepted by the SPs that require carrying out business with the Principal.

As it can be seen there are a number of similarities between the .NET Passport framework and the Liberty Alliance Project. Both of them have a single authority that provides authentication information based on which the other participating sites authenticate the User. Both IM systems provide capability for the User to login just once and access services across multiple providers who are all united in a circle of trust. Similarly, a single log-off request would invalidate the Users' identity across all partner Service Providers. The .NET Passport and the Liberty Alliance differ in the way they share the User authentication information within the circle of trust. The .NET Passport uses Cookie based authentication mechanisms that we explore in detail in later sections. The Passport Sign-in Server does not directly communicate with any of the other Passport-enabled sites to convey User authentication information. On the other hand, in the Liberty Alliance project, when a User authenticates with the Identity Provider, the Identity Provider federates the User Sign-in credentials with the Service Providers within the Circle of Trust. Here the User (actually the Web-Client) does not act as a medium to convey the authentication information.

3 Federated Identity Management (FIM)

A single customer may have transactions with many businesses that are Web-enabled and hence each business maintains the identity information of the User required to authenticate the User. The User has to also remember his/her authentication credentials (such as User-ID and Password) for each of the services/resources that are to be accessed through the Web. It is quite often the case that Users tend to forget their credentials and request for resetting this information. It lies in the interest of the service/resource providers to find a way to avoid the overhead and extra costs associated with resetting the authentication credentials frequently. In an attempt to address this issue, we look at the concept of Federated Identity Management. We investigate its use in the .NET Passport framework and in the Liberty Alliance Project.

By federation we mean the setting up of a trust relationship between two or more entities for a more tightly integrated approach to business. The members of one of the participating entities can freely access the resources or services provided by the other entities of the federation [PCND04]. Managing federations in essentially about a) Managing trust relationships between the participating entities and b) Managing identities of Users across the partners of the federation. Trust relationships between the participating entities of a federation can be achieved by pre-established legal agreements and implemented by the means by cryptographic techniques such as encryption/decryption and signatures. We discuss federated identity management in detail as below.

Federated Identity Management provides a mechanism for identity management across boundaries of the participating entities in the federation. One of the federation partners acts as an Identity Provider for all Users in the federation and other partners accept the authentication decisions made by the Identity Provider and allow authenticated Users to access the resources or services offered. The Identity Provider and other partners exchange User handles by means of a channel that is independent of the services offered by the partners, typically this is achieved through XML [XML] based SOAP [SOAP] messages. Once a circle of trust is established between a set of partners that includes at least one Identity Provider, the access of services or resources by a User can be carried out in two distinct ways. Firstly, the User creates an account with the Identity Provider and then goes on to access the services or resources offered by other partners of the federation. Alternatively, the User directly access any of the services or resources hosted on one of the Service Providers of the federation which directs the User to set up an account with the Identity Provider and then re-direct back the User to the Service Provider. Both of these scenarios are depicted in Figure 1 and Figure 2. The architecture for the Liberty Alliance is very similar to the described scenarios of Figure 1 and Figure 2. The User is termed as Principal and the federation of identities is achieved through the exchange of SOAP messages. The architecture of .NET Passport also resembles Figure 1 and Figure 2. However, the names by which Microsoft calls these entities of a circle of trust are Passport Sign-in Server (Identity Provider) and Service Providers configured with Passport Manager Objects (Service Providers). The .NET Passport has marked difference in the way it achieves federation of identities. Passport transfers it's identity management information via cookies [ES96, KM00]. The User sets up what is known as the Passport Profile with the Passport Sign-in Server. At this time the Sign-in Server creates a set of cookies. Using these cookies the User is authenticated at various Passport enabled Service Providers. By Passport-enabled we mean those Service Providers that have the Passport Manager installed on their systems that are responsible for authenticating Users based on Passport User-ID (PUID). In the Passport framework, the Passport Sign-in Server is essentially the Identity Provider and other Passport enabled sites are Service Providers and communication between the Identity Provider and Service Providers is achieved through exchange of cookies. Later on in this paper we discuss in detail the authentication mechanism adopted by Passport.

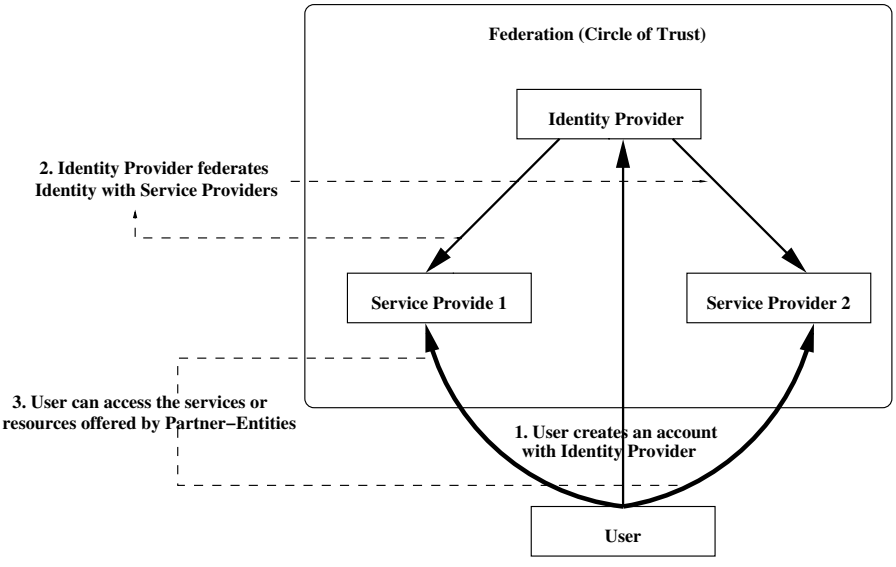

Fig. 1. User establishes Identity with Identity Provider before accessing services/resources of the federation

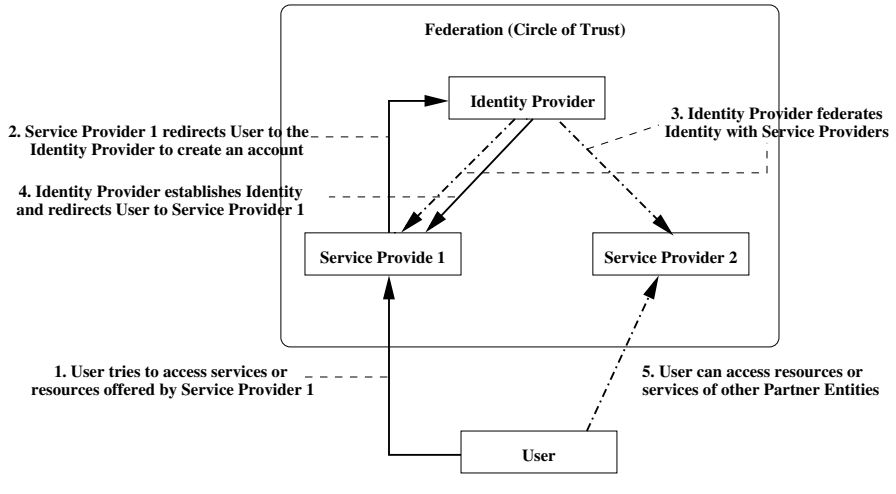

Fig. 2. User attempts to access services/resources of the federation and is directed to the Identity Provider to set up an account and then is allowed access

In the next section we look at the authentication activities involved when an existing User tries access a resource or service in the federation and how a single time signing-in provides access across the federation

4 Analyses of Authentication Mechanisms in .NET Passport

As we discussed briefly in the previous sections, the Passport framework uses "cookie-based authentication" mechanisms to convey identity related information to other Passport enabled sites and acquire service access for Users. This form of authentication is useful because, cookies are lightweight and convenient to store and forward. They involve very less processing overhead. In order to understand how .NET Passport uses cookies to authenticate Users with partner sites we analyze the mechanism in this section.

Figure 3 provides an overview of the authentication process. Before we dive into the details of the User authentication and service access mechanism, we briefly discuss the components of the Passport architecture. The Passport Sign-in Server provides authentication mechanism by which a User can be established as a trusted User among the Passport enabled sites. It takes the help of a User Authentication Database to authenticate any given User. On the other end, a Web-server serves the specific service that the User wants to access. The Service Provider is configured with a Passport Manager that is responsible for verifying the authenticity of the User.

The User starts by accessing the website that provides the desired service. It is assumed that the website is Passport enabled; by this we mean that the site has agreed to be a member of the federation of all Service Providers accepting the .NET Passport as a central identity manager and authenticator. The Web-server of the Service Provider, checks for available cookies issued by the Passport server, and if it cannot find any, it redirects the User to the Passport Sign-in server. (If Cookies are available on the Users system, probably saved from the previous access made to the Service Provider, then the process from step 7 of Figure 3 will be executed.) The User forwards his/her Sign-in credentials such as the email-address and password to the server. The server verifies these credentials of the User against the database and notifies the User of successful authentication. At this time the Passport login server places its cookies into the User's browser. Using these cookies the User is authenticated at various Passport enabled Service Providers. Having embedded the cookies in the browser the User is redirected to the Passport-enabled site where the Web-server extracts the cookies and forwards it to the Passport manager component. The Passport manager then verifies the User as already authenticated and signed-in and notifies the Web-server of the authenticity of the User. The Web-server then grants the User the permission to access the services provided. The site also records its URL in one of the cookies to reflect the visit of the User to that site. If the User goes on to access other Passport enabled sites, since he/she already has the required cookies embedded in the browser, the User can do so without any extra interaction either with the Passport Sign-in Server or the Service Provider.

It is however important to note here that, the Passport Manager on each Passport-enabled Service Provider only checks that the browser of the User accessing the website is embedded with the Passport Cookies. It does not in any way try to establish a link between the Cookies in the browser and the User

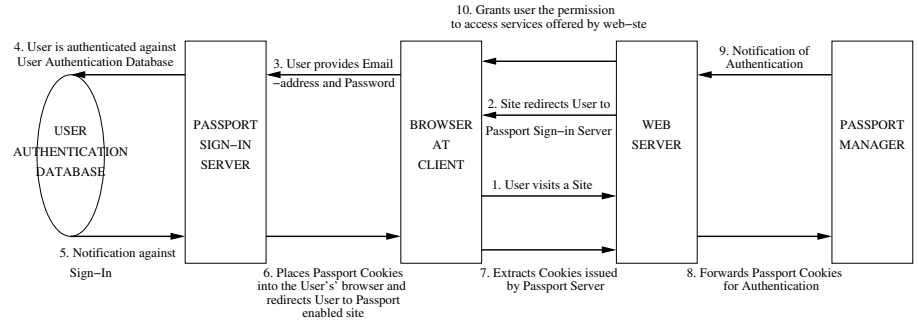

Fig. 3. Cookie-based authentication mechanism used by .NET Passport

whose browser is presenting the cookies to the Web-server. This is one of the main vulnerabilities in the current .NET Passport system that we discuss in the next sections.

5 Security Concerns in Conventional Cookies

Having stated that .NET Passport uses cookies for authentication, we briefly look at what cookies really are and discuss the security concerns associated with them.

Cookies are basically text files that were used to enhance stateless-HTTP by introducing what is known as a Session in HTTP. By this we mean that cookies are text files containing information that can be used to achieve continuity while surfing the internet. Cookies often contain information that was previously entered by a surfer, but they can also contain information added by Web-servers. A site stores cookies on the User's machine that it retrieves on subsequent visit to the site. The information contained in the cookies is what the User or the Web-server entered during the Users's previous visit to the site and the site extracts the corresponding cookies to obtain information.

There are no direct threats posed by cookies if they are used to store non-confidential User information. However, there are situations when a site might need to store some sensitive information such identity, passwords, etc. It is then that cookies become point of interest to malicious users. We now look at the possible security threats to cookies [PS00]. These threats can be classified into the following groups: Network Threats, End-System Threats, and Cookie-Harvesting Threats. As cookies are transmitted in clear text, Network Threats are implemented by snooping and replay with or without modification. One easy solution to thwarting Network-Threats on cookies is to use the "secure" flag field in the cookie. Setting the flag ensures that those cookies are transmitted only on SSL [WS96] so that the cookies are protected on the network. However, this does not mean that the cookies are protected in the end-systems, since SSL does not work in end-systems. The second type of threat is the End-System Threats. Cookies

reside in the Users systems as plain text files. These cookies are open to be modified or copied to other systems with or without User consent. This gives rise to impersonation by identity forging. Actually, this is a serious drawback in the current .NET Passport authentication. One good example of the End-System Threat is the Cookie-Poisoning Attack [Kle]. In this form of attack, a malicious user modifies the contents of a Cookie to gain the identity of a genuine User and gain unauthorized access. The Cookie-Poisoning Attack is a Parameter Tampering Attack that involves the tampering of cookie parameters. Lastly, we have the Cookie-Harvesting Threat, where a malicious user collects User's cookies by claiming to be a site that accepts User cookies. The attacker then goes on to use these harvested cookies to gain access to the sites that actually accept these cookies. A good example of the Cookie-Harvesting threat is the Cross-Site Scripting attack. In this form of attack a malicious user can exploit the vulnerabilities of a Website that displays data, provided by a User, which has underlying malicious intent. For example, a malicious user could embed a script in a URL and place it in a Discussion Forum, Website, Web message board, or email. The underlying script could be activated when the User comes across the hyperlink and decides to follow it. The script would then copy all the cookies in the User's machine and relay they back to the malicious user for use with or without modification.

In all of the above cases it is seen that there is no means by which a site can establish a strong link between the cookies and the User providing the cookies. Furthermore, cookies do not provide confidentiality or integrity of their contents. All these attacks use the weak assumption made by a cookie-accepting site that the User providing the cookies is the actual owner of those cookies.

6 Security Vulnerabilities in .NET Passport Cookies

The .NET Passport uses a number of cookies that each carries state information for various tasks to be carried out by the Passport Sign-in Server and the participating Passport-enables sites. These cookies are divided into two groups, namely the "Domain Authority Cookies" and "Participating Site Cookies". Cookies that are written to the .Passport.com domain cannot be directly accessed by a participating site. On the other hand the Participating Site Cookies are written to the participating domain site and in path to which the participating site's Passport Manager Object is configured and enable the User to sign in at any Passport participating sites during a browser session. The cookies written in the passport.com domain are encrypted with the Passport key. The cookies written to the participating sites domain are encrypted with the participating sites' Passport key. Most of the Passport Cookies are temporary cookies and are not stored in the User's browser after each session. When a User signs out of either the Passport site or one of the participating sites, all Passport Cookies are deleted. By deleting the cookies at the end of each session, the possibility of end-system threats is thwarted. However, every session requires a new set of cookies from the User's browser, which is inconvenient to the User. It would make it

much easier if the Users could preserve their cookies that provide automatic authentication each time the User uses the Passport site or any Passport-enabled site in the future. Technically, it should be possible to configure the Passport server to store its cookies in the User's browser after each session. However, this is not recommended because of the security vulnerabilities in the current Passport Cookies, although it could provide more transparent services to Users. We will discuss in the following sections how to solve this problem by using our Secure Cookies.

The Passport Cookies currently have two security features: namely the encryption using Passport Key and transmitting the cookies using SSL. The encryption provides protection against unauthorized disclosure. Setting the Secure flag in the cookie makes sure that it is transmitted over SSL thus giving away possibility of network-threats by malicious user. However, these features are not sufficient to protect the cookies in the User's machines(discussed in Section 5).

Furthermore, the current cookies don't have a mechanism to associate them with their actual owners. The cookies don't have a mechanism by which it can be established that the person forwarding the cookies is necessarily the owner of the cookies. This weakness can be exploited by steeling the cookies and reusing the cookies without modification before the cookies expire. In this way a malicious user pretends to be the cookie-owner and gains unauthorized access. Once any User signs in at the Passport Sign-in Server, that server places required cookies in the User's system to be used for authentication with Passport enabled sites. Any malicious user can carry out an end-system attack in order to obtain the cookies and use them as its owner. Therefore, we believe the current .NET Passport IM system shouldn't not store it's cookies in Users machines, even though it is technically possible and would be more convenient to Users.

7 Secure Cookies for .NET Passport Framework

We now look into the aspect of enhancing the security services in the Passport framework. We propose the use of Secure Cookies to improve the security of the cookies used in the current .NET Passport framework.

7.1 Cooking Secure Cookies

Secure Cookies provide three types of security services: authentication, integrity, and confidentiality. Authentication verifies the cookie's owner. Integrity protects cookies against unauthorized modification. Finally, confidentiality protects cookies against being revealed to an unauthorized entity. Detailed descriptions about Secure Cookies are available in [PS00].

Since typical cookies do not support authentication, a malicious user can simply snatch cookies from other Users and impersonate the real owner to the server that accepts those cookies. To solve this problem, we introduce three possible authentication methods for cookies. Authentication cookies can be address-based (IP_Cookie), password-based (Pswd_Cookie), or digital-signature-based

(Sign_Cookie). To prohibit individuals, perhaps even the cookie owner, from reading sensitive information in cookies, the Web-server can encrypt the contents of the cookies such as, names, roles [PSA01], credit card numbers, and so on. We use the Key_Cookie, to store an encrypted session key, which is used to encrypt sensitive information in other cookies. The session key can be encrypted either by the proper public or secret key. Finally, the Seal_Cookie determines if cookies have been altered. The Seal_Cookie's contents depend on the cryptographic technologies used - essentially, either a public- or secret-key-based solutions (e.g. digital signatures, message authentication codes).

As a result, Secure Cookies can be stored in the User's computer, even when it is off, after each session. This is possible because the Secure Cookies can be provided integrity and authentication services as well as encryption. Therefore, once the User obtains Secure Cookies, the information in the cookies can be used until the cookies expire. This approach completely solves the stateless problem of HTTP and security problems in typical cookies used in .NET Passport.

7.2 Enhanced Passport Architecture with Secure Cookies

Before we talk about the effect of using Secure Cookies, we describe the architecture that supports the use of Secure Cookies. In order to support the use of Secure Cookies we propose Proxy-Authentication architecture that is both transparent to and compatible with the current .NET Passport framework. Figure 4 is a schematic representation of the same. Figure 4 depicts the use of a Proxy-Authenticator to generate Secure Cookies from Passport Cookies. We have not shown the User Authentication Database and Passport Manager entities in Figure 4 as they do not form the core of the Proxy-Authentication architecture that we propose. However, these entities are required in the overall system. If this architecture is compared to the existing Passport architecture, we can clearly see that the User is no more talking directly to the Passport Sign-in Server. Instead the User provides the credentials to the Proxy and Proxy signs into Passport

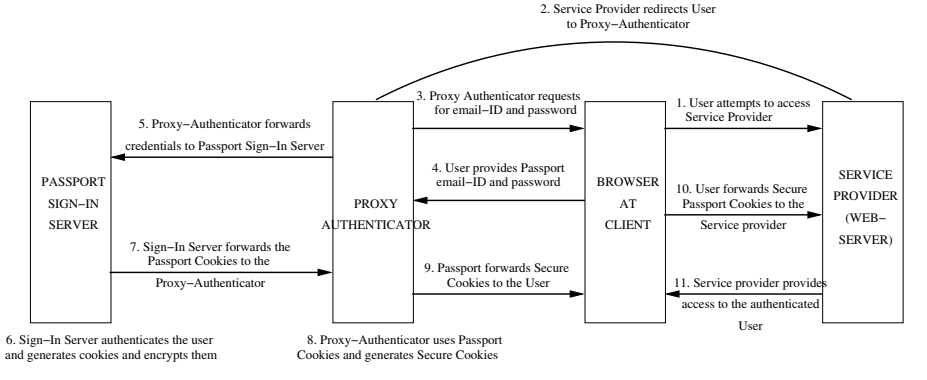

Fig. 4. Proxy Authenticator used to provide security to Passport Cookies by generating Secure Cookies

on the behalf of the User. The key role of the Proxy is to receive the post authentication Passport Cookies from Passport Sign-in Server, and to generate the Secure Cookies using them and forward them to the User. The Proxy thus wraps the less secure Passport Cookies in a secure envelop that is difficult to break. The User can then use these cookies for gaining access at each Passport partner site. The architecture also requires that the partner Service Providers possess the capability to unwrap the Passport Cookies from the Secure Cookie envelop. Hence each Service Provider needs to be configured not only with the Passport Manager Module but also a Secure Cookie Unwrapping module that extracts Passport Cookies and forwards them to the Passport Manager Module for verification. (This module is not shown in the architecture diagram presented in Figure 4). Furthermore, by using the proposed Proxy-Authenticator, our approach is still transparent to, and compatible with existing .NET Passport servers. Our approach does not require any change in the .NET servers. Instead, in our experimentation, we successfully changed the original content of the cookies issued by the Passport Server for our purposes. Actually, .NET Passport does not have any idea about the content change. Although changing the original cookie contents is not for a malicious purpose in this case, it clearly indicates that we can break the integrity of the original .NET Passport Cookies because they are insecure.

7.3 Operational Scenario

Having described the role of Proxy, we now present the detailed discussion on the sequence of the steps that take place for a User trying to gain access into a partner Service Provider that is Passport enabled. The User starts by trying to access the Service Provider and requests for the desired resource. The Service Provider does not find the required cookie in the User's Web-client and hence redirects the User to the Proxy to Sign-in to the Passport. At this time the User is interacting with the Proxy-Authenticator and not the Passport Sign-in server. However, this difference is transparent to the User. The User provides the Sign-in credentials just as he/she would interact with the Passport Sign-in server. The Proxy collects these credentials and forwards it to the Sign-in Server and signs in on behalf of the User. At this point a successful Sign-in would result in the User's Passport Cookies being written to the Proxy. The Proxy then generates a set of Secure Cookies for the User by wrapping all the Passport Cookies. These Secure Cookies are transferred to the User's Web-Client. The User is now redirected to the Service Provider that he/she initially tried to access. Now the Service Provider finds the valid cookies that it can use to verify the User and grant access. It is important to point out here that when the Secure Cookies are read by the service the User will be prompted for a password that is stored in the Hashed Format in the Authentication Cookie of the Secure Cookies (if the authentication mechanism in the Secure Cookie is implemented using Digital Signature or IP, then the verification mechanism is oblivious to the User). If the User provides the correct password then the Service Provider successfully verifies the User and strips the Passport Cookies from the Secure Cookies set

and presents it to the Passport Manager Module. Based on the credentials in the Passport Cookies the Service Provider offers a range of services to the User.

7.4 Advantages

Having discussed the sequence of interactions in the new architecture we now shift our discussion towards the strengths of the proposed architecture. In spite of introducing a new component between the User and the Passport Sign-in Server, the amount of overhead is very little. In fact the overhead incurred is all on the Proxy and nothing on the User. The User still signs in at only one place. However there is one more level of communication behind the Proxy (between Proxy and Passport Sign-in Server) but this is transparent to the User. Again when the User's Web-client presents the Service Provider with the Secure Cookies (instead of the Passport Cookies as per the existing system) it is the Service Provider that first strips the outer Secure Cookie set and then the Passport Manager consumes the Passport Cookies. The Proxy based architecture is also robust as it does away with the Single-Point failure. The current Passport framework has a single Sign-in Server and can fall victim to single point failure. There could be multiple proxies thus eliminating single point failure for the signing-in part. The use of a Proxy is also secure in the sense that, the Proxy does not see the contents of the Passport Cookies as the cookies are already encrypted with the Passport Key when they arrive at the Proxy. It just adds another layer of security on the cookies making them more secure. Also the cookies are not stored at the Proxy, after creating them, the Proxy forwards them to the User. This eliminates any possibility of an end-system attack on the Proxy.

The use of Secure Cookies itself adds a number of advantages to the .NET Passport framework. Primarily the authentication cookie of the Secure Cookie set helps establishing a link between the Cookies and the Owner. If the authentication mechanism chosen is a Pswd_Cookie then the User incurs a minor overhead of having to remember a password to be provided at each Service Provider. Using Digital-Signature will also incur some amount of overhead as there needs to be a secure channel set up to exchange Security Keys and also each time the User needs to intervene in the process. However, if IP-based based authentication is used then there is no overhead for the User and the authentication mechanism is transparent to the User. Also we can incorporate access control mechanism by using the Secure Cookies. Secure Cookies can be stored on the User's system without concern of being tampered upon. Any unauthorized change to the contents of Secure Cookies would result in invalid cookies and denial of access. Also, if Secure Cookies comprising of Passport Cookies is stolen by a malicious user and used, the cookie will not be successfully validated as the malicious user has to know the password of the actual owner of the cookie. We can optionally add another cookie (e.g. Role_Cookie) to the Secure Cookie envelop that has defines access rights of the User. This concept can be further extended by using Role-Based Access Control (RBAC [PSA01, SCFY96]). We can include a Role_Cookie into the Secure Cookie envelop that caries the User's Role in it. This role information can be used by each of the Service Provider to

make access decisions regarding User's request for services/resources. In such a scenario the Service Provider maintains a mapping between the defined Roles and corresponding access permissions. Thus Secure Cookies not only strengthen the existing Passport Cookies but also can provide a great deal of extended capability to them such as scalable access control.

7.5 Implementation

We carried out our implementation on Windows systems. We used the .NET platform for our development. We developed an initial version of Proxy Authenticator using ASPX and hosted it on IIS. We used the System.Security.Cryptography library to perform the necessary encryption, decryption and Digital Signing of the cookie contents. We use C# to implement the logic required by Proxy as well as the module on the Service Provider that performs the unwrapping of Secure Cookies to get the Passport Cookies.

In order to test the strength of the Passport Cookies we signed into Passport with the "sign me in automatically" option selected. This gave us access to the Passport Cookies that were written to the User's Web-client. We then accessed msn.com and selected the Passport "Sign-in" option and we were silently signed into the site. We then opened up the Passport Cookies stored on the system and modified a numerical value field in the MSPAuth cookie. We saved the changes and once again accessed the msn.com site. On clicking the Sign-in options for Users with Passport identities, we were silently authenticated and granted access to the site instead of being directed to the Passport Sign-in server for re-signing in. Keeping these changes intact we further modified the cookies by changing a similar numerical value field in the MSPProf cookie. We then accessed expedia.com (another Passport enabled site) and we were silently signed-in. Similar behaviors was seen when modifications were made to the MSPPre and MSPSec cookies. Hence this led us to conclude that there are points of vulnerabilities in the current Passport Cookies that can be exploited. It is this kind of attacks that the Secure Cookies look to thwart.

8 Conclusions and Future Work

In this paper, we analyze the structure of a generic Federated Identity Management System and explore .NET Passport framework in depth. We explore the current security mechanisms adopted by .NET Passport and identify potential security weaknesses. We then propose our new approaches to enhance the security services in .NET Passport by using Secure Cookies. Our approaches are transparent to and compatible with the current .NET Passport server. Finally, we prove the feasibility by implementing our ideas in a real system.

In our future work we intend to add a Role_Cookie in the set of cookies that forms a set of Secure Cookies to enable Role-Based Access Control that provides strong and scalable access control mechanisms. We intend to equip Proxy with capability to determine a role for the User who wants to Sign-in to Passport and

add a Role_Cookie to reflect this information. We will be enabling the Service Providers with ability to map Roles to Permissions, so that on receiving the Secure Cookies from the User the Service Provider can not only get the Passport Cookies but also have access to a role cookie. This Role_Cookie can be used to determine the set of permissions that the User has while touring the Service Provider's website. The set of roles can be fixed but the access permissions associated to each role are flexible and the Service Providers can autonomously determine the set of permissions for each role. In this way we are able to embed RBAC mechanism into .NET Passport framework.

References

[AAM] American Association of Motor Vehicle Administrators (AAMVA). *Identification Security.* http://www.aamva.org/IDSecurity/

[Cla94] R. Clarke. *Human Identification in Information Systems: Management Challenges and Public Policy Issues.* Information Technology and People 7(4): 6-37, 1994.

[ES96] Easonn Sullivan. *Are Web-based cookies a treat or a recipe for trouble?* PC Week, June 26, 1996.

[GC02] Greenwood, D., D. Combs, et al. *Identity Management: A White Paper.* Lexington, KY, National Electronic Commerce Coordinating Council: 68, 2002.

[ILAIA03] Liberty Alliance Project. *Introduction to the Liberty Alliance Identity Architecture.* Identity Architecture Whitepaper. March, 2003. http://www.projectliberty.org/resources/whitepapers/LAP

[ISLSI03] Liberty Alliance Project. *Identity Systems and Liberty Specification Version 1.1 Interoperability.* February 14, 2003. https://www.projectliberty.org/resources/whitepapers/Liberty and 3rd Party Identity Systems White Paper.pdf

[Kle] A. Klein. *Hacking Web Applications Using Cookie Poisoning* Sanctum Inc. http://www.cgisecurity.com/lib/CookiePoisoningByline.pdf

[KM00] D. Kristol and L. Montulli. *RFC 2965, HTTP State Management Mechanism.* Network Working Group, October 2000.

[KPF01] Myong H. Kang, Joon S. Park, and Judith N. Froscher. *Access Control Mechanisms for Inter-Organization Workflow.* Proceedings of the 6th ACM Symposium on Access Control Model and Technologies (SACMAT), Chantilly, Virginia, May 3-4, 2001.

[MNP04] Microsoft .NET Passport. *Review Guide.* January 2004. http://www.microsoft.com/net/services/passport/review_guide.asp

[NRC02] Computer Science and Telecommunications Board, N. R. C. *IDs - Not That Easy: Questions about Nationwide Identity Systems.* Washington, DC, National Academy of Sciences, 2002.

[PCND04] Joon S. Park, Keith P. Costello, Teresa M. Neven, and Josh A. Diosomito. *A Composite RBAC Approach for Large, Complex Organizations.* Proceedings of the 9th ACM Symposium on Access Control Models and Technologies (SACMAT), Yorktown Heights, New York, June 2-4, 2004.

[PCZG04] Joon S. Park, Pratheep Chandramohan, Artur Zak, and Joseph Giordano. *Fine-Grained, Scalable, and Secure Key Management Scheme for Trusted Military Message Systems.* Proceedings of The Military Communications Conference (MILCOM), Monterey, CA, October 31-November 3, 2004.

[PKF01] Joon S. Park, Myong H. Kang, and Judith N. Froscher. *A Secure Workflow System for Dynamic Cooperation.* Proceedings of the 16th International Conference on Information Security (IFIP/SEC 2001), Paris, France, June 11-13, 2001.

[Pass05] .NET Passport. http://www.passport.NET

[PS00] Joon S. Park and Ravi Sandhu. *Secure Cookies on the Web.* IEEE Internet Computing, Volume 4, Number 4, July-August 2000.

[PSA01] Joon S. Park, Ravi Sandhu, and Gail-Joon Ahn. *Role-Based Access Control on the Web.* ACM Transactions on Information and System Security (TISSEC), Volume 4, Number 1, February 2001.

[PSG99] Joon S. Park, Ravi Sandhu, and SreeLatha Ghanta. *RBAC on the Web by Secure Cookies.* Proceedings of the 13th IFIP WG 11.3 Working Conference on Database Security, Seattle, Washington, July 26-28, 1999.

[SCFY96] R. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. *Role Based Access Control Models.* IEEE Computer 29 (2), February 1996.

[SOAP] Simple Object Access protocol. *Version 1.2 Specification.* June 24, 2003 http://www.w3.org/TR/soap/

[WS96] D. Wagner and B. Schneier. *Analysis of the SSL 3.0 Protocol.* Proc. Second Usenix Workshop on Electronic Commerce, Usenix Press, Berkeley, Calif., Nov. 1996, pp. 29-40.

[XML] Extensible Markup Language. www.w3.org/XML/