Abstract (Size 10 This research paper deals with Web 2.0 Services Security Architecture Design extended to Cloud, Big Data and IOT.

Keywords - Security Engineering, Security Architectures, Web Services, Cloud Computing, Big Data, IOT

I. INTRODUCTION

In this paper, methodology for Web 2.0 Services Security Design is provided, with implementations and validations of AJAX (Asynchronous JavaScript and XML) design, Web 2.0 Security design at Secure Socket Layer (SSL) and Web 2.0 Services authentication, is given, all of those design is based on Model Driven Architecture based Agile Modeled Layered Security Architecture design.

4.1 WEB 2.0 SECURITY DESIGN:

Software as a Service is a software delivery model that runs on a subscription – based fee [Francis HSU]. The users do not buy the license of the software, but only the right to use it. The solution provider retains the responsibility of supporting hardware, and maintenance of the solution. [Sumeet Mishra] Here lie the security issues. Outsourcing an application means the organization relinquishes some control. The most important thing that should be focused is control of the data [Basin D].

Web 2.0 Security

Web 2.0 technologies, such as AJAX, expand both the attack surface and the security gaps in the application. Today, about 70 percent of malicious code in the wild is downloaded via Ajax, according to Yuval Ben-Itzhak, chief technology officer of security vendor Finjan.

Why existing threat model approaches are not appropriate for Web 2.0 Apps?

There is no general notion of security. For each application different aspects of security as confidentiality, authenticity, integrity or availability may be relevant. Though abstract security policies may be defined, the security requirements are heavily influenced by the kind of attacks that are expected for the given system and the application domain.

All existing threat modeling approaches focus on given specific applications and apply some formal or informal methodology for threat modeling the given application. How ever what makes the Web 2.0 security so complicated, as stated earlier is that it covers such a broad range of applications.

A typical Web 2.0-based systems may be polling Really Simple Syndication (RSS) feeds from multiple sites, exchanging user-generated information with various Web 2.0 functionalities like blogs, forums etc. using JSON and communicating with partners web service over SOAP. All these services are accessed from a Rich interface using AJAX and/or Flash.

A Web 2.0 based systems can load several JavaScript’s, Flash components, and widgets in the browser which can then access cross-domain information from with in the browser itself. But introducing these new technologies without dynamic web security protection can lead to data loss.

There are many approaches available to secure the rich applications at various levels, for example at client-server level or at the network gateways. But most of these approaches are for intermediate mitigation. For effective collaboration and to promote utilization of these rich functionalities and web as an enterprise tool, we needed to secure information sharing of user information where user input for a particular system is utilized for another application. What’s essential is to implore security at application level as well as client communication gateway.

Web 2.0 based systems and services are collaboratively developed by harnessing under intelligence and are involuntarily constantly under development (said to be in perpetual beta). They present new weak links for malware to attack because of lack of universal threat model governing the interim releases and conglomeration of features being added continuously. Sometimes features themselves are developed to be further matured by user’s collective intelligence and they lack various security measures.

Hence it’s very difficult to secure such a development system and services. We need to identify threats, attacks, vulnerabilities, and countermeasures that could affect our system, because of type of development methodology and involves technologies, kind of deployment and polled services and the communication mechanism between them, at the outset for securing the system as well as consumed services. We need to use threat modeling to model systems design for meeting security objectives, and thereby reducing risk against malware attack or security breach. Securing Web 2.0 based systems has rich service consumers, requires a comprehensive threat modeling approach which
should be applicable to the known usage patterns for threat modeling web application and must be expandable to adapt new standards for classification of Web 2.0 threats vectors, attack patterns and mitigation.

Related Work on threat modeling of Web 2.0 Services, preliminary step involves, Decompose Application using DFD’s and UML diagrams as DFD’s focus on flow of data between processes and UML activity diagrams focus on flow of control between processes.

Collaborative development: Companies use Web 2.0 as a platform not only to communicate with customers and business partners but also as a development platform for employees, allowing them to achieve the goal of true real-time collaboration which increases productivity and provides companies with a way to more easily promote their products. In particular, the creation of online communities and blogs or wikis is to initiate conversations and share knowledge is proving to be particularly interesting to companies.

For Web 2.0 development, due to the approach of agile and collaborative development a full implementation of any security model is not possible. Informal approaches that have shown useful in practice are based on threat identification and risk analysis where the system and its environment are investigated in detail in order to determine the kind of possible attacks , their probability and the loss in case of the attack being performed.

In order to make it simpler we propose a three dimensional security approach suitable for any kind of web 2.0 development. These act as a checklist to ensure secure application at each stage of development. Since Web 2.0 based systems development does not follow SDLC and system is collection of many Web 2.0 based services mashed up to function as a single system, such system cannot be modeled at the outset with respect to security objectives and requirements. We cannot ascertain all threats, vulnerabilities, attacks and counter measures for this single system. We need to model the attacks, threats and vulnerabilities onto web 2.0 system into system-asset-attacker centric dimensions.

System centric: This is based on securing the system or part of it this can be either part of system development process or it could be addressed at later stage by decomposing the system. So it can be integrated with the software development lifecycle from the start.

Attacker Centric: It starts with the analyzing goals of the possible vulnerabilities in the system which an attacker can exploit. In this step the entry points and the assets of the system are secured from any kind of attack.

Asset centric: This step takes into account of any sensitive or important information like customer names with credit card numbers that the application processes. This level of security is introduced by methods like user based access controls to the application.

In case of web or desktop based systems disintegrating the application and looking at the three dimensions of the security is usually sufficient to secure it. In case of the web 2.0 based systems third approach doesn’t hold sufficient. The boundaries where each of the dimension intersect presents completely different scenario.

4.2 IMPLEMENTATIONS AND VALIDATIONS:

4.2.1 Web 2.0 Asynchronous JavaScript and XML (AJAX) Security Design

Web 2.0 increases web based access to data processing particularly on the client side (AJAX Asynchronous JavaScript and XML) that enables web applications which contain enriched functionality. Web 2.0 technologies have wide range of technologies and protocols which enable Web architectures greater access to data and functions. The technologies include AJAX (Asynchronous JavaScript and XML), XML, JSON (JavaScript Object Notation), SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language), REST Web API’s, Microsoft Silverlight, RSS, RDF and Atom. Web 2.0 vulnerabilities include XML, JavaScript, RSS, AJAX, SOAP, JSON, WSDL, in decreasing order of their attack statistics.

We had implemented securing AJAX Web Services issues. In order to design a secure web tier for AJAX applications, we studied the architecture of the AJAX architecture. The client running in the user’s browser makes requests to the server using Hypertext Transfer Protocol (HTTP). These requests are processed by the Web server processes, such as Servlets, dynamic pages, etc. The response time is returned to the client in the form of the streams of data. The web services or pages are accessed by the external entities, without any additional work on our part. It may be that we encourage outsiders to use our web services in this way, and we even publish an API, as eBay, Amazon and Google among others have done. Even in this case, though, we need to keep security in mind. There are two things that we done, the first one is to design our web services interfaces, or API in such a way that external entities would not be able to subvert the purpose of our web application, e.g. by ordering the products without paying for them (designing a secure web tier). The other one is to look at the techniques to restrict access to the Web services to particular parties. When we design a web application, we typically have an end-to-end workflow in mind. For example in a shopping site, the user will browse the store, add the items to their basket, and then proceed to checkout. The check out process on the other hand has its own well-defined work flow with the choice of delivery address, shipping options, and confirmation of order. As long as the application is
calling the shots, we can trust that the workflow is being used correctly. If an external entity starts to call our web services directly, however, we may have problems.

We had implemented this example using third party services, creating a search control, configure it to search across Local search, Web search, Video, and Book search, and then place the cursor on our page. (The Figure 4.1 provides the screen shot of Page of the mygoogle_search.html). We are securing our browser using JavaScript, both for the same browser and cross browser. We are used the features of Google search API directly from the site of Google. The function onload is used to load the Google search as soon as the page gets loaded. First, we created a Google search control and assigned it a variable searchcontrol. Then we added the searchers required for our example. We have used only four searches and left the image, news and blog searches. Then we set the Local Search center point, we have a method searchcontrol.draw( ). This method is used to display different draw modes – tabbed or linear. We have used the linear mode in this example. Next, we have the nsearchcontrol.execute( ) method, which is used to initialize the search, and some code for the presentation logic of the page that we can easily understand. We have initialized the search for Ajax.

![Image of powered by Google Search API](image)

4.1 .Initial Page of the mygoogle_search.html

This example generates a MD5 encrypted hash. (The screen shot is presented in figure 4.2) We show how to protect our data and to restrict unauthorized users to access it. The page contains two text boxes and a button for clearing the value entered by the user. The first text box is used to enter the password string (here it’s an example university) and the other is used to show the encrypted form of the password using MD5 algorithm. An encryption algorithm will generate a random-looking, but predictable, output from an input string e.g. MD5 algorithm. It has a few key features that make it useful for security. First, MD5-ing a piece of data will always generate the same result, every time. Second, two different resources are monumentally unlikely to generate the same MD5 digest. Taken together, these two features make an MD5 digest (that is, the output of the algorithm) of a resource a rather good fingerprint of that resource. The third feature is that the algorithm is not easily reversible. The MD5 digest can therefore be freely passed about in the open, without the risk of a malicious entity being able to use it to decrypt the message. For example, the MD5 algorithm will generate the digest string for the password string entered by the user. We can encrypt the password on the client site and transmit the encrypted form to the server. The server on the other hand, will fetch the password from the database and encrypt it using the same algorithm. The server then compares the two encrypted strings. If the strings match successfully, it would allow the user to log on. However, we can’t transmit the straight MD5 digest across the Internet in order to login. An unauthorized user may not be able to find the exact value for the password, but they would soon learn that the particular digest grants them to the user account.

![Image of Encryption of data using JavaScript (MD5 algorithm)](image)

4.2.2 Web 2.0 Services Security Design using Secure Socket Layer (SSL)

Web Services (WS) security has undergone an enormous development, as carried out by the major organizations and consortiums of the industry over the last few years. This has brought about the appearance of a huge number of WS security standards. Such a fact has made organizations remain reticent about adopting technologies based on this paradigm, due to the learning curve, which is inevitable in the integration of security into their practical deployments.
The open nature of the Internet is promoting the development of web Services (WS) security as a paradigm that enables complex scenarios of business workflow integration, while at the same time providing the intra and inter-firm hyper-connectivity with security. There are many existing security standards met. But our motive of this research implementation is to add more security as the user wishes to.

SSL (Secure Sockets Layer) was originally developed by Netscape as a means of securing communications across the Internet, primarily for electronic commerce. A total of three versions of the protocol were developed, with the last one, SSL version 3.0, being released in 1996. Since then, the development of the original idea has continued with TLS (Transport Layer Security), with the first version being published in RFC 2246 in January 1999. At this point, virtually all financial institutions have endorsed the use of SSL for electronic commerce.

This research implementation features include:

- We initiate our implementation by exploring the java cryptography architecture JCA and java cryptography extension JCE.
- Next move on to the details of the provider BC bouncy castle how to install it and the policies to modify.
- We define few classes in java.security package to provide more security
  - These classes are SSLsocketfactory class and SSLserversocketfactory class.
- These classes are designed with the help of already defined classes that exists in java.net package.

Based on literature survey, a little work is reported in this regard and based on so many problems faced by the existing standards of SSL architecture and hence our new proposed modifications as above mentioned will surely solve those problems up to certain level. Here by we conclude by saying our implementation would serve several web services in improving their way of approach in providing services in an efficient way.

The Functional Requirement document defines the capabilities and functions that a System must be able to perform successfully. The functional specification is meant for a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

- The user can start a new session every time with more secure key messages and certificates generation.
- The client can sit back about the security of his information.
- The server can also set back about the clients' authorization.

In requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions.

In general, functional requirements define what a system is supposed to do whereas non-functional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Non-functional requirements are functionality, reliability, usability, efficiency, maintainability and portability. Define their modules and their functionalities

Two of them are factory classes, and the other two are socket classes.

- SSLSocketFactory class.
- SSLServerSocketFactory class.
- SSLSocket class.
- SSLServerSocket class.

Hence we have now all the information about the requirements about the research we now move on with the next phases of our research implementation.

The Figure 4.3 below provides the class diagram of this implementation using Agile Modeling.

The Figure 4.4 provides the sequence diagram of this implementation using Agile Modeling.

The Figure 4.5 provides the detailed sequence diagram of this implementation using Agile Modeling.

We have now the blue print of the idea with the few important uml diagrams. Hence we move on to the next phase of the research implementation. The goal of this phase is building the target system based on the specifications developed in the previous phases. Transferring the specification algorithms into a programming language, creating and integrating the system components. This is the phase where the software is installed /deployed at the client side and works in the real time environment.

Basic handshake protocol: The Figure 4.6 provides basic handshake protocol.
Message Flow for Authenticated Server, Unauthorized Client Handshake means message no always sent. This is done by the server presenting the client with a certificate chain (the Certificate message) after the client initially connects. The client then uses the end entity certificate to encrypt an appropriate pre-master secret, which it sends back to the server (the ClientKeyExchange message). The pre-master secret is then converted by both ends into a master secret, which provides the bits required for a symmetric key required for doing encryption on the data and the MACs used for verifying the integrity of the data. Both ends start encrypting and communication on the link then proceeds. So do deal with the problem definition we have defined the following code. Two of them are factory classes, and the other two are socket classes used to represent the end points of the SSL connection.

Factory classes are SSLSocketFactory class and SSLServerSocketFactory class. Socket classes are SSLSocket class and SSLServerSocket class.

**The SSLSocketFactory Class**

The javax.net.ssl.SSLSocketFactory is used to create SSLSocket socket objects. The class is an extension of the java.net.SocketFactory class and, in the same way as its parent class. createSocket() method returns a Socket object, which can be cast to SSLSocket if required. getDefaultCipherSuites() method on both classes returns an array of String objects representing the cipher suites that are enabled by default. The default list excludes null ciphers. getSupportedCipherSuites() method on both classes returns an array of String objects representing the cipher suites that are available to be used for SSL.

**The SSLServerSocketFactory Class**

The javax.net.ssl.SSLServerSocketFactory is used to create SSLServerSocket socket objects. It is an extension of the java.net.ServerSocketFactory class and, in the same way as its parent class. createSocket() method returns a ServerSocket object, which can be cast to SSLServerSocket if required. getDefaultCipherSuites() method on both classes returns an array of String objects representing the cipher suites that are enabled by default. The default list excludes null ciphers. getSupportedCipherSuites() method on both classes returns an array of String objects representing the cipher suites that are available to be used for SSL.
to SSLServerSocket. getDefaul() method returns an object providing a default implementation of the SSL. getDeaultCipherSuites() method and the getSupportedCipherSuites() method provide the same information as the methods of the same name on the SSLSocketFactory class. ServerSocketFactory class.

The SSLSocket Class
Like the class it is an extension of, java.net.Socket, objects of the type javax.net.ssl.SSLSocket are not created directly but are created using a factory methods on SSLSocketFactory objects or as a result of an SSLSocketServerSocket.accept() returning. setEnabledCipherSuites() method takes an array of String objects representing the cipher suites that can be used with sockets that result from this server socket doing an accept. The method has a corresponding get() method, getEnabledCipherSuites(), which returns a String array representing what cipher suites are currently enabled. setEnabledProtocols() method takes an array of String objects representing the variations on the SSL protocol that can be used with this socket. The method has a corresponding get() method, getEnabledProtocols(), which returns a String array representing what protocols are currently enabled.

setEnableSessionCreation() method takes a single boolean parameter, which if true, allows the SSLSocket object to create new SSL sessions. If the parameter value is false, then the socket can only be used to resume existing sessions. The method has a corresponding get() method, getEnableSessionCreation(), that returns true if new SSL sessions may be established by this socket. A return value of false indicates the socket can only be used to resume existing sessions. The method has a corresponding get() method, getUseClientMode(), that returns true if connections on the socket will be in SSL client mode, false otherwise. The startHandshake() method starts an SSL handshake on the socket on which it is called. The method will throw an IOException if there is a problem performing the handshake. If a handshake fails, the SSLSocket is closed and no further communication can be done.

The SSLServerSocket Class
The java.net.ssl.SSLServerSocket class is an extension of java.net.ServerSocket, and objects of the SSLServerSocket type are created using factory objects of the type SSLServerSocketFactory. They are used to accept incoming connection requests using the accept() method, which returns Socket objects that can be cast to SSLSocket.

setEnabledCipherSuites() getEnabledCipherSuites() setEnabledProtocols() getEnableSessionCreation() setUseClientMode() getUseClientMode()
Same as SSLSocket class methods.

The Figure 4.8 displays the key generation of this application.

![Figure 4.8: Key generation of this application](image)

After this, if everything goes right and if certificate is valid then new socket is created and a session is started else no new session is started, the old sessions are resumed.

The outputs of the implementation stage include fully functional software that satisfy the requirements and design elements previously documented.

To conclude this implementation, Computer security is a vast topic that is becoming more important because the world is becoming highly interconnected, with networks being used to carry out critical transactions. The environment in which machines must survive has changed radically since the popularization of the Internet. Deciding to connect a local area network (LAN) to the Internet is a security-critical decision. Although software security as a field has much maturing to do, it has much to offer to those practitioners interested in striking at the heart of security problems. The goal of our research implementation is to familiarize with the current best practices for keeping security flaws out. Cryptography and secure systems are fast growing subjects. Our effort with this research implementation is a minute part in this system. Hence there is a wide scope for this research implementation in future.

4.2.3 Web 2.0 Services Security Design for authentication using Agile Modeling
The web services standards are a group of agreements designed to facilitate interaction and provide common protocols in all areas of web services. They are produced by the OASIS organization, www.oasis-open.org, which has a large number of participants including many of the large software companies such as IBM and MICROSOFT.

In this implementation we take a look at WS-Security agreements and use the Microsoft implementation, known as WSE 3.0, in conjunction with visual studio 2005 to produce a web service that requires authentication. There are a number of ways to secure a web service. A simple way might be to use SSL together with client side certificates to ensure that the potential user is someone allowed to call the services method. The problem with this approach would be to pass the user credentials as part of the method call. This can be overcome by using SOAP protocol along with Secure Socket Layer (SSL). In this implementation we want to implement web 2.0 services security using visual studio 2005/2008.

A Web Service is a software entity deployed on the Web whose public interface is described XML (eXtensible Markup Language). It can interact with other systems by exchanging XML-based messages, using standard Internet standard protocols. The web services standards are groups of agreements designed to facilitate interaction and provide common protocols in all areas of web services. There are a number of ways to secure web service. Here we will use SOAP to access the web service and include security related information in the SOAP Header.

In this fast changing world of the web, Security is of paramount importance. There was no particular standard for Web Service security (WS-Security) until April 2004, and Web service developers had to rely on the transport layer to provide security (via SSL/TLS from HTTP) or develop their own custom security mechanism sacrificing interoperability in the process. The whole idea of developing web services is interoperability across all platforms. In April 2004, WS-Security was established as an approved OASIS open standard. Web Service Enhancements (WSE) 2.0 is the first release of the software from Microsoft implementing this approved standard version of WS-Security. This means that applications built with WSE 2.0 security features will interoperate with other Web service platforms as their WS-Security-compliant implementations become available. This implementation shows us how to secure a web service using a User Name and password. This implementation also shows us how to secure the password with a password digest and thwart man-in-the-middle and replay attacks. The User Name and password must be known before hand to both parties (Server and Client).

In the today world Web Services are secured by following methods.

• First way is to use SSL together with Client-Side certificates to ensure a potential user is someone allowed to call the service’s methods. The problem with this approach is the distribution of the certificates, and the fact that not all the clients support them.

• Second way to secure web services is to pass user credentials as a part of method call. If SSL were also used, the username and password would be protected and the method could perform authorization checks before returning the Stock price. The trouble with this approach is that it requires the developer to know in advance all methods that require authentication.

Solution to all the problems listed in the existing system methods is to use SOAP protocol along with Secure Socket Layer (SSL). SSL encrypts the user credentials to provide confidentiality and we will use SOAP to access the web service and include security information in the SOAP header, separate from the data to be passed to the actual web methods. The actual data is separated from the data related to security information because SOAP header will consists of all security related information.

In order to achieve the proposed system qualities we are implementing four modules. They are:

• Creating a Web Service
• Adding Policy
• Adding the Custom Authentication
• Creating Client

The description of modules is as follows:

Creating a Web Service:

• In this module we create a web service as follows. First open Visual Studio and add a blank solution named UserTokenDemo.
  • Right-click the solution in the solution explorer and add a new website. We can choose the file system based one or one that resides in IIS, both will work equally well. Name the Web Service StockQuote.

Adding Policy:

• In this module we add security policy to the web service by using WSE 3.0.
  • First enable the WSE settings and add the SOAP protocol factory, when prompted choose a name such as Serverpolicy and then select secure a service application and select Username as authentication method.
  • Now a policy has been created that the username will be used for authentication, the web service needs to be told to use the Policy.

Adding Custom Authentication:

• At this moment, the service will accept a request with a username and password but will not perform custom authentication.
  • It will actually try to authenticate using Windows; in other words, it will try a log on with
the credentials and unless the details happen to match a user on the machine or a domain user the authentication will fail.

- When we want to authenticate from a known list of users us will have to override the default behavior. There are two steps to do this process. First we have to create a class that inherits from using Microsoft Web Services3.Security.TokenUserNameTokenManager. Secondly this class needs to be registered with this service.

Creating Client:
- In this module we will create and configure a client to use the service. Fire up a new instance of Visual Studio .NET and create a Console Application. Name the research implementation "SecureWSClient". Enable WSE support for this Console Application research implementation by right clicking on the Research implementation in Solution Explorer and selecting the WSE Setting 3.0 options on the context menu. Place a check mark in the first checkbox and click OK. (The second checkbox will be grayed out since this is not an ASP.NET research implementation.)

- Since we enabled WSE support for this console application, Visual Studio generates a proxy which contains an additional object with the name Service1Wse. If there was no WSE support enabled, we would have just had the standard Service1 object.

- UsernameToken is sent as part of the SOAP header. The password is transmitted as a hash using the SHA1 hashing algorithm. This ensures confidentiality and thwarts man-in-the-middle style of attacks.

Figure 4.9 provides the package (being a collection of classes) diagram of this implementation using Agile Modeling.

The figure Figure 4.10 provides the detailed sequence diagram of the implementation using Agile Modeling.

Figure 4.11 to 4.13 provides the screenshot of this implementation, with authentication failure notification.
password

Figure 4.13 On entering wrong username or password the error is displayed as follows, violating the authentication mechanism.

The Web Service Enhancements 3.0 library provides simple and powerful methods to manage message level security in Web Services. The user authentication module can be updated without recompiling the Web Service itself by making changes to the WSESecurity library and redeploying it in the GAC or just copying it into the bin folder of the Web Service. The web service code need not worry about security and can concentrate on business logic.

CONCLUSION

In this paper we discussed about Web 2.0 Services Security Design, using appropriate Agile Modeled Design strategies of Case Studies of AJAX Security, Web 2.0 Services design at SSL and Web 2.0 Services Authentication mechanism.

REFERENCES