Web Application Security

by Peter Joyce
BA in CS
Advisor - Dr. Hauser
April 15, 2014

Table of Contents

Introduction

This capstone project was an examination of the methodology used to provide security to web applications. Over the course of this capstone the focus of information has slowly shifted away from the attacks and exploits that originally attracted my attention to the subject and instead focused on necessary elements of security for an application running over the internet. The first challenge for applications provided a service over the web is providing reliable authentication. Man in the middle attacks can impersonate a server and redirect traffic to intercept and control a

user's session. Cryptographic functions are examined in detail to show how it is possible to provide reasonable amount of authentication using asymmetric cryptography. Then SSL is examined detail as it represents a protocol that performs both authentication and encryption for a secure socket connection. After a user has been authenticated it is the responsibility of the application to manage the users access to server resources. This project looked at 4 different approaches to defining authorization including dynamic systems that take into consideration additional security context in addition to the users credentials to determine authorization policy. Authorization can be maintained by proper session management which ensures that the person using a connection stays the same as the person who established the connection. Finally a few additional security considerations are discussed that may or may not affect a specific web application.

1. Authentication

1.1 Introduction

Authentication can be thought of as the process of identification.[3] When a police officer asks for the drivers license from the driver of a speeding car the first thing the officer does is check to see if the picture on the card matches the person in the seat. This is an example of authentication performed by people in the world. Over the internet the principle of authentication stay the same, but the methods to perform reliable authentication become much more complex.[3] Below is a diagram of a simple authentication between a user and a server.

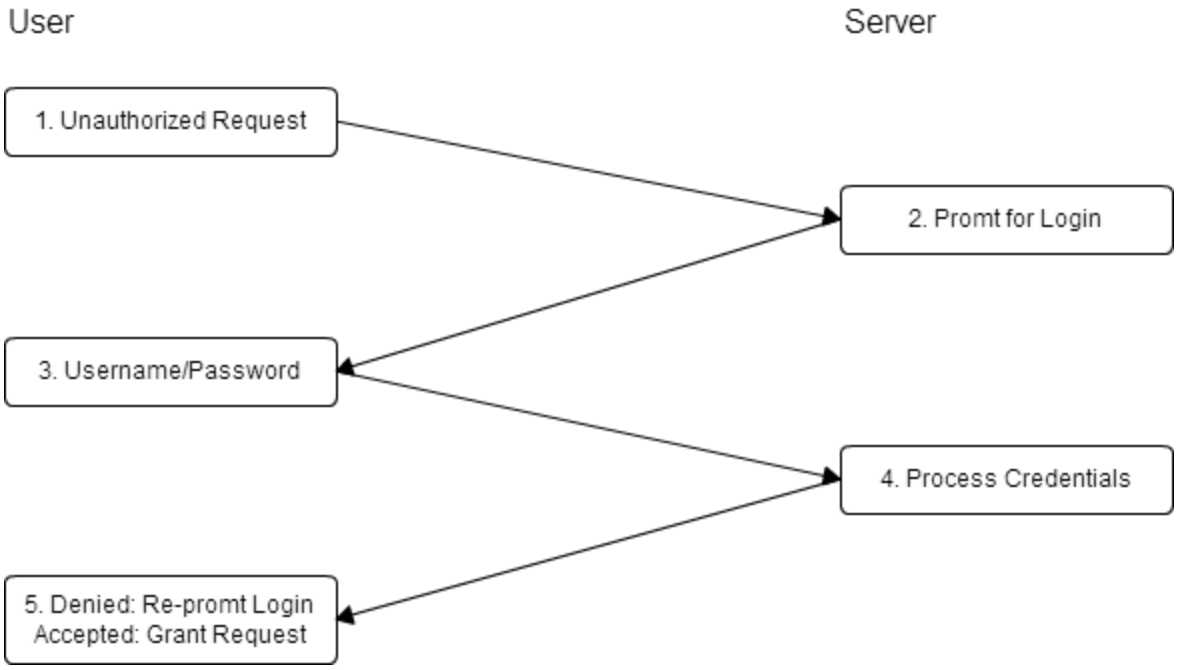

Figure 1

1. The first step in the Authentication process takes place when an unauthenticated user requests a protected resource from a web server.
2. The web server will see that the user has not yet been authenticated and will reply to the user with a request for security credentials, usually a username/password combination.
3. At this point the user will identify themselves by sending their username and password to the server.
4. The server will check to see if that username exists and if it does it will also check to see if the password is correct.
5. If the username does not exist or the password is incorrect the server will reply by sending back the login page with an error message, or if the password is correct the server will present the protected resource to the user.

1.2 Man in the Middle Attack

Man in the Middle (MiM) attacks are a form of eavesdropping, but instead of just 'listening' or recording the traffic over a connection a MiM attack redirects the connection traffic to themselves, and relays messages between the intercepted connections.[9] This can make the user and the server think that they have established a private connection, but in reality it has been compromised.[9] For a MiM attack like the one in the diagram below to be successful the MiM must be able to adequately pretend to be both the user to the server, and the server to the user.

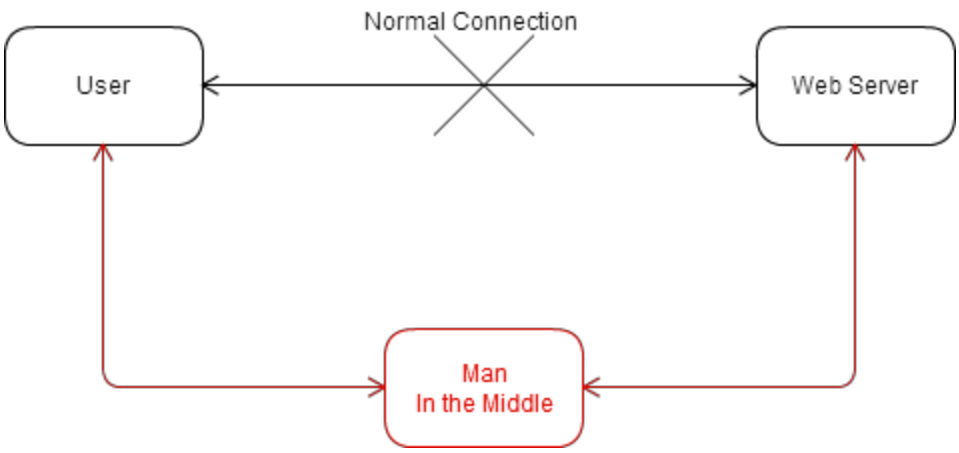

Figure 2

In order to protects against MiM attacks it is necessary to make it difficult for someone to impersonate your Server. Cryptography is an extremely helpful tool to protect against MiM attacks and eavesdropping in general. [5]

1.3 Cryptography

Cryptography is the process of encryption and decryption of private messages so that they cannot be read by third parties. Encrypting information means transforming the information so that it's original contents can only be understood by a specific party. Decryption is the process of translating the encrypted information back to it's original state. The encryption process itself relies on complex mathematical formulas that generate encrypted output based on keys. These keys when combined with a cipher, an algorithm for performing encryption, determine how the text of the message will be encrypted.[1]

There are two types of cryptographic keys Symmetric and Asymmetric. For Symmetric cryptography the key used to encrypt the information is the same key that is used to decrypt the information. For Asymmetric cryptography there are two keys where what is encrypted by one key is decrypted by the other. Public-key cryptography is one example of Asymmetric-key cryptography.[1]

1.3.1 Symmetric Key Cryptography

Symmetric key cryptography can be thought of much the same as a box with a lock on it. If I lock a message into a security box and mail it to another person that person will only be able to read the message if they have a copy of my key to get into the box. This is an acceptable form of encryption, but it relies on the ability to keep the key secret. However if a third party such as an attacker performing a MiM attack where to get a hold of the secret key the encryption will be useless because the attacker will have the key for decryption. Because of this symmetric cryptography is often only used after a securely established connection has been made. This is often done using asymmetric cryptography.[5]

1.3.2 Asymmetric Key Cryptography

The primary difference to symmetric and asymmetric key cryptography is the number of keys used by the algorithm. For symmetric keys the key the encrypts the information is the same key that will decrypt the information. This is not the case for asymmetric keys, instead asymmetric keys come in key pairs. The key pairs function in reverse of each other i.e. what one key encrypts the other decrypts and vice-versa. A common implementation of asymmetric cryptography is the Public/Private key model. [1] In this model the public key can be freely distributed while the private key must remain secret. In this model the public key is used by users to encrypt messages sent to the server, the users can be confident the messages will only be read by the server holding the private key. Knowing this fact the user is able to generate a secret key and pass it encrypted to the server with the public key and the understanding that only the server with the private key will be able to decrypt the secret key. After this point both the user and the server have a shared secret key, and they can switch to symmetric cryptography for the remainder of the connection.[3] This is done because Asymmetric cryptography is more computationally intensive, and it is not necessary after a

secret key has been established. A simplified version of Asymmetric cryptography being used to establish a connection can be seen below. [5]

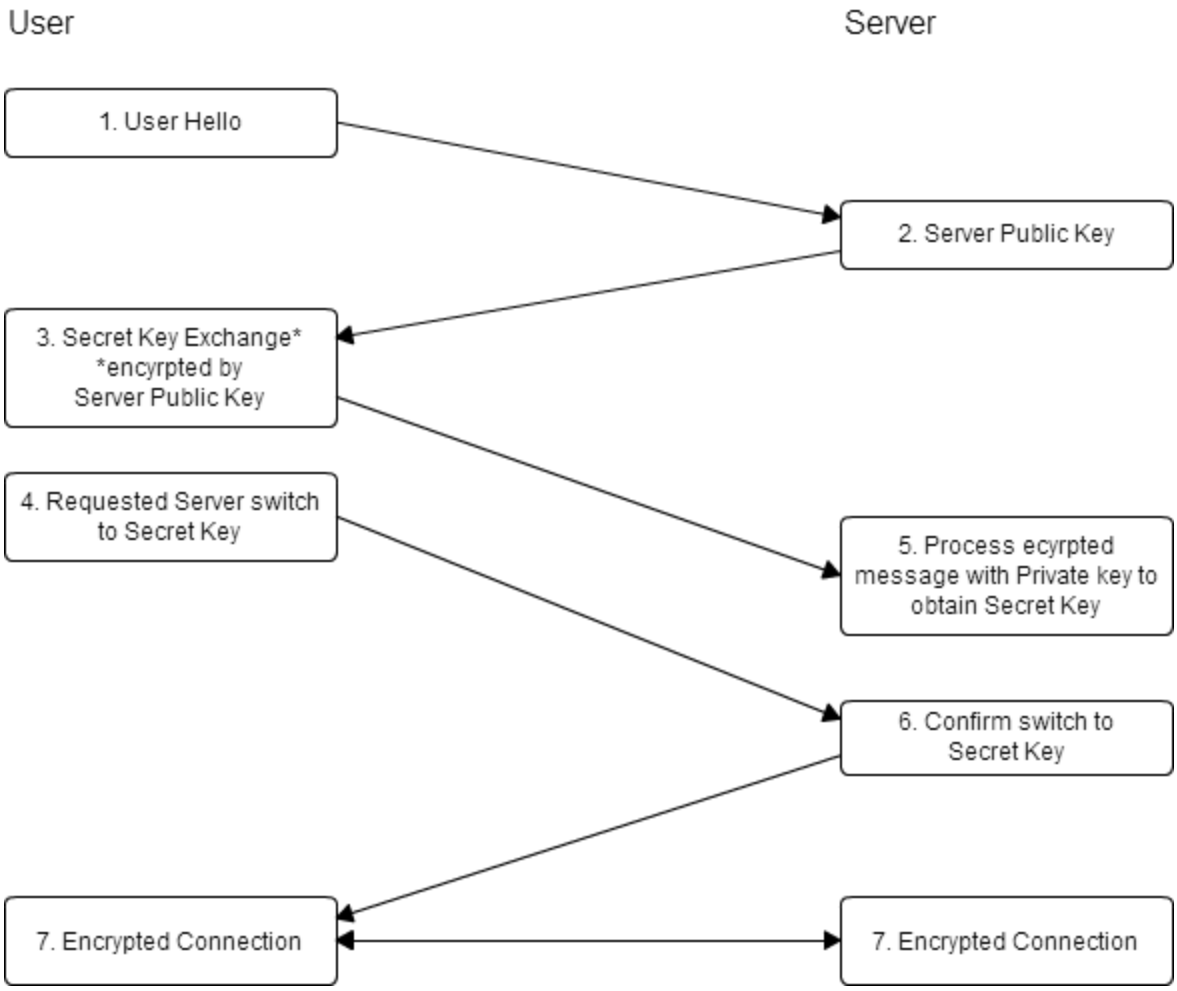

Figure 3

1. First the user must initiate a hello message to notify the server of an attempted connection.
2. The server replies with a message containing their public key.
3. The user encrypts a secret key with the public key sent by the server.
4. The user requests for the server to begin using the secret key.
5. The server processes the encrypted message with private key to obtain the secret key.
6. The server confirms the switch the secret key and a secure connection has been established.

It is important to note that in order for a public key to be reliable a third party must verify that the public key does in fact belong to a specific server. Without this third party verification a MiM attacker could intercept the server's public key and replace it with their own. For connections using the Secure Socket Layer this verification is done by Certificate Authorities. Certificate Authorities issue digital certificates that show ownership of a public key by an

explicitly identified party. This allows users to trust a public key really belongs to who it says it does.

1.4 Secure Sockets Layer (SSL)

1.4.1 Introduction to SSL

The Secure Sockets Layer (SSL) is a commonly used protocol for the authentication and encryption of connections over the internet. It utilizes asymmetric encryption to establish a connection and and symmetric encryption to keep information private after a connection has been made.

There are four elements to the SSL protocol.

- The Record protocol describes the basic format for passing messages via SSL. All SSL messages are sent using the Record protocol.
- The Handshake protocol describes the chain of events required to establish a new SSL connection. This process utilizes digital certificates and public key cryptography to authenticate the server, the server also has the option to request authentication from the user.
- ChangeCipherSpec protocol is the message sent after the secret key has been established to switch the encryption to the new cipher.
- Alert protocol is not passed under normal circumstances, but can be passed by either party to signal a problem with the connection. If a fatal alert is sent or received both parties will immediately close the active connection.

1.4.2 Record Protocol

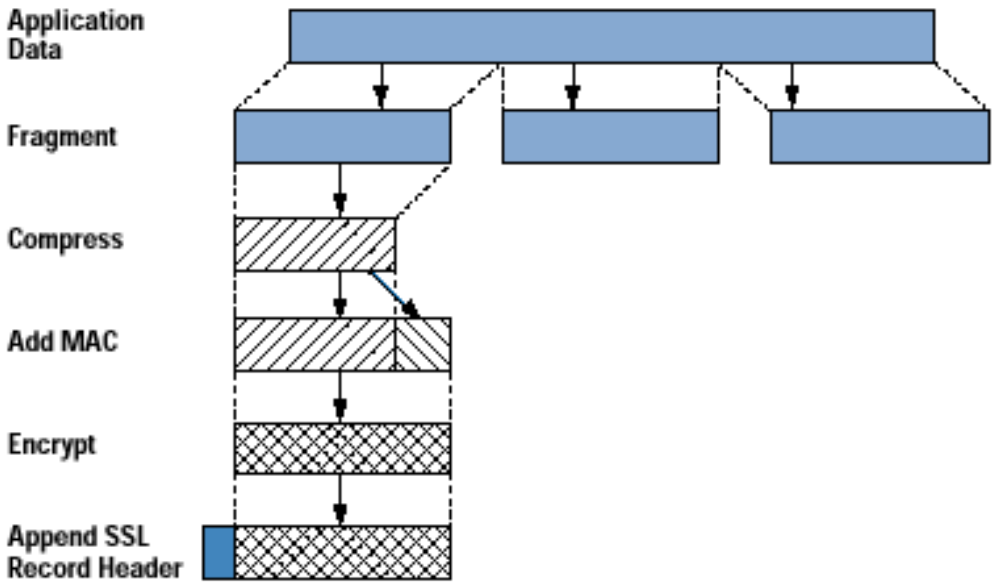

Figure 4 [5]

The record protocol is the basic format for passing messages with SSL. It takes the information to be passed along the connection and breaks it into fragments. Each fragment is compressed, and a method authentication code (MAC) is added. A MAC is short piece of information that ensures the authenticity and integrity of the original message. The message and the MAC are then encrypted and sent across the connection. Below is a diagram of how the MAC encryption happens during the handshake process.

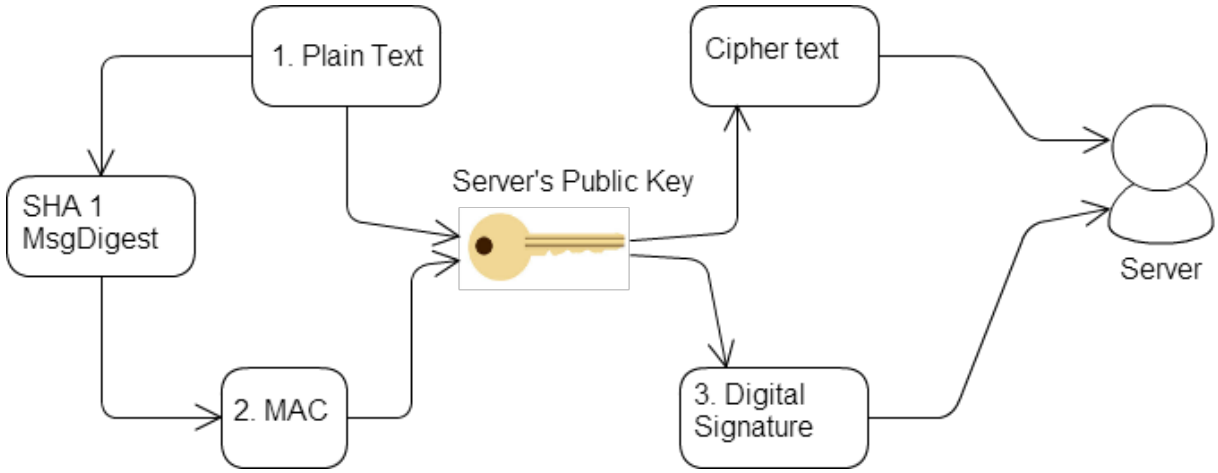

Figure 5

1. The plain text is put through a hash function to obtain the MAC
2. The MAC is a unique hash output based on the plain text input, and can only be generated by putting identical input into the same hashing function.
3. The MAC is encrypted by the Server's Public key and sent to server as a Digital Signature.

The diagram below shows the processing of the message after if has been received by the server.

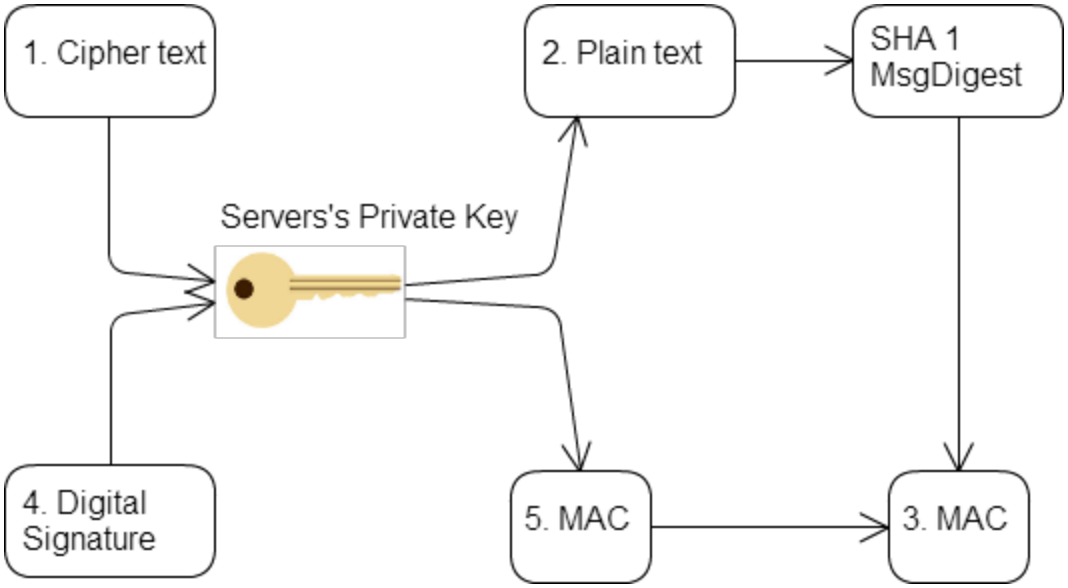

Figure 6

1. First the cipher text of the message is decrypted to Plain text using the server private key
2. The plain text is then put through the same hash function used to obtain the original MAC
3. The server generates an expected MAC for the received message.
4. The digital signature sent by the server is decrypted using the server private key to obtain the original MAC sent to the server.
5. The server compares the expected MAC (3) to the MAC received from the sender (5). If the MAC received as the digital signature matches the expected MAC generated from the message text, the server can validate that the message was not modified in transit.

1.4.3 Handshake Protocol

The handshake protocol begins when the user requests a connection and an internet browser sends a hello message to the server. The hello message actually contains a lot of the information necessary to establish a connection between the user's browser and the server.

- Version Preference

The first piece of information is the SSL/TLS version of the protocol the browser wishes to use. TLS (Transport Layer Security) is a successor to SSL and is backwards compatible with SSL 3.0 and up. [2]

- Random

The next piece of information contained in the hello message is a random or pseudo-random number that is used later along with a random from the server to create the secret key for the connection.

- Session_ID

Additionally if there has been a previous connection in the user's browsers history a session id can be sent to reuse a previously established connection.

- Compression Preference
- Cipher Specifications

The hello message also includes the support compression formats and cipher specifications. It is with the combination of these cipher specification and the cryptographic key that encryption is created. [5]

The server hello replies confirming the browser's preferences or switching them to mutually supported versions. At this point the server sends it's SSL Certificate to the user. The SSL certificate contains:

- The domain name of the server.
- The legal name of the organization.
- City, State or Region, and Country where the organization is located.
- An email address used to contact the organization.
- Server's Public Key

Trusted third party Certificate Authorities (CA)s are responsible for verifying the information contained within SSL Certificates before digitally signing them to be used on a server. [6]

The user's browser then checks its list of trusted CAs to order to verify that it trusts the digital certificate. If the CA is trusted the user's browser will respond with the secret key exchange and the method to signal switching over to the next cipher negotiated with the secret key. The user is now finished with their end of the handshake. The server acknowledges the changeCipherSpec message and replies that it is finished with the handshake and the connection continues using the established connection encrypted by the secret key. Below is a diagram of the SSL handshake from start to finish.

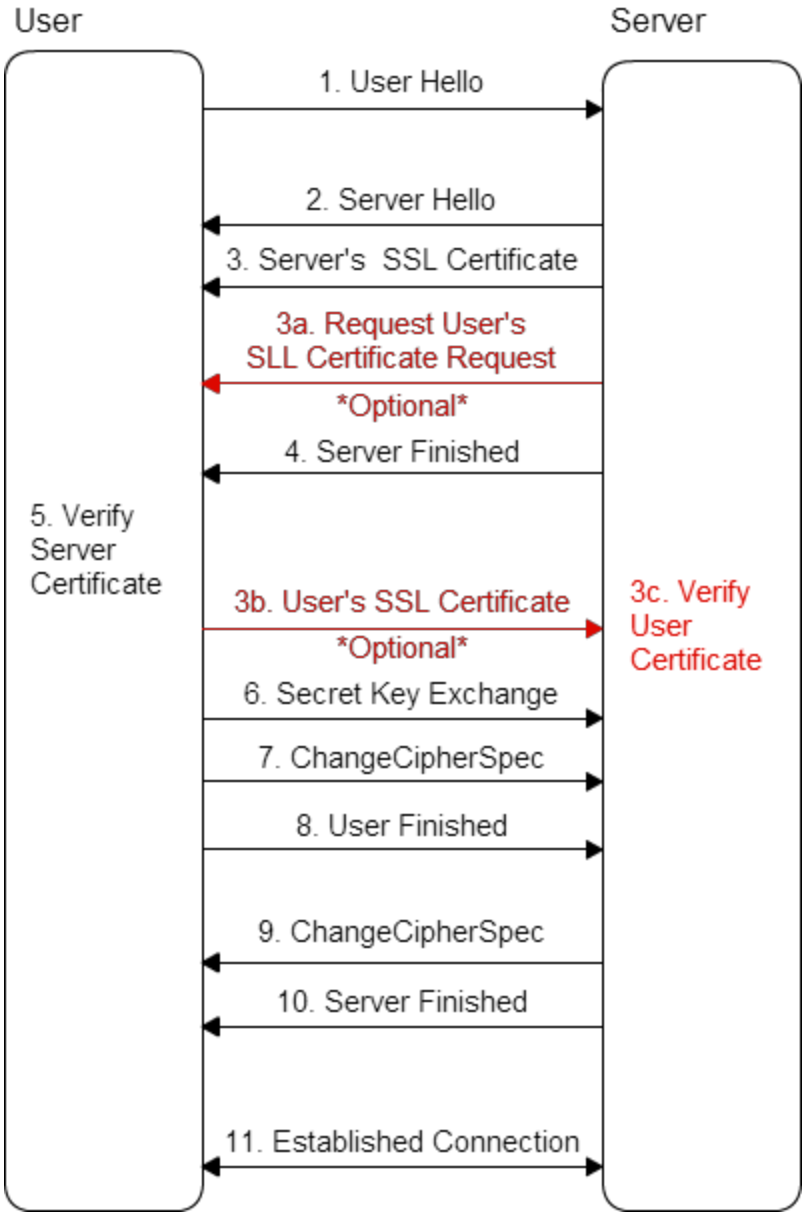

Figure 7 [6]

1. User send a hello message to initiate the handshake.
2. The server replies with it's hello message.
3. The server sends its SSL certificate.
3a. The server can optionally request a SSL certificate from the user
4. Server is finished with setup
5. The user's browser verifies that the certificate was signed by a trusted CA
6. The Secret key is exchanged encrypted with the server's public key from the certificate.
7. The user's browser sends the changeCipherSpec message.
8. User is finished with handshake and is ready to open a connection
9. The server acknowledges the changeCipherSpec message.

10. Server is finished with the handshake
11. On the receival of the changeCipherSpec both the User and the Server switch over to using the secret key for encryption, and a secure connection has been established. [6]

1.4.4 Certificate Authorities

A Certificate Authority (CA) is an organization that stores information about public keys and their owners. When a certificate is received the user's browser already posses a copy of that server's public key from a trusted CA; it can then verify if the key was sent by the server bound to that key. It is the CA's responsibility to assure correctness between an entity and the information present about that entity. [5] This is a difficult task and CA's use government bureaus, economic infrastructure, and third party databases to help authenticate these entities. Despite security measures taken to verify the correctness of entities obtaining certificates there is always the possibility of a single CA issuing a bad certificate being to an imposter. It is also possible to register entities with the same or very similar names, this can lead to confusion especially if one entity is attempting to impersonate the other. [5] The Certificate Transparency Initiative proposes that all certificates should be cataloged in a mandatory public log that would force CA's to publish certifications in public record.

2. Authorization

2.1 Introduction

Returning to the analogy of the police officer requesting a drivers license from a speeding car. If authentication is the officer making sure the driver matches the picture on the license authorization can be thought of as the checks the police officer makes when he returns to the patrol car. The officer checks that the drivers license is not expired, and the officer checks the car to see if it's been listed as stolen. These steps are a way to determine if the person in the drivers seat has the authority to be driving that car. In the same way authorization in a computer system is the process by which the system determines if a user can access, or 'drive', a specific system resource.

There are two steps to successful implementation of authorization. The first of these steps is the Policy Definition. Policy definition is the assignment of permissions for the system. Permissions are assigned based on the Principle of Least Privilege which means giving only the bare minimum permissions required for user to perform their work. These permissions are stored in Access Control Lists that specifies users and operations granted access to the resource.[4]

The second step to successful implementation of authorization is the Policy Enforcement. Policy enforcement is the process by which the system approves or disapproves access requests. This process requires a combination of the system's authentication protocol and the access control lists protecting its resources. This document will examine four approaches to

Policy Enforcement, Resource based, Role based, Task based, and Claims based authorization.[4]

2.2.1 Resource based Authorization

Resource based Authorization is named because the access permissions are granted on the resource itself. In a resourced based approach the server "impersonates" the user's security credentials and passes them along to the backend resource like a database. This approach relies on the backend resources to perform their own access control. While this approach works well for implementing applications where resources are secured individually such as an File Transfer Protocol (FTP) application. It suffers from efficiency because many requests that are destined to fail are still passed through the application only to fail after being passed to the database. It also suffers when the request is for data spread across multiple different sources as is common with most web applications. [4]

2.2.2 Role based Authorization

Resource based Authorization is named because the access permissions are granted on the resource itself. In a resourced based approach the server "impersonates" the user's security credentials and passes them along to the backend resource like a database. This approach relies on the backend resources to perform their own access control. While this approach works well for implementing applications where resources are secured individually such as an File Transfer Protocol (FTP) application.[4] It suffers from efficiency because many requests that are destined to fail are still passed through the application only to fail after being passed to the database. It also suffers when the request is for data spread across multiple different sources as is common with most web applications.[7]

2.2.3 Task based Authorization

A Task based approach to authorization is similar to that of the role based approach, except permissions are assigned dynamically during runtime instead of by predefined roles. This allows the security system to actively use additional contextual information in the decision to grant a user access to a resource. These permissions are assigned to the user in a just in time fashion based on tasks currently in progress along with the task being requested. This allows the server to dynamically manage permissions as the user progresses. A task based approach can record the amount of usage a task receives and the assignment of permissions for a specific task. This can detect and prevent an abuse of permissions by a malicious user. [7]

2.2.4 Claims based Authorization

A claims based approach to authorization relies on a security authority that issues claims about a user's security context. The security authority makes a claim about the permission of a user, and the claim will grant the user permission to use specific resources. Just like in a task based approach claims are asserted in a just in time fashion and can be made to quickly expire

for sensitive processes. Claims are often stored using a security token that can be updated every time the user requests a resource from the server. The security token containing the permissions claim can also be associated with a user's session id. This allows for the application to simplify all of authorization information needed for the user in a single session id attribute. Another advantage of a claims based approach is that if multiple applications both trust the same security authority a user will not have to sign in multiple times as claims from the security authority will be trusted by both applications. [7]

2.3 Session Management

When a user connects to a secured website they must be authenticated in order to verify their identity. However once a connection is established session management allows for the user not to have to re-authenticate every time a resource is requested. Session management makes sure that the user who is connected is the same as the user who was originally authenticated. This is important to protect against session hijacking an attack where an unauthorized user will impersonate an authenticated user by using their session id or cookie to request a resource blocked to unauthorized users. [8]

The main purpose of a session is to maintain state information about a connected user. This is done by associating information about a user's state with a session id tag. This session id tag is then maintained between the server and the user until the user logs out. Session id's are often implemented with HTTP cookies. An HTTP cookie is a packet of information sent between the server and the browser every time they communicate. Cookies can contain any arbitrary information the server chooses and are used to maintain state between the server and the browser. [8]

2.3.1 Session Management Flowchart

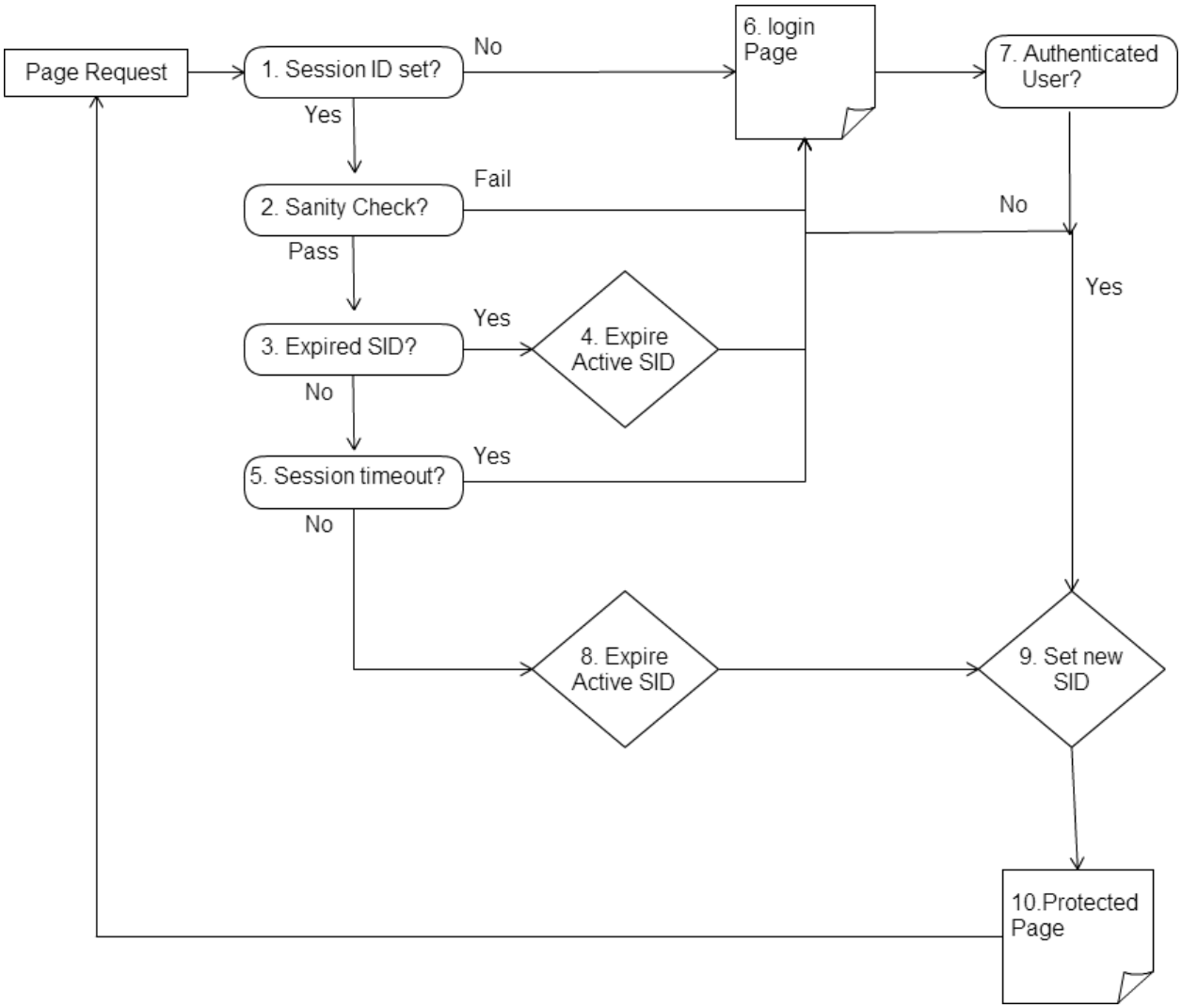

Figure 8 [8]

1. If the user does not have a session identifier (SID) set they are not logged into the system and the user will be forwarded to the login page.
2. Sanity checks attempt to determine if the session could be an attempted session hijack. Sanity check are not limited to but can include SSL certificate checks and some level IP address checking.
3. Determine if the SID being passed is the most recent version of the SID or an older SID.
4. If an older SID is being used it is most likely an attempted hijack, and the session with expire the active SID in addition to redirecting the user to the login page.
5. Has the session timed out due to inactivity? Note that session timeout is different than an expired SID from part 3. Expired SIDs are actively discontinued everytime a user access a resource. Session timeouts on the other hand expire an SID after a specified period of inactivity by the user.

6. The login page is the main authentication page and can be accessed directly or indirectly when the user requests a page without a valid SID.
7. Authentication is typically handled by a username/password combination provided by the user.
8. If a SID is in fact the active SID for the session that SID is expired at this time.
9. A new SID is set containing updated state information for the user this will include an update to the user's security permissions.
10. The protected resource is presented if the user has appropriate security permissions or the user is redirect to the login page if they do not.

2.3.2 Security Checks for Session Management

1. Remember to use encryption.
With encryption from a protocol like SSL the online system will be almost completely exposed to anyone attempting an attack because the data will be sent using plain text leaving the server vulnerable to a simple man in the middle attack. [7]

2. Only store SID on the user browser.
The server should maintain the session and state information for the user. It is especially important not to store a username or password in the session cookie because if a session hijack is successful it will expose the user's login credentials and create more vulnerabilities for the server. [7]

3. Perform sanity checking
There are two typical sanity checks typically used to check for an attempted session hijack. The first is by examining the user-agent information in an HTTP response including the browser information and information about the underlying operating system. The user-agent string is fairly stable and can be used to detect session hijacking, but is is also quite predictable and therefore not entirely secure. The second is by checking the network portion of the IP address of the connection. IP address checking must be handled with extra care to not lock out legitimate users. Internet service providers often use proxy servers to fill user requests, and each of these servers could show a different IP address for the user. [7]

4. Dynamic SIDs
Frequently expire old SIDs and generate new ones. One way of doing this is by reissuing the SID on every page request. This considerably disrupts session attacks because the SID lifespan will be short when the user is moving between pages. [7]

3. Additional Security Considerations

Authentication and authorization are just the beginning foundations for a secure web application. As the complexity of the application increases the exposed surface area of the application also increases. These are a few general security considerations that might arise as application complexity increases. [9]

3.1 Input and Data Validation

Improper input and data validation can lead to SQL injections or Cross Site Scripting attacks. If the user is able to input information the server must make sure the user input is valid and safe. A server must filter, scrub or reject user input before doing additional processing. For example html tags like "< >" should be escaped so that it will not display as code in the browser. [10]

3.2 Configuration Management

At it's worst poor configuration management can lead to unauthorized access to administrative interfaces. The exposure of plain text configuration secrets is also a major vulnerability of bad config management. Issues can range from forgetting to reset the default administration password to flaws in the administration of permissions within application logic. Using over privileged accounts can also be considered a configuration error because the principle of least privilege has not be upheld. [10]

3.3 Exception Management

Exception management is how the server reacts when the application fails. When an application crashes user friendly error messages should not reveal too much information about the exception. It can be useful to print exception information when doing error checking and debugging, but it can be dangerous to do so if it can present sensitive information about the server's code. Poor exception management can increase the potential risk of deliberate denial of service attacks (DDos). If a malicious user can repeatedly exploit an exception bug that crashes the application it could be used as a means to crash the application continuously for all users. [10]

Summary

First we examined the necessity of encryption and authentication by observing a vulnerability to man in the middle attacks. We took a close look at how the SSL protocol performs these functions to establish a secure connection. After a user is authenticated authorization becomes the next requirement for securing application resources. We examine a number of approaches to authorization and notice that proper session management can be a great boon for creating a secure authorization policy. With encryption, authentication, authorization, and session management established we have the beginnings of a secure web application. However as the web application increases in functionality and complexity, many more vulnerabilities are exposed which are spoken of in a general context because not all vulnerabilities can be foreseen until the functionality of a web application has been established. It is hoped that with a secure foundation of authentication and authorization additional security considerations can be handled as they arise in an application specific manner.

Glossary

Access Control Lists - Access Control Lists specifies the users and the operations granted permission to access to the resource.

Asymmetric key cryptography - A key algorithm that encrypts and decrypts data using a key pair e.g. Public/Private key cryptography

Authentication - The process of identifying a user over the internet. Ensuring that an entity is who they say they are.

Authorization - The process for determining if an authenticated user has been granted access to the resource they are requesting.

Backend Resource - A backend resource can be thought of as any resource accessed by a server that is not a part of the middle tier (application logic) or the front end resources (public server pages). This could be a simple database or another server providing a web service.

Browser - A browser is a computer program used for accessing sites or information over a network such as the Internet.

Certificate (SSL) - Data files that digitally bind an organization's information with the server domain, and a public cryptographic key.

Certificate Authority (CA) - A Certificate Authority (CA) is an organization that stores information about public keys and their owners. The CA is responsible for verifying the correctness of certificates before digitally signing them.

ChangeCipherSpec - ChangeCipherSpec protocol is the message sent after the secret key has been established to switch the encryption to the new cipher.

Cipher - Ciphers are algorithms that take a cryptographic key as a parameter and are used for performing encryption or decryption of message data.

Claims based Authorization - A claims based approach to authorization relies on a security authority that issues claims about a user’s security context. The security authority makes a claim about the permission of a user, and the claim will grant the user permission to use specific resources.

Configuration Management - Configuration management refers to how an application administered, how permissions are assigned, and how are these settings secured.

Cryptography- Cryptography is the process of encryption and decryption of private messages so that they cannot be read by third parties.

Eavesdropping - The act of discretely listening in on a private conversation, typically between two hosts across a network.

Encryption - A method for encoding messages so that only authorized entities can read them.

Exception Management - Exception management refers to how an application responds when the program fails. Does it fail gracefully or expose a security vulnerability?

File Transfer Protocol (FTP) - Standard protocol used to transfer computer files from one host to another over a TCP connection.

Handshake Protocol - The Handshake protocol describes the chain of events required to establish a new SSL connection.
HTTP - Hypertext Transfer Protocol is the foundation for web communication. It behaves as a request and response protocol for a user and server to communicate.

HTTP cookie (cookie)- A cookie is a packet of information sent between the server and the browser every time they communicate. Cookies contain arbitrary information from the server about a user's session.

Input and Data Validation - Input validation refers to how an application filters, scrubs, or rejects input before additional processing.

Method Authentication Code (MAC) - A MAC is short piece of information that ensures the authenticity and integrity of the original message.

Policy Definition - Policy definition is the assignment of permissions for the system.

Policy Enforcement - Policy enforcement is the process by which the system approves or disapproves access requests.

Principle of Least Privilege - Giving only the bare minimum permissions required for a user to perform their work. This protects resources from being exploited by users who do not need access to those resources.

Record Protocol - The Record protocol describes the basic format for passing messages via SSL.

Resource based Authorization - Authorization where access permissions are granted by the backend resource being accessed.

Role based Authorization - In Role based authorization users are mapped to predefined 'roles' which have predefined permissions to access resources.

Sanity checking - Sanity checks attempt to determine if the session could be an attempted session hijack. Sanity check are not limited to but can include SSL certificate checks and some level IP address checking.

Secure Sockets Layer (SSL) - A cryptographic protocol designed to implement secure communications over the internet.

Session Identifiers (SID)s - A piece of data used over a network to identify a session.

Session Management - Session management makes sure that the user who is connected is the same as the user who was originally authenticated. This includes keeping track of a users session from the initiation of the connection to the time when the user logs out.

Symmetric key cryptography - A key algorithm that encrypts and decrypts data using a single key

Task based Authorization - For Task-based access control authorization constraints are specified for tasks permissions. This is an active security system because it takes into account tasks currently in progress when evaluating a user access request.

Transport Layer Security (TLS) - The successor to SSL further improving on security exploits found in earlier versions, the most secure version is TLS 1.2, but TLS is backwards compatible down to SSL 3.0.

Annotated Bibliography

[1] "Public-key cryptography," *Wikipedia, the free encyclopedia*, May-2014. [Online]. Available: http://en.wikipedia.org/wiki/Public-key_cryptography.

[2] F. A. and K. P., "The Secure Sockets Layer (SSL) Protocol Version 3.0,"*Request for Comments: 6101*, Aug-2011. [Online]. Available:http://tools.ietf.org/html/rfc6101.

This reference was the RFC 6101 historical documentation for SSL version 3.0

[3] D. Richard, "An Overview of Different Authentication Methods and Protocols ," *SANS Institute InfoSec Reading Room*, Oct-2001. [Online]. Available: https://www.sans.org/reading-room/whitepapers/authentication/overview-authentication-methods-protocols-118.

This reference gave an overview of authentication and the different protocols that can be used to authenticate in a network environment.

[4] M. J.D., M. Alex , D. Michael, and V. Srinath, "Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication," *Microsoft Developmer Network*, Jan-2006. [Online]. Available: http://msdn.microsoft.com/en-us/library/ff649350.aspx.

This reference showed detailed implementations of Role based and Resource based Authorization.

[5] O. Tomasz, "Authentication, Access Control & Encryption: Secure Socket Layer," *WindowSecurity.com*, Jul-2002. [Online]. Available:http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Secure_Socket_Layer.html.

This reference is a description of the SSL protocol. This source provides information on the information within specific messages of the SSL including the hello message. Along with an excellent breakdown of the handshake.

[6] S. William, "SSL: Foundation for Web Security," *The Internet Protocol Journal - Volume 1, No. 1*, 0000-00-00 (nd). [Online]. Available:http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html.

This reference is a description of the SSL protocol, but this source provided additional information about cipher suites and specifics about the connections that are established during the handshake protocol.

[7] Z. Lang, "A Role-based Access Control Security Model for Workflow Management System in an E-healthcare Enterprise," *Department of Computer and Information Sciences Florida Agricultural and Mechanical University*, Nov-2008. [Online]. Available:http://www.cis.famu.edu/~hchi/langzhao_Thesis_final1.pdf.

This reference was a Master's thesis paper about Access Control for E-Healthcare. It contained information regarding advantages and disadvantages of different authorization approaches, and information about task based authorization and security context.

[8] M. Luke, "Secure Session Management: Preventing Security Voids in Web Applications," *SANS Institute InfoSec Reading Room*, Jan-2005. [Online]. Available:

http://www.sans.org/reading-room/whitepapers/webservers/secure-session-management-preventing-security-voids-web-applications-1594?show=secure-session-management-preventing-security-voids-web-applications-1594&cat=webservers.

This reference contained a majority of the information about session management including basic overview of good session management and key pitfalls to avoid when doing session management.

[9] "OWASP Top Ten 2013 Project," *Open Web Application Security Project*, Jun-2013. [Online]. Available:https://www.owasp.org/index.php/Top_10_2013-Table_of_Contents.

This Reference contained most of the information regarding exploits and vulnerabilities. Present in web applications. It also defined some key security principles necessary for secure applictions.

[10] Meier J.D., M. Alex , and W. Blaine , "Cheat Sheet: Web Application Security Frame," *Microsoft Developer Network*, May-2005. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms978518.aspx.

This reference grouped vulnerabilities, attacks, and countermeasures into more general security categories.