Security Threats Against Secure Sockets Layer (SSL)

by

Nancy Smith

A research paper submitted in partial fulfillment of the requirements
for CISD765

Graduate School of Computer and Information Sciences
Nova Southeastern University

2010

An Abstract of a Research Paper Submitted to Nova Southeastern University
in Partial Fulfillment of CISD765

Security Threats Against Secure Sockets Layer (SSL)

by
Nancy Smith

April 2010

SSL is one of the most common protocols used for secure communication over the internet. Users have grown to trust SSL and are prone to overlook visual cues that could indicate a security breach. But SSL is vulnerable to man in the middle attacks. This survey shows that a MITM attack requires three phases to succeed. First, the attacker must be placed between the client and server by diverting the communication path. Second, server authentication must be circumvented to allow the attacker to impersonate the server. Third, the attacker must visually fool the user with fake browser content so the user will provide sensitive information.

This survey selects papers from these three areas as a sampling of the types of current research problems. It is by no means an exhaustive list, nor does it go into great detail on any particular topic. The goal is to provide an overview of the security issues intrinsic to SSL by categorizing the diverse and seemingly unrelated problems into three phases: diverting the communication path, circumventing authentication, and web spoofing. Understanding the relationship of these security issues may help researchers focus on areas to make SSL more secure.

Table of Contents

List of Figures

Figures

Chapter 1

Introduction

Statement of the problem

Secure Sockets Layer (SSL) is designed to provide endpoint authentication and communication encryption over the internet. Web-based applications rely on SSL to provide security in sensitive transactions such as home banking and e-commerce. There is much recent research analyzing attacks against SSL. SSL is vulnerable to man in the middle (MITM) attacks, where the attacker is in the path between the client and server. Once in this position, the attacker may be able to gain access to sensitive information. These attacks fall broadly into two categories: using fake certificates to impersonate the server, or injecting evil scripts to violate the same-origin policy that browsers rely on for security.

Users have grown to trust that the SSL protocol keeps their communication secure. The average user pays little attention to warnings and visual cues that could warn when a man in the middle attack is occurring. But some attacks are so sophisticated that even alert users can be fooled. Fortunately, the credit card companies also play a large role in making e-commerce secure. Since the user has a limited liability in fraudulent charges, the credit card companies have long established techniques for authorizing charges and detecting possible fraud (Bisel, 2007). However, other types of critical data may be more vulnerable. By understanding the nature of the attacks, steps can be taken to minimize the risk of attack.

The following section will provide background information that will be useful in

understanding the SSL threats. Then the components of a MITM attack will be examined to provide a framework for the more specific examples later on.

SSL Background

The most recent version of SSL is Transport Layer Security. It is an Internet Engineering Task Force (IETF) standards track protocol covered in the Request for Comments (RFC) 5246. The security threats described in this paper are also applicable to TLS.

The SSL protocol sits between the application and transport layers. An application, such as HTTP, would call the SSL API instead of the TCP interface. The SSL session is established with a handshake between the server and the client as shown in Figure 1. The client initiates the session by sending a ClientHello message with the ciphersuites supported by the client. The server responds with a ServerHello identifying the strongest ciphersuite supported by both parties, and the server's certificate. The client application authenticates the certificate and generates a random number called the pre-master key. The client encrypts the pre-master key with the server's public key and sends it to the server. The server decrypts the pre-master key with its private key. Both parties use the pre-master to generate the session key. At this point, the client and server exchange the ChangeCipherSuite message to indicate that all future communication will be encrypted with the session key. Finally, both parties send a Finished message with a message authentication code (MAC) of previous messages (Stallings, 2005).

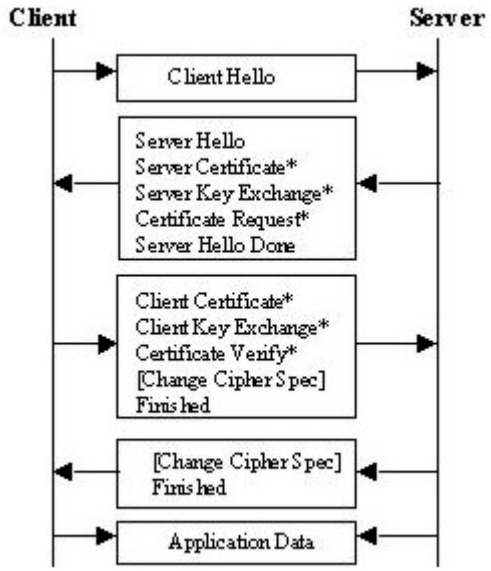

Figure 1: SSL Handshake

* optional client authentication

SSL relies on the concept of using X.509 digital certificates to authenticate the server and its public key. The client communicates with a certificate authority (CA) to validate the server's identity and public key. CAs communicate with a Domain Name Server (DNS) to validate identity. Browsers have a list of trusted CAs pre-defined. Clients are also able to install self-signed root certificates.

While server authentication is mandatory, client authentication is optional and rarely used in practice. Users are reluctant to accept the burden required by client authentication by a certificate authority, and so username-password authentication is more commonly used instead.

Man in the Middle (MITM)

A MITM attack occurs when an attacker is able to divert the communication between the client and the server to pass through the attacker's server. The attacker can then act as a relay, and defeat the authentication process or inject malicious scripts. The

client and server believe they have a secure connection between them, when in reality they are each communicating through the attacker.

There are several methods of diverting the communication path. Emails may contain fake links to a web page with a similar domain name to what the user is expecting. For instance, instead of microsoft.com, the user may be diverted to micr0soft.com or microsoft.org. Another method that is used in environments that rely on the address resolution protocol (ARP) is for the attacker to inject fake ARP messages to inform the client that the attacker is the gateway, and inform the gateway that the attacker is the client (Chomsiri, 2007). Yet another method is to answer web-proxy-auto-discovery queries in such a way as to establish the attacker as a proxy (Chen, Mao, Wang, & Zang, 2009).

This survey covers three methods of bypassing server authentication once the MITM is in the path. The attacker can use a valid certificate with a similar name to the real server. In this scenario, the client authenticates the attacker's certificate and uses the attacker's public key (Oppliger, Hauser, & Basin, 2006). Another method is tricking the user into installing a self-signed root certificate. Once installed, the browser trusts these certificates automatically and the user generates the pre-master-secret using the public key of the fake certificate (Huawei & Ruixia, 2009). In either case, the attacker decrypts using his private key, and re-encrypts using the server's public key before relaying the messages to the server. In the case of a proxy, the attacker may use certificates cached in the browser through the clever use of error and redirect messages (Chen et al. 2009).

After establishing himself as MITM, the attacker may intercept sensitive information and divert it to a third party. In some cases, evil scripts can be substituted for

legitimate scripts from the server to prompt the user to reveal the desired information. In the most extreme case, the attacker may launch a browser spoof in which a fake browser window is drawn on top of the existing browser (Qi, Tang, & Wang, 2008).

Another class of MITM attack that does not require fake certificates from the MITM exploits security holes in the browser's implementation of SSL. Browsers rely on the same-origin policy to prevent JavaScripts from accessing methods or cookies that do not originate from the same domain. The MITM acting as a proxy can insert evil scripts using 3xx and 5xx messages to trick the browser into running the script in the domain of the legitimate SSL session (Chen et al. 2009).

Relevance

SSL has gained widespread acceptance throughout the internet. E-commerce is conducted over SSL and enjoys a very low fraud rate. However, this gives users a false sense of security since the success of e-commerce is due largely to the efforts of the credit card companies, rather than to the security of SSL (Bisel, 2007).

SSL is vulnerable to MITM attacks. Sensitive information such as banking, medical, voting, and proprietary corporate data are increasingly entrusted to what is widely believed to be a secure encrypted channel.

It is important to understand the nature of the MITM threat so the protocol can be made secure against MITM. Moreover, users can be made aware of the best practices that can minimize their exposure to MITM attacks.

Barriers and issues

SSL relies on many independent distributed components across the internet. Many of the underlying protocols were developed before security became a concern and lack the means to authenticate transactions. Even the most recent and secure components, such as browsers and certificate authorities, are constantly plugging the latest security holes. Moreover, the need to be backward compatible with components that do not support the latest security standards undermines the attempt to plug well-understood holes. For example, the original DHCP protocol has no security provisions. An RFC for DHCP authentication was accepted and widely implemented. However, since a large number of clients and servers do not support this authentication, servers are forced to accept clients without this support. Security is an evolving process.

Limitations and delimitations of the study

This survey focuses on the technical details of three aspects of a man-in-the-middle attack:

1. Diverting the communication path.
2. Circumventing the authentication process.
3. Web spoofing.

Research papers from each of these areas will be presented, including details on the threat and the proposed solution.

Definition of terms

Certificate Authority: An entity that issues digital certificates for use by other parties.

Man in the Middle: An entity that makes independent connections with two parties, relaying information between them without their being aware of its existence.

Pharming: Redirecting traffic to a different website.

Phishing: Criminally fraudulent process of acquiring sensitive information.

Web Spoofing: Sending web pages or scripts from a fraudulent website to impersonate an actual website.

Summary

SSL provides endpoint authentication and encryption over the internet. It is widely used for secure communication, especially in the area of e-commerce. However, SSL is vulnerable to MITM attack.

SSL begins with a handshake between the client and server to perform authentication and to establish a session key. Server authentication is done with the use of X.509 certificates. Authentication relies on components in the browser, CA, and DNS to operate flawlessly. Client authentication typically relies on passwords.

A MITM is able to divert the communication path so the traffic between the client and server passes through the attacker's server. The MITM will typically trick the client into authenticating the MITM rather than the server. Other attacks trick the browser into running an evil script in an established SSL session. In either case, the attacker is able to obtain sensitive information.

It is important to understand the details of the MITM attacks in order to develop enhancements to the protocol. Moreover, these details can give insight into the best practices that can be used to minimize exposure to MITM attack.

Chapter 2

Review of the Literature

The theory and research literature specific to the topic

The literature is rich in research that zeroes in on particular security flaws and solutions for the given scenario. In an effort to categorize these scenarios, this section will group the research under diversion, authentication issues, and web spoofing. The technical details of the attacks will be examined, as well as some of the proposed solutions.

Diverting the Communication Path

An IP connection begins at the browser goes through the gateway router and some number of intermediate routers, possibly to a proxy, before finally ending up at the destination server. The diversion attack occurs very close to the browser, while the browser is searching for its destination. This can take the form of fake address resolution protocol messages (ARP poisoning), DNS spoofing (pharming), fake hypertext links (phishing), or fake WPAD responses.

ARP Poisoning

ARP poisoning, or ARP spoofing, is possible in networks that rely on ARP to associate the IP address to the local hardware address or Media Access Control (MAC) address. ARP is common in local area network (LAN) environments and WiFi networks. It is a link-layer protocol that uses a simple request and response messages without any means of validation. The gateway will broadcast an ARP request for a particular IP address that is either missing or expired from its cache. The attacker responds to the

request with an ARP reply containing its own MAC address. Moreover, the attacker can send an unsolicited, or gratuitous, ARP reply to the intended host, associating the gateway's IP address with the attacker's MAC address. If an attacker is able to connect to a trusting LAN and monitor the IP traffic, it can divert the communication path with two fake ARP messages.

Chomsiri (2007) provides detailed information on freely available ARP and DNS spoofing software that can be downloaded from the internet, including the steps and syntax used to launch a MITM attack. Chomsiri concludes that this attack is best thwarted by techniques to prevent ARP poisoning: Static ARP, ARP Watch software, and Anti-Sniff software. Static ARP can be used on switches that support the Port Secure feature, and requires an administrator to manually change the ARP table entries. ARP Watch is a program that detects changes in the ARP table and warns the administrator when changes occur. Anti-Sniff is software that detects machines that are operating in promiscuous mode. This program alerts the administrator of machines that are running data capturing software such as Ethereal.

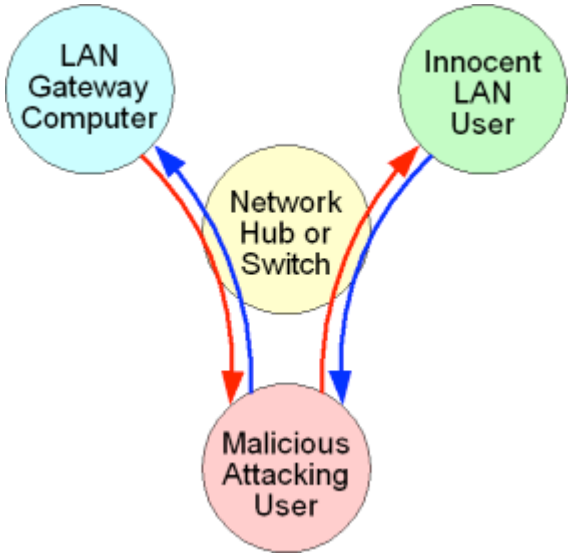

Figure 2: LAN Diverted Communication Path

Domain Name System Spoofing

DNS is used to map host or domain names to IP addresses needed for routing. Typically, a DNS query is sent to a sequence of DNS servers to obtain the address for a symbolic name. The communication between DNS servers is neither authenticated nor encrypted. DNS spoofing, or DNS cache poisoning, occurs when a DNS server queries the attacker's DNS server, and the attacker adds false data to the reply messages (Fetzer, Pfeifer, & Jim, 2005).

DNS servers also support online updates because many entries rely on DHCP server assignments. The integrity of these updates is not validated and is another source of DNS cache poisoning (Fetzer et al. 2005).

A secure version of DNS has been under development for over ten years, but due to the difficulty and cost of deployment, it has not been implemented on a wide scale. There is a coordinated effort between the Internet Corporation for Assigned Names and Numbers (ICANN), the U.S. Commerce Department's National Telecommunications and Information Administration (NTIA), and VeriSign, Inc. to roll out DNSSEC in 2010.

Email Spoofing

Perhaps the best known method of directing traffic to an attacker's server is placing false hypertext links in emails impersonating a site such as a bank or the Internal Revenue Service. Header fields in the email can be faked to appear to come from a source other than the attacker. If the user clicks on the fake link, they are directed to a fake website that impersonates the actual website and attempts to collect sensitive

information. The fake website may even install malicious software such as keystroke loggers to collect information after the connection is terminated.

Once a user is connected to a forged website, it can be difficult to detect the deception. Email links must never be clicked to protect against this attack.

Web Proxy Auto Discovery

Browsers use Proxy Auto Config (PAC) scripts as an alternative to manual proxy settings. The PAC script provides proxy settings for a given URL and hostname. In order for users to be able to browse the web from different networks without changing their configuration, the WPAD is usually enabled. WPAD requests the URL of the PAC script from the DHCP server, and if there is no response, from the DNS servers. Upon receipt of the URL, the PAC script is obtained and the browser configures its proxy settings. An attacker can sniff the network traffic for WPAD queries and respond with the URL of an evil PAC script, setting up an evil server as the proxy. Since the attacker is typically positioned closer to the user, the evil response will be received first, and any legitimate responses are discarded as duplicates since they have the same TCP sequence number (Chen, Mao, Wang, & Zhang, 2009).

Most browsers enable WPAD by default. The WPAD option must be disabled to protect against this intrusion.

Authentication Issues

Authentication is crucial to a secure connection. The three techniques of circumventing authentication in this survey are using fake certificates with a similar name, tricking the user into installing a self-signed root certificate, and using a cached

certificate.

Similarly Named Certificates

It is possible for an attacker to obtain a valid certificate with a similar name to the server in a targeted attack. For example, one attacker used a valid certificate for www.mountain-america.net to spoof the website of www.mtnamerica.org. In such cases, it difficult for users to recognize the attack even if they are paying attention to URL addresses (Opplinger, Hauser, & Basin, 2006).

Preventing such attacks requires modifying the SSL protocol. One such proposal is to make the authentication depend on combining user authentication with the SSL session establishment. It adds a user authentication code (UAC) based on information from the SSL session state that is cryptographically hard to alter. The MITM can no longer simply retransmit it. The user's CertificateVerify contains a hash of all previously exchanged messages, including the server's certificate. The UAC is a function on the encrypted hash and the PIN. The server knows the function, the PIN, the hash, and can validate the user. This mechanism is a lightweight alternative to using client certificates (Opplinger et al. 2006).

Self-Signed Root Certificates

A root certificate is a special kind of certificate that is often self-signed. They may either be pre-installed in the browser or be installed by the user. If the user chooses to install a root certificate, it is not strictly verified. An attacker can generate a self-signed root certificate with issuer information from a trusted CA, but its public key belongs to the attacker. If the user can be tricked into installing the certificate, it will

automatically be trusted by the browser (Huawei & Ruixia, 2009).

To protect an SSL session from evil root certificates in the browser, the SSL protocol must be modified. Huawei et al. (2009) propose adding credible data to the SSL handshake. Before the user begins the SSL connection, a mobile phone is used to contact the server to request a random session number. The server sends this random session number by a short message. The user's phone number is sent in the ClientHello message. The user is prompted to enter the password after receiving the server's certificate. The browser uses the random session number, two additional random numbers, and other key materials to compute the master key, key-block, and session keys. The server performs the same operations.

Cached Certificates

Chen et al. (2009) demonstrated that an evil proxy is able to fake a login page by taking advantage of a cached certificate from a previous SSL handshake. In this scenario, the browser issues a request for https:://www.paypal.com. The proxy intercepts this request and returns an HTTP 502 with a meta element and an img element. The meta element starts a timer and the img element requests an image from https://www.paypal.com which issues another SSL request. This time the proxy allows the handshake to succeed. When the timer expires, the browser is redirected to https://www.paypal.com. The proxy intercepts this request and returns another HTTP 502 that contains a fake login page. When the browser renders this page, it uses the cached PayPal certificate and displays it on the address bar. The authors call this the perfect GUI spoof. The attack occurs in only one window and does not use scripts. Even if the user closes the browser and opens a new window and uses a bookmark, they still

get the faked page.

To protect against this attack, the browsers must modify the interaction between the GUI and the certificate cache. The authors notified the browser venders of the security flaws. Only Firefox was immune to this particular security hole.

Web Spoofing

Once the user is connected to the MITM, web spoofing techniques are used to acquire sensitive information. Some of these techniques are described below, including browser spoofing, use of proxy error and redirect messages to insert evil scripts, and website forging.

Browser Spoofing

Once the attacker successfully completes the SSL handshake, they may create a new browser window that overlays the original window with an embedded Java applet. The bogus window contains the status line, security lock, and contents of the original target page. Event response functions are handled in case the user clicks GUI components to convince the user that the page is authentic. For instance, if the user clicks the lock icon, the certificate window is displayed. This certificate window, however, can only be moved within the confines of the fake browser window. This is the only visible weakness that can alert the user that the browser window is fraudulent (Qi, Tang, & Wang, 2008).

The solution proposed by Qi et al. (2008) is to have the browser create and embed a random piece of art into the web browser at a random location. On a new HTTPS request, the browser takes a snapshot of the region with the art. Whenever there is action

in the browser, the snapshot is compared to the new browser window. Since the attacker does not know the image or the location, they are unable to fool the verification. This research is an improvement on previous papers suggesting random backgrounds or logos or boundaries which rely on user verification.

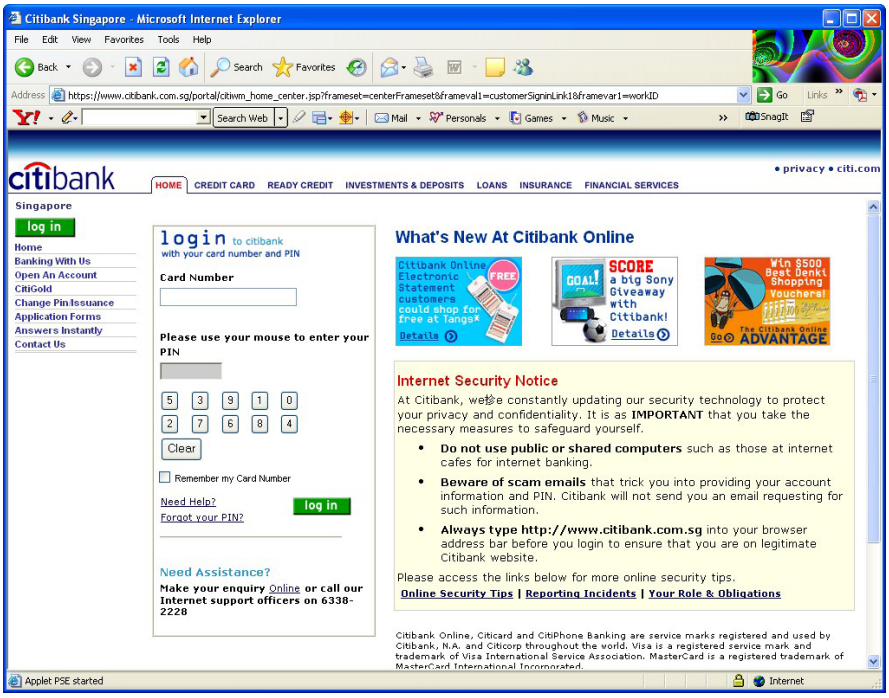

Figure 3: Random Art Indicator (Qi, Tang, & Wang, 2008)

Proxy Script-Based Exploits

Error Responses

When a browser sends a request for https://NonExistentServer.com, a proxy will legitimately return an HTTP 502 error to the browser. Browsers render this page in the context of https://NonExistentDomain.com even though the server does not exist. Since the browser relies on the proxy for the tunneling, the proxy can lie to the browser as in the following scenario. The browser requests https://myBank.com. The proxy returns 502 with an iframe (inline frame) and a script. When the browser renders the error

message, the iframe will cause the browser to send another request to https://myBank.com. This time, the proxy allows the request to go through to the server. The user's banking data will be loaded into the iframe. The embedded script is running in the context of https://myBank.com and is allowed access to the user data, which it can then send to a third party (Chen et al. 2009).

Redirect Responses

Proxies also have the ability to redirect requests. A user may make a request to https://a.com and receive a response 302 moved temporarily. The browser renders this page in the context of https://b.com. Thus, the redirected request to https:/b.com is the same as a direct request to https://b.com. An evil proxy can exploit this function in the following scenario. Suppose a page from https://myBank.com includes a third-party script such as https//x.thirdparty.net/foo.js. A script element does not have its own security context but runs in the context of the frame that imports it. The proxy can use a 302 message to redirect the script request to https://EvilServer.com and cause the script https://EvilServer.com/foo.js to be imported and run in the context of https://myBank.com. The evil script now has access to the user's banking data (Chen et al. 2009).

HTTP but HTTPS loadable pages

Websites that simultaneously open HTTP and HTTPS ports are vulnerable to the following script-based attack. Once the user has established a secure connection, if a non-secure HTTP page is requested from the same domain, the browser will unlock the security icon and (depending on the browser) warn the user with a pop-up dialog for

permission to render the insecure HTTP items. However, these checks and warnings can be bypassed when an evil proxy inserts an invisible iframe into an insecure HTTP page. The iframe loads an HTTP page with a script request in the HTTPS context. The proxy will provide a malicious script in place of the actual script. The checkout button on the insecure merchandise page is overwritten so that when the user clicks on it, the HTTPS checkout page opens in a separate tab, and the personal data is obtained by the attacker. This vulnerability is pervasive across the internet (Chen et al. 2009).

Cookies

Cookies are used by browsers to store the state of transactions. Cookies have a secure attribute that is intended to be set for HTTPS connections, but unfortunately often is not set. Cookies without the secure attribute can be stolen in the following scenario. The proxy waits for the user to establish a secure connection with the target website. After that, once the user requests any HTTP page, the proxy embeds an iframe for an HTTP page of the target website into the response. When the browser renders the iframe, it makes a request to the target website with the authentication cookie in plain text (Chen et al. 2009).

All of the proxy-based script attacks must be fixed by the browser vendors, with the exception of cookie attributes which must be fixed by the various websites that fail to set the secure attribute. Chen et. al. (2009) notified the vendors of the vulnerabilities before publishing their research, and it is reasonable to expect most of these holes to be fixed. Meanwhile, the authors suggest that users be cautious when using wireless access points or network proxies, avoiding any critical transactions when connected to these resources.

Summary of What is Known and Unknown

SSL is a widely implemented protocol for securing communication over the internet. It is an attractive target for attack, and is quite vulnerable to man-in-the-middle attacks.

ARP spoofing, DNS spoofing, fake hypertext links, and fake WPAD responses are techniques used to divert the communication path through an attacker's server. These technologies are inherently insecure and can be exploited for a MITM attack. A necessary second step to establish a MITM is to circumvent server authentication. A few of the techniques used to bypass the authentication system are to use fake certificates from a legitimate CA, trick the user into installing a self-signed certificate, and use scripts to access cached certificates. Once the authentication is bypassed, web spoofing techniques such as forging website pages can trick the user into divulging sensitive information. Fake webpages containing invisible iframes and scripts can be constructed to violate the same-domain security policy of browsers and gain access to the user's private data.

Feasible solutions have been proposed for most of these security holes. Due to the ubiquitous nature of the internet, it takes time to agree on standards and implement them without disrupting existing functionality.

Contribution This Study Makes to the Field

This study attempts to organize the many security issues by categorizing them as diverting the communication path, circumventing authentication, and web spoofing. It selects a sample of papers from each category to present an overview of the types of security problems involved with SSL.

Chapter 3

Conclusions and Recommendations

Conclusions

Diverting the communication path exploits properties of ARP, DNS, DHCP, and WPAD. These protocols were developed in the early days of the internet before security was a concern. There are ways to secure ARP, but they are labor intensive for large organizations. DNS security has been under development and is finally due to roll out in 2010. Authentication was later added to DHCP, but since many clients and servers do not fully support it, servers must support clients that do not support DHCP authentication. IPSec is often used to make DHCP servers secure. WPAD is only as secure as the DHCP and DNS servers with which it communicates.

Standards for these ubiquitous protocols will continue to evolve as the internet evolves. RFCs are constantly being added and critiqued. There are at least 114 DNS-related RFCs, some of which update or replace previous RFCs. In an environment as diverse and pervasive as the internet has become, implementing change is slow. Even when the standard has been accepted, as in DHCP authentication, backwards compatibility leaves gaping security holes.

Server authentication using digital certificates relies on bug-free and secure implementations of the browser, the CA, and the DNS server. If users could properly authenticate the servers with which they communicate, they would be protected from MITM attacks (Opplinger, 2006) even if the communication path was successfully diverted. But there are weaknesses in the authentication process that can be exploited. For example, browsers can be compromised with the malicious use of iframes and

scripts, CAs can be subverted into providing certificates with deceptive domain names, and users can be fooled into installing self-signed certificates.

Once a MITM has circumvented server authentication, the final security issue is visually fooling the user into providing critical data with web spoofing. Examples of web spoofing include generating a forged browser window that covers the actual browser, sending fake web pages to the existing browser to be rendered, or overwriting checkout or login buttons to redirect the browser to an evil server. By the time the attack has reached this point, it may be impossible for the browser or the user to detect the deception.

Implications

The fact that our most ubiquitous security protocol is vulnerable to MITM attack indicates a need for research to discover feasible ways to secure the underlying mechanisms that SSL relies upon, or modify the SSL protocol. Since a MITM attack requires diverting the communication path, bypassing authentication, and providing fake webpages, successfully securing any one of these three areas would provide a significant amount of protection.

Recommendations

Until SSL can be made more secure, there are measures that can minimize the risk of a MITM attack. First, it is desirable to prevent an attacker's ability to monitor the IP traffic to be hijacked. In an environment using ARP, it is advisable to use static ARP, ARP watchdog software, or disable the ability to run in promiscuous mode to prevent sniffing. Secure versions of DHCP should be used whenever feasible, or IPSec can be

used to protect DHCP. In wireless settings, users should always use encryption such as WPA with a strong password in private networks. Proxy settings should be disabled whenever sessions requiring a proxy have ended. Secure communications should not be conducted in public WIFI areas or in untrusted networks unless using Virtual Private Networks.

To help protect against server authentication fraud, users must pay attention to subtle URL changes, and be suspicious of unusual behavior such as browser warnings or requests to install certificates. Often, attackers exploit a user's lack of awareness of their role in the authentication process. The problem of using fake certificates with similar names to the legitimate certificate remains an unsolved problem. Clearly, more research is needed to make server authentication immune to circumvention.

The last step of the MITM attack is web spoofing. The best defense against web spoofing is to disable JavaScript, but this may also block legitimate desirable web content. Web browser vendors are quick to implement fixes to security holes that are exploited by evil scripts as they are identified. Users should be aware that it is important to update browsers to obtain these security fixes. Research to discover browser security flaws, such as that by Chen et al (2009) should be encouraged.

Summary

SSL is one of the most common protocols used to secure communications over the internet. However, it is vulnerable to MITM attacks. The use of ARP poisoning, DNS cache poisoning, fake Email links, and fake WPAD responses are among the techniques that can divert the communication path through an attacker's server. Server authentication can be bypassed with the clever use of similarly named certificates, self-

signed certificates, or cached certificates. Users can be visually fooled by sophisticated browser spoofing, or by carefully forged website pages. Scripts can be constructed that can violate the same-origin policy and allow access to sensitive user data.

Solutions have been proposed to address many of the security holes in SSL. However, SSL relies on many different protocols spread across a vast number of components across the internet. Solutions undergo a lengthy analysis process before being accepted as an accepted standard. Standards are slow to be implemented without strong incentives. In the meantime, users can protect themselves by being aware of the risks and minimizing their exposure.

As illustrated, SSL security issues arise from independent distributed components, each with widely varying degrees of security. For a successful attack, holes must be found to divert the communication path, circumvent the authentication, and trick the browser into rendering forged web pages. As with most areas of security, SSL security is an evolving process, and will be an area of active research for many years to come.

Reference List

Bisel, L. (2007). The Role of SSL in Cybersecurity. *IT Professional*, v9, no. 2, 22-25.

Chen, S., Mao, Z., Wang, Y., and Zhang, M. (2009). Pretty-Bad-Proxy: An Overlooked Adversary in Browsers' HTTPS Deployments. *In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*., 347-359

Chomsiri, T. (2007). HTTPS Hacking Protection. *In Proceedings of the 21st international Conference on Advanced information Networking and Applications Workshops*. v1. 590-594

Fetzer, C., Pfeifer, G., and Jim, T. (2005). Enhancing DNS Security using the SSL Trust Infrastructure. In *Proceedings of the 10th IEEE international Workshop on Object-Oriented Real-Time Dependable Systems,* 21-27

Huawei, Z., and Ruixia, L. (2009). A Scheme to Improve Security of SSL, *Pacific-Asia Conference on Circuits, Communications and Systems,* 401-404.

Oppliger, R., Hauser, R., Basin, D (2006) SSL/TLS session-aware user authentication - Or how to effectively thwart the man-in-the-middle, *Computer Communications,* v29, Issue 12, 2238-2246.

Qi, F., Tang, Z., Wang, G. (2008) Attacks vs. Countermeasures of SSL Protected Trust Model. *The 9th International Conference for Young Computer Scientists*, 1986-1991. 2008

Saito, T., Hatsugai, R., and Kito, T. (2006). On Compromising Password-Based Authentication over HTTPS. In *Proceedings of the 20th international Conference on Advanced information Networking and Applications, v1*, 869-874.

Stallings, W. (2005). *Cryptography and Network Security (4th Edition)*. Prentice-Hall, Inc.