The Problem with Multiple Roots in Web Browsers - Certificate Masquerading

Capt James M. Hayes
The Office of INFOSEC Research and Technology
Suite 6534 - 9800 Savage Road
Fort George G. Meade, Maryland 20755-6534
jmh@tycho.ncsc.mil

Abstract

Much work is going into securing the public key infrastructure (PKI). Various models for trust exist; Pretty Good Privacy (PGP) and the Progressive-Constraint Trust model are examples.[3] These models describe how to protect and ensure the interrelationships of their certificate based structures; however, vulnerabilities may arise when structures based on certificate authorities (CAs) are involved. The vulnerability is based upon multiple root certificate authorities. This paper examines the need for improved methods for verifying the binding of a certificate authority (root) to the source of a protocol's messages. The protection mechanisms developed for protecting and ensuring this binding within a CA hierarchy can break down in environments where multiple roots exist. This can lead to the possibility of a CA undermining the trust placed in a peer CA.

1. Introduction

In the book *Secure Electronic Commerce*, the authors state "... the certificate user may hold multiple root public keys and may make decisions that one root key is trusted for some purpose and another root key is trusted for another purpose. (In saying the root key is trusted, we actually mean that all certification paths starting from that root key are trusted.)"[3] If a root makes a decision to imitate a peer root, severe consequences can result.

The following examination considers the World Wide Web (WWW) and uniform resource locators (URLs) used in the Secure Socket Layer (SSL) Protocol and the Secure Hypertext Transfer Protocol (S-HTTP). Various scenarios are presented that demonstrate the subtleness of the problem of trust and certificates. These scenarios are based on an analysis of the design of current Web browser features, use of certificate servers in the PKI, and an expansion of the discussion on "Handling Multiple Certification Authorities" in [7].

This author describes how trust can be manipulated to take advantage of this potential security problem with Web browsers using S-HTTP 1.3, SSL 3.0, or potentially any WWW-based secure protocol that operates in an environment susceptible to outside tampering. The key to the vulnerability is the unprotected and unverified exchange of public keys in certificates. It is envisioned that the exploitation of this problem would in some instances go unnoticed by users of Web browsers.

Certificates can be used to securely process transactions across the WWW. A server's certificate will contain information such as the name of the holder, its URL, a public key, and other identification information. In addition, a good certificate will be certified by a CA indicating that the holder of the certificate is indeed authentic. A *valid* certificate is defined as a certificate that has been signed by a CA that a browser is willing to accept without challenge. If a client computer is presented with a *valid* certificate, it can encrypt a message with the public key of the certificate and send the message securely to the holder of the certificate. The holder of the certificate will decrypt the message with her private key. The holder keeps the private key in a secure place so that no one else will be able to retrieve information from the messages that are encrypted with her public key.

Certificate masquerading allows a masquerader to substitute an unsuspecting server's *valid* certificate with the masquerader's *valid* certificate. The masquerader could monitor Web traffic, picking up unsuspecting victims' surfing habits, such as the various net shopping malls and stores a victim may visit. The masquerader could change messages at will without detection, or collect the necessary information and go shopping on his or her own time. In each instance, the masquerader

need not worry about problems addressed in Web Spoofing: An Internet Con Game [2], i.e., rewriting URLs, modifying the status line, or worrying about the user looking at the source document. All URLs would be unchanged. Certificate masquerading could also involve replacing the client's certificate with the masquerader's client certificate; however, this paper does not address client certificate masquerading.

To demonstrate one situation where certificate masquerading can take place, and how the purpose of a legitimate root could be abused, the author will enlist the assistance of a couple of the usual characters needed for demonstrating protocols: Bob, a good guy, and Mallory, a malicious active attacker. In this illustration, Bob and Mallory will be playing the roles of employees who work at the Bureaucratic Institution for Mismanagement (BIMM) Corporation.

2. A Case Study of the BIMM Corporation

Bob often performs on-line trading of stocks. His on-line trading company uses SSL with server authentication. The server's certificate is signed by Ultra Trust Security Services, Inc. The Ultra Trust root is distributed within Bob's Web browsers at home and at work. The on-line trading server does not require that Bob use a client certificate. This allows him to perform transactions at home and at work.

Mallory is a poorly paid system administrator at the BIMM Corporation. He is responsible for network security duties, which include monitoring of http traffic, setting up firewalls, and something new to the organization called certificate authority management. During the course of his http monitoring duties, Mallory notices that Bob visits www.host.com. Mallory realizes that this is Bob's on-line trading site. Mallory is jealous of Bob because Bob is a manager and pulls down a six figure income. Mallory decides to steal money from Bob's on-line account.

It is a Monday afternoon and the stock market seems to be taking a dive. Bob connects to www.host.com to sell some of his shares. He notices that the security icon of his browser is on, and no alerts have sounded. Bob believes that everything is fine and signs on, using his password, and makes the sale.

Two days later, Bob is back at work and the market is on the rise. He signs on to www.host.com to buy some shares, and finds out that he has no funds left in his cash account. Bob does not know it, but Mallory stole all of his money. The following sections will show how Mallory may have performed this feat.

3. Certificate Masquerading Basics

The masquerade begins with Mallory sitting between Bob and the on-line trading server. This is the traditional "man-in-the-middle" scenario. The first task for Mallory is to set up an environment where he can monitor network traffic and change the initial, unprotected handshake messages of the protocol at will (S-HTTP, SSL). Next, Mallory creates a *valid* certificate and keying material for the www.host.com server. Once these two tasks are complete, Mallory is ready to start the masquerade.

Figure 1 [6] shows how the basic masquerade is put into play after Bob initiates a request for the www.host.com server's certificate information. Typically, this is done by an HTTPS or S-HTTP request in which the server responds with certificate information.

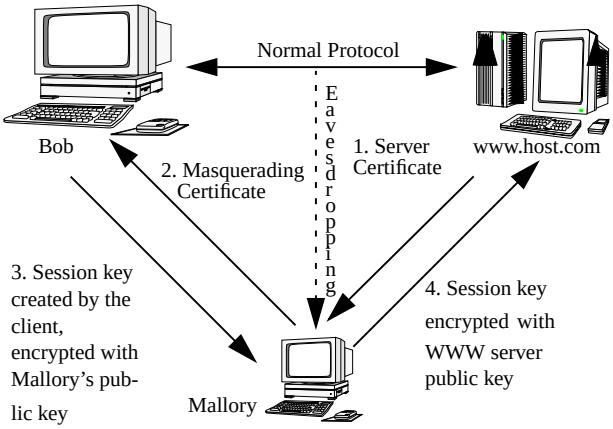

Figure 1 - Masquerade Initialization

Step 1 Mallory intercepts the server's certificate;
Step 2 Mallory then substitutes his *valid* certificate for that of the server's *valid* certificate (Note: Mallory's certificate does not come from the Ultra Trust root CA, it is linked to the BIMM Corporation root CA, both of which are in Bob's Web browser at work);
Step 3 Bob's Web browser generates a session key, encrypts it with the public key found in Mallory's certificate, and sends it back to the server;
Step 4 Mallory intercepts Bob's message, decrypts the (S-HTTP or SSL) encapsulated message to obtain the session key, re-encrypts the message with the server's public key signed by the Ultra Trust CA, and sends it on to the server.

This would be the first cycle of many messages in which Mallory could simply monitor or tamper with S-HTTP or SSL message traffic. Protocols that encrypt session keys using the server certificate's public key

may be susceptible to this masquerade. The goal is to determine why the masquerade is possible and how to prevent it. In order to do this, one must have an understanding of Web browsers' defense mechanisms.

4. WWW Browser Defense and Limitations

It should be pointed out that a Web browser-based defense is not the only possible line of defense. Directory services can also serve a purpose in an enterprise setting; but on the WWW, Web browser defense is the only practical means presently available.

A typical Web browser has two basic forms of defense for validating and verifying server certificates.

> Server authentication in the Web browser validates both the Internet identity of the server and the integrity of the public key certificate. For example, when Bob's browser accesses a secure URL pointing to host.com, the SSL protocol delivers the public key certificate from host.com to Bob's browser. First, the browser verifies the digital signature on the certificate, essentially verifying that its crypto checksum is intact. Then the browser compares the host name in the certificate against the host name in the URL. If the two match, then Bob is definitely establishing an SSL session with host.com. If the two do not match, then Bob may be connected to a bogus server masquerading as host.com.[7]

As the previous quote illustrates, the assumption in the past was that a man-in-the-middle had to control the server. Further discussion will show why Bob's SSL connection to host.com may be definite, yet still compromised by Mallory.

The shortcoming in current Web browser defense is that Bob must examine various items before he can make a decision as to whether or not to connect to a site. Keep in mind that in the example, Bob did not get a security alert and therefore, he did not bother to examine the server's certificate. Even if he did, he may not have come to the correct conclusion. The following analysis will show some situations where it would have been difficult for Bob to determine if his connection was actually secure.

5. Server Certificate Masquerading

The following section will examine three forms of server certificate-based masquerading which could have been executed in the BIMM example. In each case, the Web browser would not have given a warning. In the second and third cases, the look-alike Web server certificates would display with the proper CA names, but they would not be digitally equivalent to the authentic certificate.

5.1 Certificate Masquerading with the Subject Name - No Browser Warning

This form of masquerade is based on the lack of granularity checking in present day browsers. Mallory controls the BIMM CA and knows that Bob's browser is willing to accept certificates from both the Ultra Trust and BIMM CAs. Mallory has the ability to create certificates that chain up to the BIMM root CA, found in Bob's Web browser at work.

The masquerade takes place as follows. Mallory creates a *valid* certificate that looks like an ordinary certificate that is signed by the BIMM Corporation CA. However, the common name (CN) points to the targeted WWW server, www.host.com. There are no warnings for unrecognized CA and URL binding given by the browser. Table 1 illustrates this property. If Bob had looked at the certificate, would he have known which CA should have signed the www.host.com certificate?

Table 1

Subject:	Subject:
US	US
New York	New York
New York	New York
The Barter Group	The Barter Group
On-Line Trade Div	On-Line Trade Div
www.host.com	www.host.com
Issuer:	Issuer:
US	US
Ultra Trust Security Services, Inc.	BIMM Corporation
Certification Authority	Certification Authority

5.2 Certificate Masquerading Using a Subordinate Certificate Authority or Cross Certification - No Browser Warning

A subordinate CA (SCA) is a CA that is created by a higher level CA. The higher level CA can be a root or a CA in the path of the root that has the authority to sign CA certificates. Cross-certification is a process in which two CAs securely exchange keying information

so that each can certify the trustworthiness of the other's keys. Essentially, cross-certification is simply an extended form of third-party trust in which network users in one CA domain implicitly trust users in all other CA domains which are cross-certified with their own CA. [1] In the case of the BIMM Corporation, Mallory could have created a fraudulent Ultra Trust look-alike SCA or a fraudulent Root CA and cross-certified with it.

The following three certificates shown in Table 2 demonstrate this property using a fraudulent SCA: Certificate 1 is the BIMM root; Certificate 2 is a fraudulent SCA; and certificate 3 is Mallory's www.host.com masquerading certificate. The crypto handshake will show that the sender's (Mallory's) certificate will chain correctly to the BIMM root in Bob's Web browser and that the sender has the private key associated with the public key in the certificate. In addition, the CN (www.host.com) will resolve correctly with the URL.

Table 2

1. BIMM Root	2. Fraudulent Ultra Trust "Root"	3. Masquerading Certificate
Subject:	Subject:	Subject:
US	US	US
BIMM Corporation	Ultra Trust Security Services, Inc.	New York
Certification Authority	Certification Authority	New York
		The Barter Group
		On-Line Trade Division
		www.host.com
Issuer:	Issuer:	Issuer:
US	US	US
BIMM Corporation	BIMM Corporation	Ultra Trust Security Services, Inc.
Certification Authority	Certification Authority	Certification Authority

Since Mallory could have created a fraudulent SCA or cross-certified CA, Bob would have had a difficult time determining whether or not he had the "authentic" certificate. If Bob did check, he would have seen the fraudulent certificate signed by the fraudulent "Ultra Trust" created by Mallory.

6. Name Constraints - No Defense in an Environment of Multiple Roots

The following description of the X.509 name constraint model is given by [3]:

> The X.509 name constraint model allows any certification authority to specify, when it certifies another certification authority, exactly what names are allowed in subsequent certificates in the certification path.

Although any given root CA may manage its name space through name constraints, it cannot apply these constraints across multiple root CAs. Name constraints across multiple roots must be managed by the application or an external service to the application.

7. Defending Against Certificate Masquerading

The following is an analysis of ways to prevent these scenarios. In terms of certificate masquerading, the following question must be answered: Does the certificate presented belong to the organization in question for the validity period given, and is it representative of the purpose for which it is intended to be used? There are several aspects to that question, only some of which can be answered in this paper.

Solutions will be presented to show what Bob could have done with his present Web browser and Web technology. Solutions will then be suggested for possible improvements for the future.

7.1 Present Web Technology Defenses

There are two possibilities of present day defense that may have helped Bob: certificate fingerprints and SSL 3.0 client authentication.

7.1.1 Certificate Fingerprints

Certificate fingerprints, like human fingerprints, are unique. Generally speaking, they consist of an MD5 or SHA-1 hash of the certificate. If Bob had prior knowledge of the www.host.com certificate fingerprint, he could have checked the certificate's fingerprint and determined that Mallory had slipped a masquerading certificate into Bob's protocol handshake with the server.

This solution requires diligence on Bob's part. He would have to check the server's certificate each time he

connects. Bob would also be required to have prior knowledge of the server's fingerprint. This solution may work in this situation, but would require Bob to securely acquire the fingerprint of every Web server with which he attempted to perform secure communications.

7.1.2 SSL 3.0 Client Authentication

SSL 3.0 supports client authentication; however, what is unique about SSL 3.0 is that it supports a CertificateVerify message that will allow the server to detect any manipulation of the SSL 3.0 Handshake Protocol. The assumption is that Mallory cannot create a client certificate that the server is willing to accept as Bob.

If Bob has a client certificate that has a signing capability and the www.host.com server trusts Bob's certificate, then Bob's browser can perform the SSL 3.0 CertificateVerify message. The sequence of events for detecting Mallory's attack would be as follows: Bob's browser performs the CertificateVerify which consists of a hash of all handshake messages starting at client hello up to but not including the CertificateVerify message and then creates a signature over that hash. [4] It is not important to know what all of the messages are that make up the Handshake protocol; however, it is important to know that the Handshake messages include the server certificate. The browser sends the CertificateVerify to the server. Since it is signed by Bob, Mallory cannot change the message. Mallory cannot create a fake CertificateVerify because he does not have a client certificate that the server is willing to accept. When the server receives Bob's CertificateVerify, the server will detect that the server hello messages (server's messages in the Handshake protocol) were manipulated. This is because Bob's browser will calculate the CertificateVerify message using Mallory's certificate and not the server's. Mallory will be caught.

This solution works fairly well in applications where the server has prior knowledge of the client's certificate. "In practice, SSL client authentication is primarily used for special applications and not for general-purpose authentication of Internet users." [7] In Bob's situation, bringing a certificate in from home is out of the question. He fears that loading a certificate on his computer at work would leave him open to the possibility of someone using his private key if he were away from his computer. A smart card may solve this problem.

7.2 Possible Browser Improvements

It is apparent that the browser should allow the displaying of the certificate chain. In addition, features should be added that allow the browser to enforce its own form of policies. Both of these issues will be discussed in the remaining sections.

7.2.1 Displaying of Certificate Chain

A certificate chain display should be available so that the user can view a certificate chain from the end entity certificate through the root certificate. This capability would allow the user to quickly notice the masquerades described in Sections 5.2. In both cases, Bob might see that the Ultra Trust issuer of the end entity certificate is linked to the BIMM Corporation's CA.

This solution, however, does not help the novice to determine whether or not there is a masquerade during the session. It also requires the user to diligently check the certificates each and every time.

7.2.2 CA Regions and Granularity Checking

Generally speaking, individuals like Bob perform surfing that can be broken down into regions, based on where on the Internet a user is surfing. The regions could be Internet, intranet, and trusted sites. Intuitively, the BIMM Corporation's CA should only be used for intranet regions and the Ultra Trust CA should be used for Internet regions. This knowledge can be used in an attempt to devise a solution that ties the CAs (root keys) to their purpose.

An assumption must be made that the CAs used in the Internet region will not perform any masquerading amongst themselves, and that they will not issue certificates to incorrect entities. For example, Ultra Trust is the actual CA for www.host.com; no other CA trusted as an Internet CA should issue a www.host.com certificate to any other entity other than the on-line trading company that owns www.host.com. If an Internet-class CA was responsible for issuing a masquerading certificate and got caught, it would undermine its ability to claim to be trustworthy.

The following solution is proposed: Browsers should allow users to assign CAs to regions based on their purpose. There would be three regions, based on Internet, intranet, and trusted sites.

The Internet region would be based upon any CAs not assigned to an intranet or trusted site region. If a server's certificate is not signed by a CA in the intranet or trusted site region, then the server certificate would

have to be able to be certified by a CA in the Internet region. In Bob's case, the Internet region would consist of Ultra Trust CA and any other CAs that are distributed with his Web browser. Addition of CAs to this region should be strictly controlled. For the intranet region, Bob would assign the BIMM Corporation's CA with a granularity check requiring the URL in the CN to conform to the constraint of *.BIMM.com. If a finer grain of control is called for within the intranet region, then granularity checking similar to Table 3 may be instituted. Users could be given the ability to require that a certificate contain certain values in attributes before allowing the use of the public key of the certificate to be used for delivery of a session or transaction key. For example, if a user has a habit of connecting to *.BIMM.com domains, and all *.BIMM.com Web servers are certified by the BIMM CA, then the user could require that the certificate adhere to the granularity check presented in Table 3.

Table 3

Subject:	Issuer:
C=US	C=US
S=*	S=
L=*	L=
O=BIMM Corporation	O=BIMM Corporation
OU=*	OU=
CN=*.BIMM.com	

This ability limits the scope of masquerades. For example, the browser may be willing to accept various certificates for various CAs, but when connecting to *.BIMM.com, the masquerader would have to be in the certificate chain of the root CA. In order to reduce this possibility, the example in Table 3 shows that for all US based servers, the certificate must be signed by a US based BIMM CA. If the BIMM Corporation PKI did not support name constraints, and a foreign BIMM CA attempted to masquerade as a US based server, the browser would catch it. For trusted sites, a tighter binding must be enforced.

A trusted site can best be described as a site that is not authenticated by an intranet CA or an Internet CA. It is a site that operates its own CA that a user must connect to. The specific URL must be bound to the CA in question. The difficulty in safely assigning a CA of this type to the trusted site region is in creating the initial secure binding of the CA information to the URL in the client.

8. Conclusions

Any protocol and application that uses multiple roots and the public key in a receiver's certificate for passing a session key may be a candidate for certificate masquerading. The underlying problem in the masquerade, if not apparent by now, is Gerald Holzmann's second element of a protocol specification which states, "Each specification should include explicitly the assumptions about the environment in which the protocol is executed" [5]. Even if one believes that one has air-tight security, the fundamental question comes down to "Does one trust the host one is connected to and the path to that host?"

Acknowledgments

Andrea Colegrove, Bill Kutz, Pete Sell, Doug Maughan, Eric Harder, Mike Oehler, and Sonya Hunt provided significant input to and review of this document.

References

[1] Curry, I., *The Concept of Trust in Network Security*, Entrust Technologies White Paper, http://www.entrust.com/downloads/trust.pdf, 1995

[2] Felten, E. W., Balfanz, D., Dean, D., Wallach, D. S., Web Spoofing: An Internet Con Game, In *Proceedings of the 20th National Information Systems Security Conference, October 1997* pp. 95-103.

[3] Ford, W., Baum, M. S., *Secure Electronic Commerce*, Prentice Hall PTR, Upper Saddle River, N.J., 1997.

[4] Freier, A. O., Karlton, P., Kocher, P. C., *The SSL Protocol Version 3.0* http://search.netscape.com/eng/ssl3/draft302.txt, November 1996 (working draft).

[5] Holzmann, G. J., *Design and Validation of Computer Protocols*, Prentice Hall, Englewood Cliffs, NJ,1991.

[6] Josang, A., Security Protocol Verification using SPIN. In *Proceedings of SPIN95 the First SPIN Workshop*, http://netlib.bell-labs.com/netlib/spin/ws95/papers.html, October 95.

[7] Smith, R. E., *Internet Cryptography*, Addison-Wesley, Reading, Massachusetts, July 1997.