Bridge-Enabling Web Servers

Prepared By:
e-Authentication Architecture Working Group
PKI Tiger Team

Draft Version 1.0

March 30, 2004

TABLE OF CONTENTS

LIST OF FIGURES

SECTION 1: INTRODUCTION

This document discusses technical steps necessary to enable a web server to accept PKI-based user credentials and validate them through a certificate bridge (e.g., the FBCA). In so doing, the document:

- Differentiates between “hint lists” and “trust lists”
- Discusses packaging and distributing the certificate authority (CA) certificates needed to generate “hint lists”
- Provides test scripts to help automate installation of “hint lists”
- Discusses web server add-in filter modules, generically called Policy Enforcement Points (PEPs), that call remote validation services to ensure client certificates are validated with respect to acceptable trust anchors

Additionally, appendixes cover:

- Standards governing Secure Sockets Later/Transport Layer Security (SSL/TLS) client certificate authentication
- Test results of “hint list” experiments
- A survey of web server brands deployed at 91 different federal government web sites
- A survey of web browser brands used to access one web site

1.1 BACKGROUND

Web servers requiring client certificates for user authentication must provide to the user's browser (during the SSL/TLS tunnel setup phase) a list of distinguished names (DNs) of all CAs from which the web server is willing to accept end-user certificates. We call this list of acceptable issuing CAs the “hint list.”[1] Next, the user's browser presents to the user a pick list of all client certificates that have been issued by:

- Any CA on the “hint list”
- Any CA subordinate to a CA on the “hint list,” providing the user’s browser can assemble a certificate chain from the user’s certificate to a CA to the “hint list”

The user then chooses one of these client certificates to be used in the particular transaction at hand. All of this must occur before the SSL/TLS tunnel setup between the browser and web server is complete. The impact is that the web server must know *before the transaction is attempted* all possible issuing CAs. This web server configuration requirement also implies that certificate bridges (e.g., the FBCA) currently cannot be as dynamic as intended in web server environments, because if a new CA cross-certifies with a certificate bridge then each web server wishing to validate a client certificate through that bridge must be configured with knowledge of the new CA, i.e., acceptance of the new CA is not automatic. This is a technical limitation of SSL/TLS and affects web servers but not email systems. (See Appendix A for a technical discussion of the SSL/TLS tunnel setup process during which the “hint list” is transmitted.)

[1] The web server is providing is providing to the user a list of hints of which end-user certificates the web server might accept. Other factors such as certificate validity, naming constraints, and policy constraints will determine the ultimate acceptability of the end-user certificate the user chooses.

It is important to distinguish between a "hint list" and a "trust list":

- A "hint list" is a list of CA names (i.e., the issuer's DN) that is sent to the user's web browser in the `CertificateRequest` message of an SSL/TLS session during session establishment (see Appendix A) when PKI-based client authentication is required. This "hint list" is used only to assist the user in selecting an appropriate credential.
- A "trust list" is a set of self-signed certificates used by web servers that perform their own path validation. This set of self-signed certificates could also be used by validation services as trust anchors during path discovery and validation.

The "hint list" gives the user an indication of which credentials *could* be acceptable, but final acceptance is subject to: trust path validation through a trust anchor on the "trust list"; and certificate path validation.

Apache web servers can be programmed to distinguish between "hint lists" and "trust lists." Unfortunately, Microsoft Internet Information Services (IIS) web servers cannot. To produce the correct "hint list," the IIS web server requires the self-signed CA certificates be loaded into the "Trusted Root Certification Authorities" certificate store, which is also (as the name implies) the "trust list" certificate store.

We are not aware of any web server that can natively perform path discovery and path validation, therefore an external validation service is needed.

1.2 GOALS

This document focuses on two goals:

- Provide instructions and helper applications to each web server for periodically downloading and locally configuring the "hint list."
- Provide guidance for using web server filter modules to intercept client certificates and perform path discovery and validation as a condition of web content access, thereby separating the functions of "hint list" and "trust list." We assume path discovery and validation will be performed by an external validation services.

1.3 APPROACHES TO "HINT LISTS"

As discussed in Appendix B, there exist two immediately viable approaches:

- Option #1: Configure each web server with only the self-signed root certificate of the bridge CA, and configure each user's browser with their client certificate and all intermediate CA certificates in the issuing chain (but *not* self-signed root certificates), and the appropriate cross-certificate leading to the bridge CA.
- Option #2: Configure each web server with the self-signed root certificates of all possible CA chains responsible for issuing client certificates, and configure each user's browser with the complete certificate chain from their client cert to the self-signed root CA.

The first option has the advantages of (a) minimal burden on web server administrators, and (b) the web browsers construct and send the entire certificate path (including cross-certificates) to the web server. However, this option has the disadvantages of (c) placing additional burden on the user to install the cross-certificate, and (d) Internet Explorer web browsers will not construct an appropriate certificate path including the cross-certificate if the issuing self-signed root CA is included in the browser.

The second option overcomes the last disadvantage, and it requires the user to load only the standard certificate chain typically provided by their issuer. It, however, does require each participating web server be configured with all self-signed root certificates. Therefore, to reduce the burden on the user, and to permit continued use of IE browsers, we choose this second option. Implementation techniques are discussed in Section 2.

1.4 APPROACHES TO CALLING EXTERNAL VALIDATION SERVICES

Neither the Apache 1.3 family web servers nor Microsoft IIS 5.0 web servers can natively perform path discovery and path validation that include cross-certificates and checking of policy and name constraints. Furthermore, constructing the “hint list” on IIS web servers requires populating the trust root certificate store, and presumably IIS logic behavior would be such that locally defined trust would preclude the need for validation through a bridge. Also, the “hint list” is needed merely to complete the transport-layer SSL/TLS encryption tunnel does not guarantee successful PKI-based client authentication at the application layer.

To help separate “hint list” requirements from path validation needs, and to work with currently available web servers, we advocate using a Policy Enforcement Point (PEP)—implemented as an Apache module or ISAPI filter—to intercept the client certificate and ensure path discovery and validation through the bridge are performed. Implementation considers are addressed in Section 3.

SECTION 2: OBTAINING AND CONFIGURING WEB SERVER "HINT LISTS"

To update a web server's "hint list" in an automated way, the following topics must be addressed:

- Collecting (to a central location) the public certificates of all possible self-signed root CAs that can be validated through the bridge
- Packaging that collection of CA certificates as a "hint list"
- Distributing the "hint list" package
- Unpacking the "hint list" and configuring the web server to accept end-user certificates from any CA on this "hint list"

2.1 COLLECTING ALL POSSIBLE CA CERTIFICATES

The first task is collecting the root public certificates of all issuing CAs that can be validated through the bridge and that issue (directly or indirectly via a subordinate CA) end-entity certificates. It is only these self-signed root CA certificates at the top of the issuing chains that are needed in the web server. (The reason behind this, as presented in the SSL/TLS specifications, is discussed in Appendix A. The results of corresponding lab experiments are described in Appendix B.)

It is assumed that all such self-signed root CA certificates will be collected periodically to a central location for subsequent distribution. These certificates could be provided at the time of cross certification, with the requirement that if the CA cross-certifies with a second bridge, then that CA must provide all self-signed root reachable through the second bridge. Alternatively, Mitretek has been developing a "spider" program that traverses a certificate bridge mesh and returns the certificates of all issuing CAs, and this software should be available for this project.

Typically, no path pruning (e.g., based on policy constraints or name constraints) is performed during the certificate gathering process. Thus, one collection of self-signed CA certificates will be used by web servers requiring either level 3 or level 4 PKI-based credentials.

2.2 PACKAGING & DISTRIBUTING

The entire collection of root CAs for inclusion can be made available in one archive file (e.g., a tar file and/or zip file) for downloading from a web server. To ensure the authenticity of the distributor and the freshness of the list (i.e., to avoid replay attacks), it is recommended that retrievals be made via HTTPS. If all root certificates are public information, then client authentication should not be required.

Apache web servers require PEM-formatted certificates. While IIS web servers appear to require DER-formatted certificates, they will also accept PEM-formatted certificates. However, to make distribution as easy as possible, it is recommended that a collection of PEM-formatted certificates be distributed in a tarball for Apache platforms, and a collection of DER-formatted certificates be distributed in a zip file for IIS platforms.

2.3 UNPACKING AND CONFIGURING

Different configuration methods are required for different brands of web servers. We will focus initially on only Apache and Microsoft IIS web servers, with support for additional web servers being researched and added as needed.

2.3.1 Apache Web Servers

To configure an Apache web server to produce the "hint list:"

- Download the "hint list" of certificates from the authorized web site
- Untar these certificates into the directory specified by the `SSLCACertificatePath` directive in the `httpd.conf` (or `ssl.conf`) file
- Create the necessary symbolic links by executing the shell script `mklinks` whose source code appears in Appendix C.1
- Stop and restart Apache via the commands: `apachectl stop` and `apachectl startssl`, respectively

2.3.2 Microsoft IIS Web Servers

To configure an IIS web server to produce the "hint list:"

- Download the "hint list" of certificates from the authorized web site
- Unzip these certificates into a temporary directory
- Using the perl script provided in Appendix C.2 to load the certificates into the "Trusted Root Certification Authorities\Local Computer" certificate store

(It is not necessary to restart the IIS web server for the changes to take effect.)

SECTION 3: POLICY ENFORCEMENT POINTS

The “hint list” provides a list of CAs from which user certificates may be accepted by the web server, and the “hint list” is necessary to complete the SSL/TLS tunnel setup. At this time, the underlying SSL/TLS engines are not enabled to provide certificate path discovery and validation. We therefore recommend a separate process to ensure path discovery and validation, thus separating the “hint list” usage from the “trust list” usage. The approach is generically called a Policy Enforcement Point (PEP), and it sits logically between transport-layer and the application-layer.

By the time the SSL/TLS tunnel is established, the web server has already sent its certificate, sent the “hint list,” and received the client certificate. Additionally, the server and client have provided and checked proof of private keys, and the entire chain of certificate signatures have been verified. If a certificate contains a Certificate Revocation List (CRL) Distribution Point (CDP), the CRL has also been retrieved and checked (at least in IIS 5.0).

Next, in order of execution, a PEP will intercept the client certificate (optionally including the entire certificate chain) and send the certificate to the Policy Decision Point (PDP) module for validation. The PEP will then enforce the certificate acceptability decision of the PDP; if the PDP indicates the certificate is valid, the PEP will allow access to the web contents, otherwise the PEP denies access. We envision the PDP to be a remote certificate validation service that performs path discovery and path validation with respect to trust anchors, name constraints and policy constraints. The PEP is typically implemented as an Apache module, or as a Microsoft ISAPI filter, and the interfaces for such components are well document and understood, and are the means by which web server vendors intend others to extend functionality. By the time the PEP completes its execution, the “trust list’ has been consulted, all certificate policies have been checked, and the CRLs have been consulted a second time.

Current commercial vendors of Security Assertion Markup Language (SAML) products have implemented PEP and PDP modules, although as of October 2003 the PDP functions were lacking in path discovery and validation features. Therefore, the PDP must be chosen carefully to meet bridge-validation requirements. PEPs are also compiled for specific web server brand / operating system pairings, and are available for Apache, IIS, Domino web servers, Netscape web servers, and WebSphere. Currently, the communication protocol between the PEP and PDP are typically proprietary because of extra information (in addition to certificate validation status) that is conveyed. However, a PEP could be easily developed for Apache and IIS web servers that could communicate via the Simple Certificate Validation Protocol (SCVP) if such a PDP should become available.

APPENDIX A: SSL/TLS CLIENT CERTIFICATE AUTHENTICATION SPECIFICATIONS

Figure A-1 shows the sequence of events during the SSL/TLS encryption set-up when client certificate authentication is required. Those steps marked with an asterisk (*) are additional steps due to client authentication and do not occur when only the server has a digital certificate. (The figure was constructed by consulting similar timing diagrams in [Rescorla01 pp64, 98].)

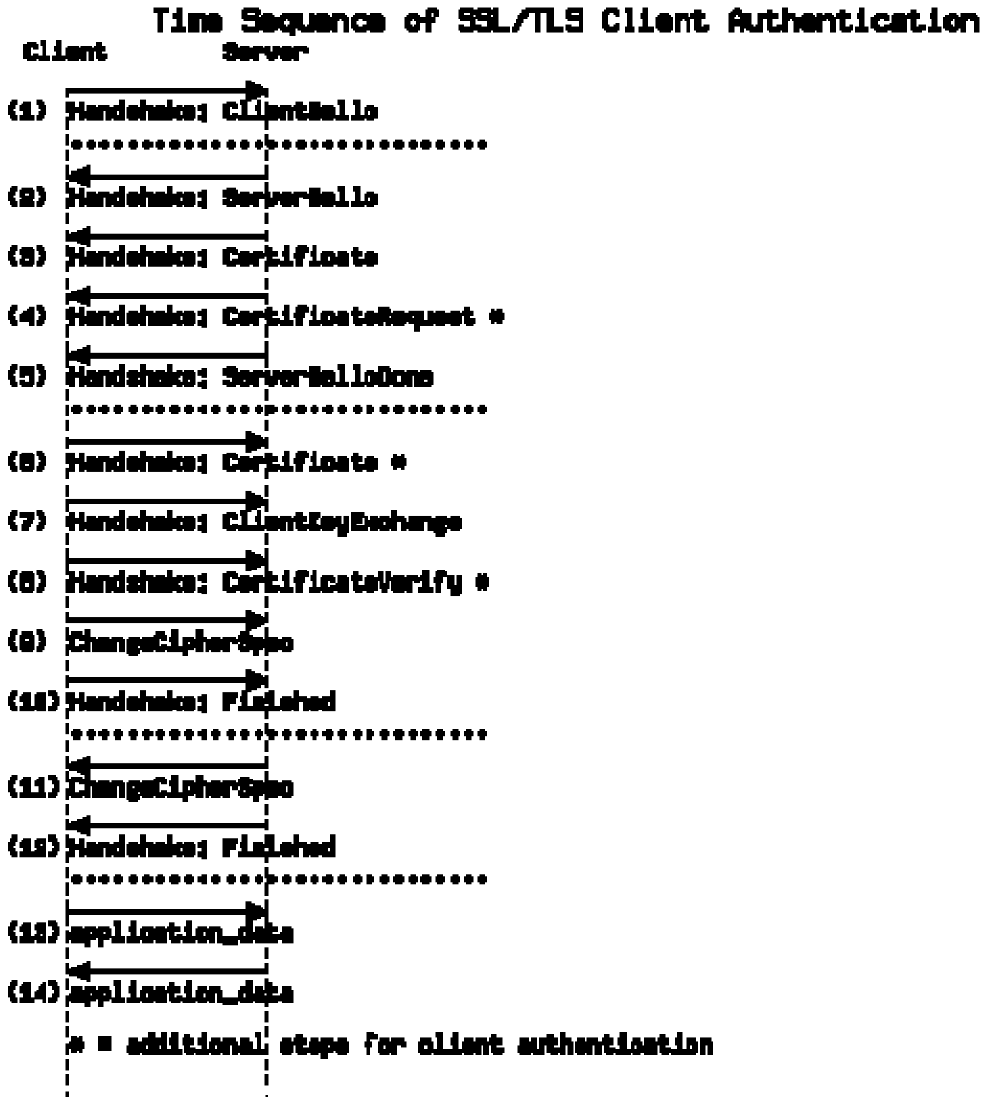
Time Sequence of SSL/TLS Client Authentication
Client
Server
(1) Handshake: ClientHello
(2) Handshake: ServerHello
(3) Handshake: Certificate
(4) Handshake: CertificateRequest *
(5) Handshake: ServerHelloDone
(6) Handshake: Certificate *
(7) Handshake: ClientKeyExchange
(8) Handshake: CertificateVerify *
(9) ChangeCipherSpec
(10) Handshake: Finished
(11) ChangeCipherSpec
(12) Handshake: Finished
(13) application_data
(14) application_data
* = additional steps for client authentication

Figure A-1: Time Sequence of SSL/TLS Client Authentication

The server must request the client's certificate (in step 4); there is no way for the client to offer its certificate unless requested via a `CertificateRequest` message. The "hint list" of CAs is delivered in `CertificateRequest` message.

The client conveys its certificate to the server in step 6 using the same `Certificate` message format as when the server conveys its certificate to the client in step 3. This `Certificate` message is an ordered sequence of X.509 certificates, with the first certificate being the certificate belonging to the server, and subsequent certificates (in order) each containing the key the "certifies" (validates) the previous certificate [Rescorla01 p77; RFC2246 Section 7.4.2 per RFC3546 Sections 3.3 and 8]. . This sequence, in the simplest form, is the complete certificate chain beginning with the web server's certificate and ending with the self-signed root certificate. Also note that this last, self-signed root certificate is optional under the assumption that server already has it [RFC2246 Section 7.4.2]. The important implication is that "if certificate chains are being used, then the CA name specified in the `CertificateRequest` message need not refer to the CA that signed the client's certificate, but may instead refer to one of the parent CAs." [Rescorla01 p110]

This was successfully demonstrated in testing; see Appendix B.

APPENDIX B: "HINT LIST" TEST RESULTS

This section describes an actual "hint list" configuration tests using Microsoft IIS 5.0 and Apache 1.3.24 web servers, and Microsoft Internet Explorer 6.0 and Mozilla 1.6 web browsers.

The following test CAs were established. The root CA `tmpCA60` issued the subordinate CA `tmpCA60a`, which issued the subordinate CA `tmpCA60a1`, which issued the end-user certificate `userCA60a1`. Similarly the root CA `tmpCA61` issued the subordinate CA `tmpCA61a`, which issued the subordinate CA `tmpCA61a1`, which issued the end-user certificate `user61a1`. Then the root CA `tmpCA60` issued a certificate to the root CA `tmpCA61`. We will denote this cross certificate as `tmpCA60xc61` (i.e., the issuer DN is `tmpCA60` and the subject DN is `tmpCA61`). Viewing the corresponding certificates graphically:

$$\begin{tikzcd}
\texttt{tmpCA60} \arrow[r] \arrow[d, "(\texttt{tmpCA60xc61})"] & \texttt{tmpCA60a} \arrow[r] & \texttt{tmpCA60a1} \arrow[r] & \texttt{user60a1} \\
\texttt{tmpCA61} \arrow[r] & \texttt{tmpCA61a} \arrow[r] & \texttt{tmpCA61a1} \arrow[r] & \texttt{user61a1}
\end{tikzcd}$$

The IIS web server was configured by putting only the self-signed certificate `tmpCA60` in the "Trusted Root Certificate Authorities\Local Computer" store. (No intermediate CA certificates or cross-certificate were needed.)

The Apache web server was configured[2] by putting only the self-signed certificate `tmpCA60` in the directory `/var/www/conf/ssl.crt` and creating the necessary symbolic link via the command:

```
ln –s tmpCA60.pem `openssl x509 –noout –hash –in tmpCA60.pem`.0
```

Thus, the "hint list" from the Apache web server consisted of only the DN of the `tmpCA60` certificate. The "hint list" from the IIS web server consisted of the DN of the `tmpCA60` certificate plus the DNs of the other CAs in the "Trust Root Certificate Authorities" store that ship natively with Windows 2000.

The following experiments were then conducted. Each time we state that a client certificate appears in the browser's pick list, we are also implying that access to web contents was subsequently granted after choosing that certificate.

B.1 EXPERIMENT #1

The user's IE 6 web browser was configured with:

- The end-entity user certificates `user60a1` and `user61a1`
- The intermediate CA certificates `tmpCA60a1`, `tmpCA61a1`, `tmpCA60a`, and `tmpCA61a`
- The cross certificate `tmpCA60xc61`

[2] It is also important to set the `SSLVerifyDepth` parameter in `httpd.conf` (or `ssl.conf`) to at least the number of certificates in the chain to be traversed to find the CA certificate in the local trust list (`tmpCA60`). In the tests being described, `SSLVerifyDepth` had to be at least 5; larger values of `SSLVerifyDepth` are also acceptable.

(Notice that the self-signed root certificates `tmpCA60` and `tmpCA61` were not configured into the browser.)

In other words, the browser was configured with all certificates necessary to create a certificate path from the end-entity client certificate up to, but not including, the self-signed root CA `tmpCA60`.

Given the above configuration, the user's browser (prompted by the "hint list") correctly displayed a pick list of the two end-entity (client) certificates `user60a1` and `user61a1`, as was expected per the review of the SSL/TLS specifications discussed in Appendix A.

The same behavior was observed in the Mozilla 1.6 browser for the same configuration.

Optionally adding the one self-signed root certificate `tmpCA60` did not change the browsers' behaviors. (Note that only the self-signed root certificate `tmpCA61` has not been loaded into the browser.)

B.2 EXPERIMENT #2

The web browser configurations for the second experiment matched those of the first experiment with one change: the self-signed root certificate `tmpCA61` was also loaded into the browser (i.e., all test certificates are loaded into each browser).

When testing with the IE 6 web browser, the client certificate `user60a1` *did* appear in the browser's pick list, but the client certificate `user61a1` did *not* appear in the browser's pick list. The reason appears to be that as IE 6 was internally assembling the certificate chain *towards* a certificate on the "hint list," it encountered a self-signed root certificate and ended the certificate chain construction, but without checking if that chain was compatible with a certificate on the "hint list."

Interestingly, when testing with the Mozilla 1.6 browser, both client certificates `user60a1` and `user61a1` correctly appeared in the browser's pick list. It appears that the latest Mozilla browser does not implement the same local certificate chain construction logic as the IE 6 browser.

B.3 EXPERIMENT #3

In this experiment, the web servers were configured with both self-signed root CA certificates (`tmpCA60` and `tmpCA61`) in the appropriate trusted root store or directory. The web clients were configured with the user certificates (`user60a1` and `user61a1`) and the intermediate CAs (`tmpCA60a1`, `tmpCA61a1`, `tmpCA60a`, and `tmpCA61a`), but not the self-signed root CA certificates (`tmpCA60` and `tmpCA61`). The presence or absence of the cross certificate (`tmpCA60xc61`) had no affect on test results.

When testing with both browsers, we observed that both client certificates correctly appeared in the browser's pick list.

B.4 EXPERIMENT VARIATIONS

Installing a self-signed root certificate in the web server's "Intermediate Certification Authorities\Local Computer" store instead of the "Trusted Root Certificate Authorities\Local Computer" resulted in the corresponding user certificate *not* appearing in the browser's pick list.

Interestingly, in one experiment variation the self-signed root certificate `tmpCA61` was in the web server's trusted root store, the client certificate `user61a1` was used to access the web page, then the root certificate `tmpCA61` was removed from the web server's trusted root store, then user issued a "refresh page" request. The user was then immediately presented with the error message "HTTP 403.16 - Forbidden: Client certificate untrusted or invalid Internet Information Services." Therefore, the web server must check the certificate's acceptability on each web page access.

B.5 CONCLUSIONS

Experiment #1 seemed to indicate that one way to bridge-enable a web server would be to configure the web server with only the self-signed root certificate of the bridge itself, and configure each user's browser the user's client certificate its issuing chain, and with a cross-certificate between a CA on that issuing chain and the bridge's root. Presumably the client certificate issuer could distribute the appropriate cross-certificate to the user. One advantage of this configuration is that the user's web browser would perform the path discovery operation for the web server. (According to Appendix A, the entire constructed certificate path is passed from the web browser to the web server.) One disadvantage is that every user's browser must be configured with a cross-certificate, which could impede scalability. However, Experiment #2 points out a potentially show-stopping disadvantage (discussed next).

Experiment #2 showed that if the self-signed root certificate at the top of the user's client certificate chain (e.g., `tmpCA61`) is loaded into the user's IE 6 browser, then the user's certificate is absent from the browser's pick list, despite the presence of the cross-certificate to complete the trust path between the CA on the "hint list" and the self-signed root certificate. This is most unfortunate, since:

- The entire certificate issuance chain up to and including the self-signed root certificate is typically distributed with the user's certificate
- Some email applications, such as Outlook and Outlook Express, require the self-signed root certificate be present in the local trusted root store in order for the user's private key to be selectable for signing an email message
- Upwards of 65% of users choose to use IE as their web browser (see Appendix E)

However, Mozilla 1.6 browsers behaved appropriately.

Therefore, to reduce burden on the end user, and to not exclude continued use of IE 6, we find the best option for bridge-enabling a web server is to load all self-signed root certificates that are cross-certified (directly or indirectly) with the bridge in each web server's trusted root store.

APPENDIX C: SOURCE CODE FOR HELPER UTILITIES

C.1 SOURCE CODE FOR "MKLINKS" APACHE HELPER UTILITY

The following source code should be run from within the "hint list" certificate directory specified by the `SSLCACertificatePath` directive in Apache's `httpd.conf` (or `ssl.conf`) file. The purpose is to create the symbolic links needed by `mod_ssl` to quickly identify the correct certificate given only its subject DN.

```
#!/bin/sh
for C in *.pem ; do
    H=`openssl x509 -noout -hash -in $C`.0
    test -s $H || ln -s $C $H
    echo $H = $C
done
## EOF
```

C.2 SOURCE CODE FOR "LOADHINTCERTS" MICROSOFT IIS HELPER UTILITY

The following source code should be run from the temporary directory into which all of the "hint list" certificates were unzipped. The purpose is to automatically load the certificates into the "Trusted Root Certificate Authorities\Local Computer" certificate store.

The following perl script requires the existence of `CAPICOM.dll` on the web server. See the following URL for downloading and installation instructions:
`http://msdn.microsoft.com/library/default.asp?url=/library/en-us/security/Security/getting_ready_to_use_capicom.asp`

(*This perl script is currently under development*.)

APPENDIX D: GOVERNMENT WEB SERVER USAGE SURVEY

This section contains a brief survey of web server and operating system brands used by selected federal government agencies. A total of 91 web sites were surveyed using statistics reported by Netcraft.[3] The first 15 web sites are e-Gov initiatives listed on the Whitehouse's web site.[4] The other 76 web sites were chosen by searching Google using the criteria "`site:.gov`". This data was collected March 19-21, 2004.

Of the 91 web sites surveyed below, the following is the distribution of web server brands used:

- 38 (41.8%) are IIS
- 22 (24.2%) are Apache
- 19 (20.9%) are Netscape
- 8 (8.8%) are Unknown
- 1 (1.1%) is Oracle
- 1 (1.1%) is WebLogic
- 1 (1.1%) is WebSTAR
- 1 (1.1%) is Zeus

The above statistics differ from the general Netcraft claim[5] that two-thirds of web servers are running Apache.

Of the 91 web sites surveyed below, the following is the distribution of operating systems used:

- 34 (37.4%) are Windows
- 32 (35.2%) are Solaris
- 16 (17.6%) are Linux
- 3 (3.3%) are Unknown
- 2 (2.2%) are AIX
- 1 (1.1%) is Compaq Tru64
- 1 (1.1%) is HP-UX
- 1 (1.1%) is IRIX
- 1 (1.1%) is MacOS

The following information is copied verbatim from Netcraft for 15 of the e-Gov initiatives:

- The site www.archives.gov is running unknown on Solaris 8.
- The site www.bpn.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.business.gov is running Netscape-Enterprise/3.6 SP2 on Solaris.
- The site www.disasterhelp.gov is running unknown on Linux.
- The site www.export.gov is running Microsoft-IIS/5.0 on Windows 2000.

---

3 http://uptime.netcraft.com/

4 http://www.whitehouse.gov/omb/egov/downloads/e-gov_initiatives.htm

5 http://news.netcraft.com/archives/2003/11/03/november_2003_web_server_survey.html

- The site www.fedbizopps.gov is running Netscape-Enterprise/6.0 on Solaris.
- The site www.firstgov.gov is running Apache/1.3.27 (Unix) mod_ssl/2.8.14 OpenSSL/0.9.7b on Solaris 8.
- The site www.geodata.gov is running Netscape-Enterprise/6.0 on Solaris 9.
- The site www.golearn.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.govbenefits.gov is running Apache/1.3.27 (Unix) mod_ssl/2.8.14 OpenSSL/0.9.7b on Solaris 8.
- The site www.grants.gov is running WebLogic Server 7.0 SP4 Tue Aug 12 11:22:26 PDT 2003 284033 on Solaris 8.
- The site www.irs.gov is running - on Linux.
- The site www.recreation.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.regulations.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.usajobs.opm.gov is running Microsoft-IIS/5.0 on Windows 2000.

The following information is copied verbatim from Netcraft, but these web sites were not listed as an e-Gov initiative:

- The site www.archives.gov is running unknown on Solaris 8.
- The site www.bpa.gov is running Netscape-Enterprise/3.6 SP3 on Solaris 8.
- The site www.cdc.gov is running Microsoft-IIS/5.0 on Linux.
- The site www.census.gov is running Apache/2.0.48 (Unix) mod_ssl/2.0.48 OpenSSL/0.9.7c PHP/4.3.4 on Solaris 8.
- The site www.cia.gov is running Netscape-Enterprise/4.1 on Solaris 8.
- The site www.cio.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.clinicaltrials.gov is running Apache/2.0.47 (Unix) mod_jk2/2.0.2 on Solaris.
- The site www.commerce.gov is running Apache on Linux.
- The site www.congress.gov is running Web on unknown.
- The site bioguide.congress.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.dhhs.gov is running Netscape-Enterprise/6.0 on Solaris 8.
- The site www.dhs.gov is running Oracle9iAS/9.0.2.2.0 Oracle HTTP Server Oracle9iAS-Web-Cache/9.0.2.2.0 (N) on Linux.
- The site www.doi.gov is running Apache on Solaris.
- The site www.dol.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.oalj.dol.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.efast.dol.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.dot.gov is running Apache/1.3.28 (Unix) on Solaris 8.
- The site www.drugabuse.gov is running WebSTAR NetCloak on MacOS.
- The site www.egov.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.export.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.faa.gov is running Microsoft-IIS/4.0 on unknown.
- The site www.fbi.gov is running Netscape-Enterprise/4.1 on Linux.
- The site www.fcc.gov is running Netscape-Enterprise/4.1 on Solaris.

- The site www.fda.gov is running Microsoft-IIS/5.0 on Linux.
- The site www.fdic.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.fedcirc.gov is running Apache on Linux.
- The site www.federalreserve.gov is running Microsoft-IIS/5.0 on Solaris.
- The site www.fema.gov is running Apache on Linux.
- The site www.usfa.fema.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site training.fema.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.fnal.gov is running Apache/1.3.26 (Unix) mod_fastcgi/2.2.12 mod_ssl/2.8.9 OpenSSL/0.9.6e on unknown.
- The site www.gpoaccess.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.pueblo.gsa.gov is running Apache/1.3.27 (Unix) mod_ssl/2.8.14 OpenSSL/0.9.7b on Solaris.
- The site www.gsa.gov is running Netscape-Enterprise/4.1 on Solaris 8.
- The site listserv.gsa.gov is running Zeus/4.2 on Solaris 8.
- The site apps.fss.gsa.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.iolp.gsa.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site rc.gsa.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.ice.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.irs.gov is running - on Linux.
- The site www.llnl.gov is running Netscape-Enterprise/4.1 on Solaris.
- The site www.nara.gov is running unknown on Solaris.
- The site www.nasa.gov is running Apache/2.0.45 (Unix) mod_perl/1.99_09-dev Perl/v5.6.1 covalent_auth/2.3 DAV/2 CovalentSSL/2.3.3 RSA/SSLC mod_jk/1.2.2-beta-1 on Linux.
- The site rsd.gsfc.nasa.gov is running Apache/2.0.46 on IRIX.
- The site gets.ncs.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site wps.ncs.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.nih.gov is running Netscape-Enterprise/3.5.1I on Compaq Tru64.
- The site www.nist.gov is running Netscape-Enterprise/4.1 on AIX.
- The site www.boulder.nist.gov is running Netscape-Enterprise/4.1 on AIX.
- The site www.ncrr.nih.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.nrel.gov is running Apache on Solaris 8.
- The site www.ntis.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.nws.noaa.gov is running Apache/1.3.27 (Unix) (Red-Hat/Linux) DAV/1.0.3 PHP/4.1.2 mod_perl/1.26 mod_gzip/1.3.26.1a on Linux.
- The site www.geo.nsf.gov is running Netscape-Enterprise/4.1 on Solaris.
- The site www.ntsb.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site apps.opm.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.opm.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.leadership.opm.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site jsearch.usajobs.opm.gov is running Microsoft-IIS/5.0 on Windows 2000.

- The site www.osti.gov is running Apache/1.3.26 (Unix) ApacheJServ/1.1.2 mod_ssl/2.8.10 OpenSSL/0.9.6e on Solaris.
- The site www.ourdocuments.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.peacecorps.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.sba.gov is running Netscape-Enterprise/3.6 SP3 on Solaris.
- The site pro-net.sba.gov is running Netscape-Enterprise/6.0 on Solaris 8.
- The site www.senate.gov is running Netscape-Enterprise/4.0 on Solaris 8.
- The site www.ssa.gov is running "" on Solaris 8.
- The site usembassy.state.gov is running Apache on Linux.
- The site www.technology.gov is running Microsoft-IIS/5.0 on Windows 2000.
- The site www.us-cert.gov is running Apache on Linux.
- The site www.usda.gov is running Apache/1.3.29 (Unix) PHP/4.3.4 on Solaris 9.
- The site www.usdoj.gov is running Netscape-Enterprise/6.0 on Solaris 8.
- The site pubs.usgs.gov is running Apache on Solaris 8.
- The site www.usmint.gov is running Microsoft-IIS/4.0 on NT4/Windows 98.
- The site www.uspto.gov is running Netscape-Enterprise/6.0 on HP-UX.
- The site products.weather.gov is running Apache/1.3.27 (Unix) (Red-Hat/Linux) mod_python/2.7.8 Python/1.5.2 mod_ssl/2.8.12 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26 mod_throttle/3.1.2 on Linux.
- The site www.whitehouse.gov is running Apache on Linux.

APPENDIX E: WEB BROWSERS USAGE STATISTICS

The following tables list monthly usage statistics of web browser brands that have accessed the IIS web site netinfo.mitretek.org as reported by the open source web log analyzer `awstats-5.3`. Table E-1 shows that the first and second most popularly used web browsers are Microsoft's Internet Explorer (IE) and Netscape, respectively. Each month, between 65% and 85% of visitors to the site used a version of Internet Explorer, while 10% to 30% of visitors used a version of Netscape. Table E-2 shows a break down of Internet Explorer usage by specific browser version number. Table E-3 gives a similar break down for Netscape version numbers.

`awstats-5.3` classifies Mozilla browsers as "Unknown" in these tables. The March 2004 statistics show Mozilla versions 3.0, 3.01, and 4.0 are in use.

Table E-1: Usage Statistics of Web Browser Brands by Month

	3/04	2/04	1/04	12/03	11/03	10/03	9/03	8/03	7/03
MS IE	67.9 %	81.7 %	68.1 %	64.5 %	74.7 %	82.8 %	84.5 %	72.2 %	69 %
Netscape	30.1 %	15 %	27.1 %	32.7 %	20.1 %	12.4 %	10.6 %	24.8 %	23.3 %
Unknown	1.5%	2.7%	4.1%	2.4%	4.4%	3.1%	3.8%	1.9%	7%
Konqueror	0.2 %	0.3 %	0.5 %	0.2 %	0.6 %	0.7 %	0.8 %	0.8 %	0.4 %
Opera		0 %	0 %	0 %	0 %	0 %	0 %	0 %	
Lynx or Links		0 %	0 %		0 %	0.7 %	0 %		
Lotus Notes Web Client		0 %	0 %		0 %	0 %			
Galeon				0 %				0 %	

Table E-2: Usage Statistics of Internet Explorer Web Browser Versions by Month

	3/04	2/04	1/04	12/03	11/03	10/03	9/03	8/03	7/03
MSIE 6.0	95.5%	90.7%	94.1%	80.8%	91.3%	88.2%	93.5%	83.6%	83.3%
MSIE 5.5	3.2%	8%	4.4%	18%	7.3%	9.5%	5.3%	12.6%	14%
MSIE 5.23		0%	0.3%	0.5%	0.2%				
MSIE 5.22						0.4%	0%	0%	0%
MSIE 5.16					0%				
MSIE 5.15					0%				
MSIE 5.14			0%				0%		
MSIE 5.01	0.7%	1%	0.8%	0.5%	0.8%	1.1%	1%	3.1%	1.5%
MSIE 5.00	0.1%			0%	0%	0%		0%	
MSIE 5.0	0.2%	0%	0.1%	0%	0.1%	0.5%	0%	0.3%	
MSIE 4.01		0%	0%	0%				0%	
MSIE 3.01						0%			

Table E-3: Usage Statistics of Netscape Web Browser Versions by Month

	3/04	2/04	1/04	12/03	11/03	10/03	9/03	8/03	7/03
Netscape 7.1	6%	2.8%	0.5%	0.3%	11.4%	0%	3.6%	0.5%	0.2%
Netscape 7.02		2.1%	0.4%	0.4%	2.5%	6.6%	5%	1%	4.2%
Netscape 7.01	89.7%	78.1%	92.6%	75.6%	30.9%	14.5%	34.6%	22.1%	33.4%
Netscape 7.0	1%	6.4%	1.1%	5%	5.4%	14.1%	3.8%	11%	10.6%
Netscape 6.2.3				0%				0.2%	0%
Netscape 5.0	1.3%	2.4%	1.5%	0.4%	0.9%	3.7%	4.6%	0.8%	1.9%
Netscape 4.8			0.2%		1.1%			0%	0%
Netscape 4.79	0.9%	6.3%	2.9%	17.5%	46.3%	58.5%	45.4%	53.4%	36.4%
Netscape 4.78		0.2%		0%	0.3%	0.1%	0.2%	8.9%	0.1%
Netscape 4.77		0.1%						0%	0.5%
Netscape 4.76				0.1%	0.2%		0.3%	0.1%	0.6%
Netscape 4.75	0.1%	0%	0.2%	0.2%	0.4%	1.2%	0.2%	0.2%	0.8%
Netscape 4.73		0.1%	0%	0%	0%	0%		1.1%	10.1%
Netscape 4.72			0%				0.8%		
Netscape 4.7	0.5%	0.8%	0%	0%		0.1%			
Netscape 4.6								0%	0.2%
Netscape 4.51							0%		
Netscape 4.5		0%	0%			0%		0%	
Netscape 4.08		0.1%				0.7%	0.8%		0.3%
Netscape 4.04	0.1%								

APPENDIX F: REFERENCES

[Rescorla01] Eric Rescorla, *SSL and TLS—Designing and Building Secure Systems*, Addison-Wesley, 2001. ISBN 0-201-61598-3.

[RFC2246] T. Dierks, C. Allen, “The TLS Protocol Version 1.0,” IETF Proposed Standard RFC2246, January 1999.

[RFC3546] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, T. Wright, “Transport Layer Security (TLS) Extensions,” IETF Proposed Standard RFC3546, June 2003.

APPENDIX G: ACRONYMS

ACES	Access Certificates for Electronic Services
API	Application Programmers Interface
CA	Certificate Authority (or Certification Authority)
CDP	CRL Distribution Point
CRL	Certificate Revocation List
DN	Distinguished Name
FBCA	Federal Bridge Certification Authority
IE	Internet Explorer
IIS	Internet Information Services
ISAPI	Internet Server API
PDP	Policy Decision Point
PEP	Policy Enforcement Point
SAML	Security Assertion Markup Language
SCVP	Simple Certificate Validation Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security