



Installing Apache 2.2 with SSL/TLS on Windows

Published by the Open Source Software Lab at Microsoft. December 2007.

Special thanks to Chris Travers, Contributing Author to the Open Source Software Lab. Most current version will be maintained at <http://port25.technet.com>.



Abstract:

Often SSL or TLS is required to secure data from web applications. Sometimes this is just prudent to prevent confidential or sensitive data from being confiscated. Sometimes this is required by regulations like HIPAA¹ or industry bodies, such as the Payment Card Industry. This guide will show how to install Apache with SSL on Windows.

¹ Health Insurance Portability and Accountability Act in the USA

Information in this document, including URL and other Internet Web site references, is subject to change without notice and is provided for informational purposes only. The entire risk of the use or results from the use of this document remains with the user, and Microsoft Corporation makes no warranties, either express or implied. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

© 2007 Microsoft Corporation. This work is licensed under the Microsoft Public License. The Microsoft Public License is [available here](#).

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Microsoft, Windows, Windows XP, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

All other trademarks are property of their respective owners.

Table of Contents

1	Introduction.....	5
1.1.1	Introduction to SSL and TLS.....	5
1.2	Obtaining Apache with SSL	5
1.2.1	6
1.2.2	Installing the Software	6
1.2.3	Downloading and Installing the Prerequisites	6
1.2.4	Installing over an existing Apache installation	6
1.2.5	Manually installing from Scratch	7
1.3	Generating the Certificate	7
1.3.1	Generating the Certificate Signing Request.....	7
1.3.2	Self-signing the Certificate	9
1.4	Installing the Certificate.....	9
1.4.1	Editing the httpd.conf and related files.	10
1.5	Sample httpd-ssl.conf	10
1.6	Final Thoughts	15
1.7	About the Author	15

1 Introduction

1.1.1 Introduction to SSL and TLS

SSL stands for Secure Socket Layer and is an encryption framework which can be used on individual network connections. In addition to securing data against eavesdropping, it also allows one to authenticate a network connection on one or both sides using a public key infrastructure based on the OSI X.509 standard².

X.509 uses a centralized hierarchy with at most a few trusted entities at its core. These trusted entities issue files which are used to distribute public keys and certify that the bearer of the file is who or what he or she claims to be. The certificates are digitally signed by the certifying entity (called a "certificate authority" or CA) to prevent forgery or alteration, and the client can validate the digital signature against the public key kept on file for the certificate authority and decide whether to trust the certified service. Certificate authorities therefore function sort of like a notary public, validating that parties to a transaction really are who they say they are.

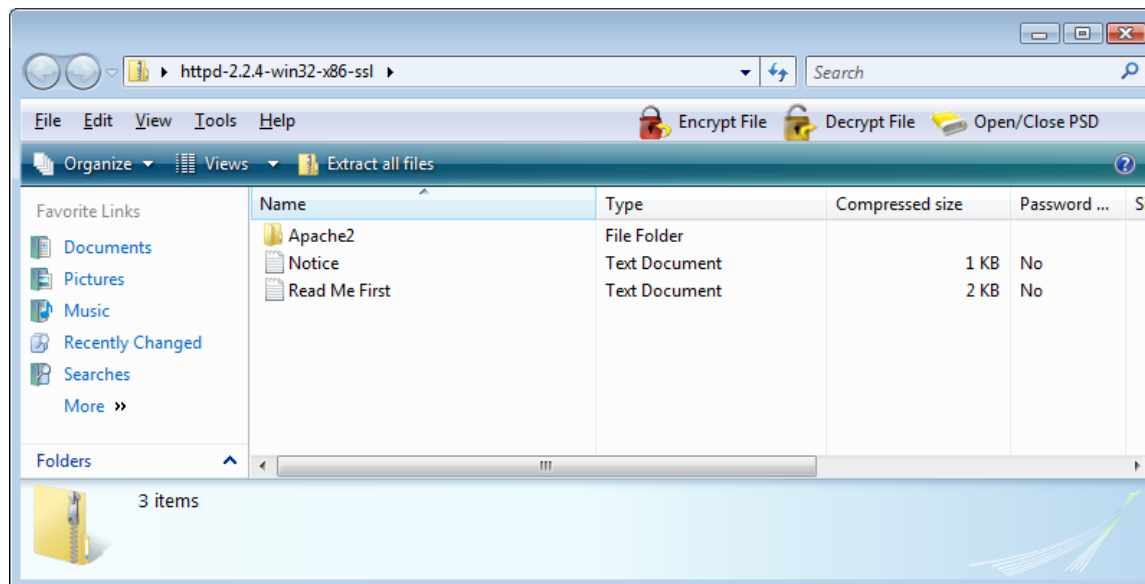
In this tutorial, I cover the generation of a self-signed certificate. Such a certificate does provide protection against eavesdropping, but it does not provide the same level of trust as obtaining one through a trusted and respected certificate authority, especially if the site is to be accessible to the public. In essence, a self-signed certificate tells the user that nobody else is vouching for your identity, while with a purchased certificate, someone else is vouching for your identity.

Transport Layer Security (TLS) is simply the latest version of SSL, and is standardized by the IETF.

1.2 Obtaining Apache with SSL

Binary packages of Apache with SSL for Windows can be obtained from <http://www.apachelounge.com/download/> but unlike the official Apache packages do not come with a Windows installer package. Instead, one simply has a zip file which contains the files and instructions for their installation. Although the installation process is covered in this paper, it is worth reading the "Notice" and "Read Me First" files in the downloaded zip file before continuing, especially if installing a version earlier than 2.2.4.

² Also refer to RFC 2459
<http://port25.technet.com>



1.2.1

1.2.2 *Installing the Software*

Unlike the official Apache packages available at <http://httpd.apache.org/>, these builds do not come with Windows installer packages and therefore require manual installation. Furthermore, a dependency is omitted and so one needs to download another piece of software and install it as well.

1.2.3 *Downloading and Installing the Prerequisites*

The package requires but does not contain the Visual C++ 2005 redistributable run-time package. Before installing the software, download and run the program from the following location:

<http://www.microsoft.com/downloads/details.aspx?FamilyID=200b2fd9-ae1a-4a14-984d-389c36f85647&DisplayLang=en>

The file will install without user input (except for Vista users being asked to allow the installation by User Account Control).

1.2.4 *Installing over an existing Apache installation*

To manually install over an Apache installation of the same version, you should follow the following steps:

1.2.4.1 1. Back up your httpd.conf file

You will need the httpd.conf file later. This file is in "C:\Program Files\Apache Software Foundation\Apache2.2\conf" if you have installed using the installer package from <http://httpd.apache.org>.

1.2.4.2 2. Copy all files from the Apache2 folder in the zip archive to your wwwroot.

By default, the wwwroot is at "C:\Program Files\Apache Software Foundation\Apache2.2\" if installed from the official package. Note that the Apache service must be stopped for this to be successful.

<http://port25.technet.com>

You can expect to be asked whether you want to copy over existing files, and user account control will ask for permission for a few directories as well.

1.2.4.3 3. Copy back the httpd.conf

Once you copy back the httpd.conf file, Apache should be able to run as it did before. You will need to edit this file, but the process is documented below.

1.2.5 *Manually installing from Scratch*

If you are installing this software on a system which has not had Apache installed previously, the easiest approach is to install manually. To do this, simply copy the files from the Apache2 directory in the zip archive to c:\apache2 and run the following command to install the software as a network service:

```
c:\apache2\bin\httpd -k install
```

Windows users wishing to use the Apache Monitor can copy that application or create a link to it in the startup folder. It is in the same folder as httpd.

1.3 Generating the Certificate

Certificates can be generated using Microsoft Certificate Server (part of the Windows Server package), or using a utility like OpenSSL. This tutorial will focus on OpenSSL since this is bundled with the version of Apache we have installed.

The first part of this section will show how to create a Certificate Signing Request, or CSR, which could be sent to a trusted certificate authority in order to obtain a full SSL certificate. If this installation is going to be publicly accessible, this is the preferred method of certificate generation. For testing and development purposes, you may wish to self-sign the CSR yourself which will be covered later.

The first thing that you must do is copy the openssl.cnf file from the wwwroot/conf directory into the c:\openssl\ssl directory (you may need to create this directory first). This is necessary because this is the only location where openssl will look for that configuration file.

1.3.1 *Generating the Certificate Signing Request*

The first stage in generating a certificate is to create a server key. This is done with the openssl utility. Note that the below path may need to be modified depending on where Apache is installed on your system:

```
"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl.exe" genrsa -des3 -out server.key 1024
```

Of course, the command above should be all on one line. Once entered, you will be prompted for a passphrase. Type the same passphrase (between 4 and 511 characters) at the two prompts. Do not lose this passphrase as this will render the certificate useless.

The next stage is to create an unencrypted key. This key must be protected carefully because it is used in key exchange. If the key is compromised, the system becomes vulnerable to a man in the middle attack. Generally this means that only the user that the Apache process on Windows starts

<http://port25.technet.com>

as should have access to the key; the SYSTEM user. Only this user should be able to read the key once it is in place.

The key is decrypted using the following command (again, adjusting the path as necessary and all in one line):

```
"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl.exe" rsa -in server.key  
-out server.pem
```

Now, we can generate an un-signed certificate called a CSR or Certificate Signing Request. The command is:

```
"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl.exe" req -new -key  
server.key -out server.csr
```

Follow the prompts to generate the SSL certificate. Note that the Canonical Name (CN) should be the fully qualified domain name for the server you are creating.

A screen shot of the entire session is below.


```

C:\Users\chris>"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl
l.exe" genrsa -des3 -out server.key 1024
Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
unable to write 'random state'
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:

C:\Users\chris>"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl
l.exe" rsa -in server.key -out server.pem
Enter pass phrase for server.key:
writing RSA key

C:\Users\chris>"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl
l.exe" req -new -key server.key -out server.csr
Unable to load config info from c:/openssl/ssl/openssl.cnf

C:\Users\chris>"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl
l.exe" req -new -key server.key -out server.csr
Enter pass phrase for server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:Washington
Locality Name (eg, city) []:Chelan
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Metatron Technology C
onsulting
Organizational Unit Name (eg, section) []:Open Source on Windows
Common Name (eg, YOUR name) []:localhost
Email Address []:chris@metatrontech.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

C:\Users\chris>

```

1.3.2 *Self-signing the Certificate*

Once you have generated the CSR, you may want to send it to a trusted certificate authority for signing. If the system is just for development and testing use, you may wish to proceed with just a self-signed certificate. Be aware that most browsers will inform the user that the trustworthiness of the certificate is in doubt, so this is not recommended for public-facing applications.

To generate a certificate valid for 30 days, you can use the following command:

```
"c:\Program Files\Apache Software Foundation\Apache2.2\bin\openssl.exe" x509 -req -days 30
-in server.csr -signkey server.key -out server.crt
```

Again the command is all on one line. Enter the key's passphrase when prompted.

1.4 Installing the Certificate

Copy the server.crt and server.pem into the wwwroot\conf\ directory (if installing over Apache, this is probably "c:\Program Files\Apache Software Foundation\Apache2.2\conf").

1.4.1 *Editing the httpd.conf and related files.*

In order for Apache to run with SSL/TLS, you must alter the configuration files and restart the software. Note that Vista users must turn off User Account Control in order to save the new configuration files.

In the httpd.conf file, change the following lines. Note that the easiest way to do this is via the Find or Search interface of your text editor. In each of these cases, all you need to do is remove the leading # sign in order to uncomment the line:

```
#Loadmodule ssl_module modules/mod_ssl.so  
and  
#Include conf/extra/httpd-default.conf
```

In the wwwroot\conf\extras\ directory (by default "c:\Program Files\Apache Software Foundation\Apache2.2\conf\extras" if installing over an existing Apache instance), modify the following lines:

Change (all one line):
SSLCertificateKeyFile C:/Program Files/Apache Software
Foundation/Apache2.2/conf/server.key

To (all one line):

```
SSLCertificateKeyFile "C:/Program Files/Apache Software  
Foundation/Apache2.2/conf/server.pem"
```

The only characters that will likely need to be changed are the last three on that line. Of course, if you want to store the key somewhere else, you will want to modify the path accordingly. If there are spaces in the path, add quotes around the entire argument.

In the httpd-ssl.conf file, you may encounter one further problem depending on how you have installed mod_ssl. If you have installed over the top of an existing Apache installation, you may find that the paths in the file which contain spaces prevent Apache from starting. You may find that you need to go through the file looking for paths with spaces and quoting them. This is only a problem in this file, not the httpd.conf since that is tested with the application bundle.

A working httpd-ssl.conf file for an installation over the top is included below. It may provide a better starting point than the one bundled with the Apache windows installer package.

1.5 **Sample httpd-ssl.conf**

```
#  
# This is the Apache server configuration file providing SSL support.  
# It contains the configuration directives to instruct the server how to  
# serve pages over an https connection. For detailing information about these  
# directives see <URL:http://httpd.apache.org/docs/2.2/mod/mod_ssl.html>  
#  
# Do NOT simply read the instructions in here without understanding
```

```
# what they do. They're here only as hints or reminders. If you are unsure
# consult the online docs. You have been warned.
#
#
# Pseudo Random Number Generator (PRNG):
# Configure one or more sources to seed the PRNG of the SSL library.
# The seed data should be of good random quality.
# WARNING! On some platforms /dev/random blocks if not enough entropy
# is available. This means you then cannot use the /dev/random device
# because it would lead to very long connection times (as long as
# it requires to make more entropy available). But usually those
# platforms additionally provide a /dev/urandom device which doesn't
# block. So, if available, use this one instead. Read the mod_ssl User
# Manual for more details.
#
#SSLRandomSeed startup file:/dev/random 512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random 512
#SSLRandomSeed connect file:/dev/urandom 512

#
# When we also provide SSL we have to listen to the
# standard HTTP port (see above) and to the HTTPS port
#
# Note: Configurations that use IPv6 but not IPv4-mapped addresses need two
# Listen directives: "Listen [::]:443" and "Listen 0.0.0.0:443"
#
Listen 443

##
## SSL Global Context
##
## All SSL configuration in this context applies both to
## the main server and all SSL-enabled virtual hosts.
##

#
# Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl

# Pass Phrase Dialog:
# Configure the pass phrase gathering process.
# The filtering dialog program (`builtin' is a internal
# terminal dialog) has to provide the pass phrase on stdout.
SSLPassPhraseDialog builtin
```

```
# Inter-Process Session Cache:
# Configure the SSL Session Cache: First the mechanism
# to use and second the expiring timeout (in seconds).
#SSLSessionCache dbm:C:/Program Files/Apache Software
Foundation/Apache2.2/logs/ssl_scache
SSLSessionCache "shmcb:C:/Program Files/Apache Software
Foundation/Apache2.2/logs/ssl_scache(512000)"
SSLSessionCacheTimeout 300

# Semaphore:
# Configure the path to the mutual exclusion semaphore the
# SSL engine uses internally for inter-process synchronization.
SSLMutex default

##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>

# General setup for the virtual host
DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
ServerName chris-PC.metatrontech.com:443
ServerAdmin admin@metatrontech.com
ErrorLog "C:/Program Files/Apache Software Foundation/Apache2.2/logs/error_log"
TransferLog "C:/Program Files/Apache Software Foundation/Apache2.2/logs/access_log"

# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# SSL Cipher Suite:
# List the ciphers that the client is permitted to negotiate.
# See the mod_ssl documentation for a complete list.
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

# Server Certificate:
# Point SSLCertificateFile at a PEM encoded certificate. If
# the certificate is encrypted, then you will be prompted for a
# pass phrase. Note that a kill -HUP will prompt again. Keep
# in mind that if you have both an RSA and a DSA certificate you
# can configure both in parallel (to also allow the use of DSA
# ciphers, etc.)
SSLCertificateFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/server.crt"
#SSLCertificateFile C:/Program Files/Apache Software Foundation/Apache2.2/conf/server-
dsa.crt

# Server Private Key:
# If the key is not combined with the certificate, use this
```

```
# directive to point at the key file. Keep in mind that if
# you've both a RSA and a DSA private key you can configure
# both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile "C:/Program Files/Apache Software
Foundation/Apache2.2/conf/server.pem"
#SSLCertificateKeyFile C:/Program Files/Apache Software Foundation/Apache2.2/conf/server-
dsa.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile C:/Program Files/Apache Software Foundation/Apache2.2/conf/server-
ca.crt

# Certificate Authority (CA):
# Set the CA certificate verification path where to find CA
# certificates for client authentication or alternatively one
# huge file containing all of them (file must be PEM encoded)
# Note: Inside SSLCACertificatePath you need hash symlinks
# to point to the certificate files. Use the provided
# Makefile to update the hash symlinks after changes.
#SSLCACertificatePath C:/Program Files/Apache Software Foundation/Apache2.2/conf/ssl.crt
#SSLCACertificateFile C:/Program Files/Apache Software Foundation/Apache2.2/conf/ssl.crt/ca-
bundle.crt

# Certificate Revocation Lists (CRL):
# Set the CA revocation path where to find CA CRLs for client
# authentication or alternatively one huge file containing all
# of them (file must be PEM encoded)
# Note: Inside SSLCARevocationPath you need hash symlinks
# to point to the certificate files. Use the provided
# Makefile to update the hash symlinks after changes.
#SSLCARevocationPath C:/Program Files/Apache Software Foundation/Apache2.2/conf/ssl.crl
#SSLCARevocationFile C:/Program Files/Apache Software
Foundation/Apache2.2/conf/ssl.crl/ca-bundle.crl

# Client Authentication (Type):
# Client certificate verification type and depth. Types are
# none, optional, require and optional_no_ca. Depth is a
# number which specifies how deeply to verify the certificate
# issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth 10

# Access Control:
# With SSLRequire you can do per-directory access control based
```

```
# on arbitrary complex boolean expressions containing server
# variable checks and other lookup directives. The syntax is a
# mixture between C and Perl. See the mod_ssl documentation
# for more details.
#<Location />
#SSLRequire (  %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#    and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#    and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#    and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#    and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20    ) \
#    or %{REMOTE_ADDR} =~ m/^192\.76\.162\. [0-9]+$/
#</Location>

# SSL Engine Options:
# Set various options for the SSL engine.
# o FakeBasicAuth:
#   Translate the client X.509 into a Basic Authorisation. This means that
#   the standard Auth/DBMAuth methods can be used for access control. The
#   user name is the `one line' version of the client's X.509 certificate.
#   Note that no password is obtained from the user. Every entry in the user
#   file needs this password: `xxj31ZMTZzkVA'.
# o ExportCertData:
#   This exports two additional environment variables: SSL_CLIENT_CERT and
#   SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#   server (always existing) and the client (only existing when client
#   authentication is used). This can be used to import the certificates
#   into CGI scripts.
# o StdEnvVars:
#   This exports the standard SSL/TLS related `SSL_*' environment variables.
#   Per default this exportation is switched off for performance reasons,
#   because the extraction step is an expensive operation and is usually
#   useless for serving static content. So one usually enables the
#   exportation for CGI and SSI requests only.
# o StrictRequire:
#   This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#   under a "Satisfy any" situation, i.e. when it applies access is denied
#   and no other module can change it.
# o OptRenegotiate:
#   This enables optimized SSL connection renegotiation handling when SSL
#   directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire
<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>
<Directory "C:/Program Files/Apache Software Foundation/Apache2.2/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant shutdown
```

```
# approach is that mod_ssl sends the close notify alert but doesn't wait for
# the close notify alert from client. When you need a different shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
#   This forces an unclean shutdown when the connection is closed, i.e. no
#   SSL close notify alert is send or allowed to received. This violates
#   the SSL/TLS standard but is needed for some brain-dead browsers. Use
#   this when you receive I/O errors because of the standard approach where
#   mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
#   This forces an accurate shutdown when the connection is closed, i.e. a
#   SSL close notify alert is send and mod_ssl waits for the close notify
#   alert of the client. This is 100% SSL/TLS standard compliant, but in
#   practice often causes hanging connections with brain-dead browsers. Use
#   this only for browsers where you know that their SSL implementation
#   works correctly.
# Notice: Most problems of broken clients are also related to the HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive" for this.
# Similarly, one has to force some clients to use HTTP/1.0 to workaround
# their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
# "force-response-1.0" for this.
BrowserMatch ".*MSIE.*" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0

# Per-Server Logging:
# The home of a custom SSL log file. Use this when you want a
# compact non-error SSL logfile on a virtual host basis.
CustomLog "C:/Program Files/Apache Software Foundation/Apache2.2/logs/ssl_request_log" \
    "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
```

1.6 Final Thoughts

Apache with SSL on Windows poses a number of unique difficulties because it is not bundled with the default Apache installation package. These difficulties are minor, but are likely to be somewhat foreign to administrators who shy away from text files and command-line interfaces. I hope that this paper helps people overcome the most common problems.

1.7 About the Author

Chris Travers is the owner of Metatron Technology Consulting, a firm devoted to helping businesses leverage open source software. He has over seven years of experience with Apache on both Windows and Linux.