

COBALT SECURE WEB SERVER 1.0

DEVELOPER EDITION



Cobalt Secure Web Server (SSL) Cobalt Microserver, Inc

Features:

- 128 bit Encryption*
- Based on Redhat's Secure Server and Apache-SSL
- SSL 3.0
- CGI support
- Certificate Generation
- Secure Server Documentation
- Single Server Advanced Cryptography License from RSA Data Security
- BSAFE Encryption Engine from RSA Data Security

Limitations:

*Due to restrictions imposed by the US Government, this software may not be distributed outside the U.S. and Canada

© 1998 Cobalt Networks, Inc. Contains Software & Licenses from the following organizations/companies: RedHat Software, RSA Data Security.



Contents

1. Introduction
2. Installing your Secure Server
 - 1.1 Installing the Package File
 - 1.2 Configuring your server
 - 1.2.1 Default Configuration
 - 1.2.2 Configuration files
 - 1.2.3 Generating a test certificate
 - 1.2.4 Starting the server
 - 1.3 Obtaining a Certificate
 - 1.3.1 About Passwords
 - 1.3.2 Certificate Authorities
 - 1.3.3 Gather Proof of Right Documents
 - 1.3.4 Generate a CSR (Certificate Signing Request)
 - 1.3.5 Submit your information
 - 1.4 Installing your Certificate
3. Keeping your Server Secure
4. Customizing your Server
 - 4.1 Directories & File Locations
5. Links
6. FAQ
7. Contacting Cobalt
8. Acknowledgements
9. License & Warranties

1. Introduction

The Cobalt Secure Server 1.0 Developer Edition is designed for developers.

The Cobalt Secure Web Server is designed for developers who wish to deploy applications that use encrypted communication conforming to SSL (Secure Sockets Layer) standards. The core technology behind the Secure Web Server is the Apache 1.2.6 web server with SSL extensions, and SSLeay, an implementation of Netscape's Secure Sockets Layer. In order to use this software effectively, you will need to have some familiarity with the Unix operating system, specifically directory structures & telnet.

About our Implementation

The Cobalt Secure Web Server is an implementation of the Apache-SSL server. There are two important differences between Cobalt's Secure Server and the publicly available version of Apache-SSL with SSLeay.

1. The Cobalt Secure Web Server contains a license for RSA's BSAFE encryption library. The BSAFE library contains routines that make SSL communications possible. It is necessary to have this license to deploy commercial SSL applications in the US and Canada.
2. RSA and Redhat Software have ported and optimized the BSAFE routines for the MIPS architecture used by the Cobalt Microserver family. Encryption performance is superior to the standard SSLeay routines.

The standard configuration includes the Apache server with no added modules. This is designed for minimum memory footprint and maximum speed.

If you want to include other modules to customize the Apache server, you can compile your own version of apache-ssl and link to the licensed version of the BSAFE library installed on the Cobalt Microserver. You must have a separate license for each Cobalt Microserver. Cobalt Networks provides no support for you to recompile Apache.

Installing the Cobalt Secure Web Server package

Installing the Package file

The procedure for installing the package is very easy.

Start by downloading the appropriate package to your hard disk. Then connect to the Admin server of the Cobalt Qube. Navigate to the install software tab in the maintenance section.

Type in the pathname and filename of the package file. Alternately, click on the browse button and select the package file from your hard disk. Then enter the admin password and click the "install a '.pkg' Package" button.

The install process takes about a minute. You will know that the software is installed when you see "SSL Development Server Release 1.0" listed in the Software window.



Configuring your server

Default configuration:

The secure server runs from `/home/httpsd`

HTML pages are served from `/home/httpsd/html/` by default. On the Cobalt RaQ, only the admin user may ftp files to this directory.

Configuration Files

The default location for your server configuration directory is:
`/etc/httpsd/conf`

Make a backup before modifying `/etc/httpsd/conf/httpd.conf`. This file contains SSL specific configuration options.

HostnameLookups: `off` / `on`. This determines whether hostnames or IP addresses are logged. Setting this to `on` results in a nominal performance penalty since the DNS software must lookup the address of the visiting host.

User: `httpd`. This option sets the user that will be running the server process. Cobalt uses the `httpd` user for the public site. Do not attempt to run the server under any other user as server instability will result.

Group: `httpd`. This option sets the group that will be running the server process. Do not attempt to run the server under any other group as server instability will result.

TransferLog: `/var/log/httpsd/access_log`. This determines the location of the host access log file. Web statistics and report software use the access log file to generate reports.

ErrorLog: `/var/log/httpsd/error_log`. This determines the location and name of the error log file. If your server fails to start properly, this file will give the reason.

ServerAdmin: `root@localhost`. This determines the email address of your server's administrator. This is normally also the Cobalt Qube administrator.

To learn more about other options, consult the Apache 1.2.6 documentation at <http://www.apache.org>.

Generating a test certificate

In order to start your secure server, you must first generate a test certificate.

You will find a short certificate generation script called 'maketest' in the /etc/httpsd/conf directory. Run this program from the command line to generate a temporary certificate.

The program will generate a 1024 bit private key and then ask you a number of questions. Here is an example:

```
[root@qube conf]# ./maketest
Using configuration from /etc/ssl/lib/sslkey.cnf
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '/etc/httpsd/conf/httpsd.pem'
-----
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]:California
Locality Name (eg, city) []:Mountain View
Organization Name (eg, company) [Internet Widgits Pty Ltd]:My Company, Inc.
Organizational Unit Name (eg, section) []:Secure Division
Server Host Name []:ssl.domain.com
Email Address []:ssladmin@domain.com
[root@qube conf]#
```

Here is what each entry means:

Country Name: the two-letter code for your country (ex. US, CA)
State or Province: The state or province name spelled out (ex. California)
Locality Name: The name of your city
Organization Name: Your company or organization name
Organization Unit: Your department or company section
Server Host Name: Your hostname & domain name (ex. ssl.cobaltnet.com)
Email Address: The webmaster administrator address (admin@xyz.com)

Once you have typed in all the information, you will be returned to the command prompt and you are ready to start the server. You will need to use this information again when you generate a certificate and submit it to a Certificate Authority.

The maketest shell script creates a couple of files in your secure configuration directory.

```
/etc/httpsd/conf/httpsd.pem
/etc/httpsd/conf/somenumbr.0
```

Starting the Server

Always use the script `/etc/rc.d/init.d/httpd` to start, stop and restart the server.

You can start the server manually by typing in:

`/etc/rc.d/init.d/httpd start`

If there is no error, you can use your browser to connect to the server.

You can also verify that the server is running from the command line,

type: **`ps uax | grep httpd`**

To stop the server, type:

`/etc/rc.d/init.d/httpd stop`.

To reload configuration files, type:

`/etc/rc.d/init.d/httpd restart`

Never send a HUP signal to the server, doing so may result in server instability.

When you first connect to the secure server from your browser, you will have to go through a certificate authentication process. This is because the server's digital ID is not guaranteed by any CA (Certificate Authority).

Obtaining a Certificate

About Passwords

If you encrypt your server with a PEM passphrase, the Cobalt Qube will be unable to start the secure server by itself. This is because Apache-SSL requires that you enter the pass phrase when you start the program. If you want the secure server to startup automatically, you should not use a PEM passphrase.

Certificate Authorities

You can obtain a certificate from a number of Certificate Authorities. The two most common authorities are Verisign and Thawte.

Company	Certificate Cost	Term	Renewal Fee	Web Site
Verisign	\$349	1 year	\$249	www.verisign.com
Thawte	\$125	1 year	\$100	www.thawte.com

All modern browsers (IE 3 or newer & Netscape 3 or newer) work with certificates from either authority.

Step 1: Gather Proof of Right Documents

Certificate Authorities require some form of proof that you are who you claim you are. This can include a number of things. Check with your CA for details.

- Your DUNS Number (Verisign only)
To obtain a DUNS number, visit: <http://www.dnb.com/aboutdb/dunsform.htm>
- Proof of your right to use the certified organization name (Articles of incorporation)
- Proof of your registration of the domain name (from the InterNIC whois database)
To obtain your domain details, visit: <http://rs.internic.net/>
- A letter of authorization from an agent of your company or organization
Thawte has an online form that generates this letter

Step2: Generate a CSR (Certificate Signing Request)

This procedure is similar to generating a test-certificate.

Script Procedure

The following script will generate a 1024 bit key, encrypt it using the triple-DES cipher, and create a CSR based upon it.: `/etc/httpsd/conf/makecsr` makes three files, one of which you will submit to the certificate authority. You will enter the same information that you used to generate the test certificate.

Manual Procedure

Use the manual procedure if you want to learn more about the key generation process. The utilities that you use to generate the private key and CSR come with SSLeay and are installed under `/etc/ssl/bin`. First, select five large and relatively random files from your hard drive (compressed log files are good). These will act as your random seed enhancers. We refer to them as `file1:file2:....:file5` below.

```
cd /etc/httpsd/conf
```

```
/usr/sbin/ssleay genrsa -des3 -rand /dev/urandom 1024 > /etc/httpsd/conf/httpsd.key
```

It is not necessary, but you can replace `'httpsd.key'` with `'www.yourdomain.key'`

You will have to modify `httpsd.conf` to reflect this change

Omit the `'-des3'` option if you do not want to use a passphrase*

```
/usr/sbin/ssleay req -new -key ./httpsd.key > httpsd.csr
```

```
/usr/sbin/ssleay req -new -x509 -key ./httpsd.key > httpsd.crt
```

PLEASE backup your `httpsd.key` and make a note of the passphrase and challenge phrase.

NOTE: when asked for your Server Host Name, enter the host & domain name of your web server (i.e. "www.mycorp.com" or "secure.foo.com"). The prompt on standard SSLeay distributions asks for "YOUR name", which is misleading.

* If you want to avoid pass phrases, and you are convinced that your machine is secure, the leave out the `"-des3"` portion of the key generation command. If you do this, PLEASE ensure that the key file can only be read by root. Your server starts up as root, so it can read the key, then it switches to whatever user you're running it as (nobody or httpd). We

recommend that you do a "**chown root.root file.key; chmod 400 file.key**" to make sure you never lose it.

Important Configuration Files:

SSLCertificateFile	/etc/httpsd/conf/httpsd.crt (will be replaced by CA key)
SSLCertificateKeyFile	/etc/httpsd/conf/httpsd.key
CertificateRequestFile	/etc/httpsd/conf/httpsd.csr

The file **/etc/httpsd/conf/httpsd.key** file is your secret key, and is linked in the **/etc/httpsd/conf/httpd.conf** file by default.

The file **/etc/httpsd/conf/httpsd.crt** is your self-signed certificate. You use it as a temporary certificate while you are waiting for a real certificate from your CA.

The file **/etc/httpsd/conf/httpsd.csr** is your CSR (certificate request). The important bit looks something like this:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBPTCB6AIBADCBhDELMAkGA1UEBhMCWkExFTATBgNVBAGTDFdlc3Rlcm4gQ2Fw
ZTESMBAGA1UEBxMJQ2FwZSBUB3duMRQwEgYDVQKKEwtPcHBvcnR1bml0aTEYMBYG
A1UECxMPT25saW5lIFNlcnZpY2VzMRowGAYDVQQDExF3d3cuZm9yd2FyZC5jby56
YTBaMA0GCSqGSIb3DQEBAQUAA0kAMEYCCQDT5oxxeBWu5WLHD/G4BJ+PobiC9d7S
6pDvAjuyC+dPanL0d91tXdm2j190D1kgDoSp5ZyGSgwJh2V7diuuPlHDAgEDoAAw
DQYJKoZIhvcNAQEEBQADQQBf8ZHlU4H8ik2vZQngXh8v+iGnAXD1AvUjuDPCWzFu
pReiq7UR8Z0wiJBeaqiuvTDnTFMz6oCq6htdH7/tvKhh
-----END NEW CERTIFICATE REQUEST-----
```

(This section reprinted/modified with permission of Thawte Consulting: <http://www.thawte.com/certs/server/keygen/apachessl.html>)

Step 3: Submit your information

Once you have your information prepared, decide which Certificate Authority you want to register with. The two most common CA's in the US are Verisign & Thawte. Here are links to their registration pages:

Verisign - <http://digitalid.verisign.com/server/enrollIntro.htm>
Thawte - <http://www.thawte.com/certs/server/request.html>

Each Service requires that you generate and submit a Certificate Signing Request (CSR), which you created in Step 2. Copy the contents of **/etc/httpsd/conf/httpsd.csr** into your Certificate Authority web submission form. Submit the other necessary data and wait for the certificate to be returned to you.

Installing your Certificate

Once you receive your certificate back from the CA, you should replace the contents of the key file: `/etc/httpsd/conf/httpsd.crt`

Restart your web server and you are ready to server secure pages to everyone on the net!

Keeping your Server Secure

If you're planning to develop commerce applications on a Cobalt Microserver, you want your transaction environment to be as secure as possible. It's important to follow a few design guidelines.

IMPORTANT: Running a secure server means that the connection to your customer/client is secure. Be proactive with your site design. Don't let other data-flow processes interrupt that security.

1. Never store unencrypted credit card numbers on the server - This holds true for customer data in general.
2. Never send credit card or other transaction data over the network unencrypted
If you send transactions over the net for latter processing via email, be sure to encrypt the data before you send it. Consider using ssh to send data.
3. Don't turn on Server Side Includes (SSI) They are considered a security hole - many people use them regardless, and open themselves up to attack.
4. If at all possible, don't store your PEM password on the Microserver
This may not be practical if you must restart your server automatically. Cobalt Microservers are designed to run for months (even years) at a time without rebooting. Invest in a UPS.
5. Maintain a checksum on your server data (html, cgi, static databases) to warn you of attacks.
6. Don't run multiple secure servers on the same machine.
7. Backup your PEM password and key in a safe place.

Customizing your Server

Assumption: You understand how to administer the Apache web server. For documentation about the Apache server, see <http://www.apache.org>.

Cobalt Microservers run several web servers simultaneously.

1. The Public server (port 80)
2. The Admin server (port 81)
3. The Secure server (port 443)

Why would I want to customize my server?

You may want to customize the web root directory. Customizing the server may be necessary to produce commerce applications.

Customization Directions

Note: The Public Server configuration files are in /etc/httpd/conf - don't edit these by mistake! The secure web server configuration files are located in /etc/httpsd/conf.

Each configuration file (httpd.conf, access.conf, srm.conf) may be edited using your own Apache directives. You can review the entire list of Apache directives at: <http://www.apache.org/docs/mod/directives.html>.

Directories & File locations

SSLeay: /usr/sbin/ssleay
SSLeay key config file: /etc/ssl/lib/ssleay.cnf
Cobalt Secure Server binary: /usr/sbin/httpsd
Cobalt Secure Server configuration directory: /etc/httpsd/conf
Cobalt Secure Server master configuration file: /etc/httpsd/conf/httpd.conf
Test Cert generator: /etc/httpsd/conf/maketest
Certificate request script: /etc/httpsd/conf/makecsr

Links

Cobalt Networks: <http://www.cobaltnet.com/>
Apache-SSL: <http://www.apache-ssl.org/>
Cryptsoft SSLeay focal point: <http://www.cryptsoft.com/>
SSLeay FAQ: <http://psych.psy.uq.oz.au/~ftp/Crypto/>
Introduction to SSL using SSLeay: <http://www.camb.opengroup.org/RI/www/prism/wwwj/>
Apache Quick Reference Card: <http://www.ford-mason.co.uk/resources/apache-refcard/>
Apache Perl Integration Project: <http://perl.apache.org/>
PHP HyperText Processor: <http://www.php.net/>
Apache-SSL SRPMs: <http://www.replay.com/redhat/apache.html>
Mod_SSL: http://www.engelschall.com/sw/mod_ssl/

FAQ

1. Error: "no start line:pem_lib.c" or "no end line:pem_lib.c" message.

The Cobalt Secure Server uses SSLeay for its security routines. SSLeay is very fussy about the format of certificate requests and certificates. In particular it wants BEGIN and END lines that look like this for certificates:

```
-----BEGIN CERTIFICATE-----  
...data data data...  
This is actually your  
certificate BASE64 encoded  
to make it easy to transport  
by mail.  
-----END CERTIFICATE-----
```

If your cert has trailing spaces after the BEGIN or END lines it will bomb with the error you see. The problem usually is that your browser created trailing spaces when you cut and pasted the cert from the browser window into a text editor to create the certificate text file. Just remove the trailing spaces (on UNIX you may also need to get rid of CTRL-M characters as well) and all will be well. ;-)

2. I get asked for a passphrase whenever I the server starts.

When you generated a key you asked SSLeay to encrypt it. This is a good thing because it means that if a hacker gets your private key they still need to guess the pass phrase used to encrypt it... not an easy task at all. However, it means that when the server starts up it needs to get the pass phrase from you so that it can get at the raw key.

3.How do I get rid of the passphrase requirement?

You can create an unencrypted copy of the key as follows:

```
ssleay rsa -in file1.key -out file2.key
```

Now file2.key will contain an unencrypted copy of the key. If you point your server at this it will not prompt you for a pass phrase. HOWEVER, if anyone gets this key they will be able to impersonate you on the net. PLEASE make sure that the permissions on that file are such that only root or the web server user can read it (preferably get your web server to start as root but run as another server, and have the key readable only by root).

4.How do I know which certs match which key?

The private key contains a series of numbers. Two of those numbers form the "public key", the others are part of your "private key". The "public key" bits are also embedded in your certificate (we get them from your CSR). To check that the public key in your cert matches the public portion of your private key, you need to view the cert and the key and compare the numbers.

To view the cert:

```
ssleay x509 -noout -text -in certfile
```

To view the key:

```
ssleay rsa -noout -text -in keyfile
```

The "modulus" and the "public exponent" portions in the key and the certificate must match.

5.The browser doesn't recognize the issuer certificate.

Your server is presenting the wrong certificate to the browser - most probably your "self signed, temporary certificate". To check the contents of the cert being presented by the server all you need to do is use your browser. Connect to your secure site and then do the following. In Navigator simply click on the Security button and then click on "View Certificate". On the left you'll see who the cert belongs to - it should be you ;-). On the right you'll see the details of the people that signed the certificate - it should be us. If the cert issuer has your details then you have Apache configured to point at the temporary certificate.

The configuration information generally resides in a file in your Apache directory structure - probably "httpd.conf".

Update your configuration files as follows:

```
SSLCertificateFile mycertfile.crt  
SSLCertificateKeyFile mykeyfile.key
```

You can have the key and the cert in a single file if you want, but we recommend separating them for clarity.

6.The documentation asks for a PEM format cert but you won't give me one.

The Apache-SSL documentation, and the docs for the SSLeay toolkit, refer to certificates and certificate requests as "PEM" files. They are not. Apache-SSL, like all SSL secure servers, uses the X.509 certificate format. Now X.509 certificates are binary files, which are difficult to send

around by mail. So SSLeay stores them in BASE64 encoded format, between -----BEGIN and END----- lines. The BASE64 encoding was defined as part of the Privacy Enhanced Mail (PEM) specification, which is why the documentation calls them "PEM-format" files.

However, some servers use a real Privacy-Enhanced Mail (PEM) format certificate packet (Lotus Domino, Webstar/SSL etc.). That's why one of the download formats for server certs is "PEM format". This IS NOT THE FORMAT YOU WANT. You want the "Standard" format, which is a BASE64 encoded X.509 certificate.

7. When you try to access a Thawte certified site, IE 4 for the Macintosh displays an error that says 'The identity certificate has expired'

There is a bug in IE4 or the Thawte certification process. As there is no workaround for the bug as of this printing, see Thawte's site documentation at <http://www.thawte.com/support/server/browsers.html>

(This section reprinted/modified with permission of Thawte Consulting: <http://www.thawte.com/support/server/apachessl.html>)

Contacting Cobalt Networks, Inc.

Cobalt Networks, Inc
440 Clyde Ave Building B
Mountain View, Calif. 94034
Tel: (650) 930-2500
Fax: (650) 930-2501
Web: <http://www.cobaltnet.com/>

Acknowledgements

©1998 Cobalt Networks, Inc. All rights reserved.

Cobalt Networks and Cobalt RaQ are trademarks of Cobalt Networks, Inc. All other company, brand, and product names may be registered trademarks or trademarks of their respective companies and are hereby recognized.

License & Warranties

©1998 Cobalt Networks, Inc., RedHat Software, RSA Data Security All rights reserved.

PORTIONS OF THIS PRODUCT ARE COVERED UNDER THE GNU GENERAL PUBLIC LICENSE

THIS PRODUCT MAY NOT BE EXPORTED TO, OR SOLD TO A NATIONAL OF, ANY COUNTRY OTHER THAN THE UNITED STATES AND CANADA.

THIS SOFTWARE IS PROVIDED BY COBALT NETWORKS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL COBALT NETWORKS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This publication and the information herein is furnished AS IS, subject to change without notice, and should not be construed as a commitment by Cobalt Networks, Inc. Furthermore, Cobalt Networks, Inc., assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and noninfringement of third party right.