yaSSL Embedded Web Server

User Manual

May 21, 2012
Version 1.4

Table of Contents

Chapter 1: Introduction

yaSSL focuses on providing embedded security solutions with an emphasis on speed and size. With dual-licensed products to cater to a diversity of users ranging from the hobbyist to the user with commercial needs, yaSSL is happy to help customers and community members in any way we can.

The yaSSL Embedded Web Server is a version of the Mongoose embedded web server with SSL functionality built in with CyaSSL. With a focus on what our customers need, the yaSSL Embedded Web Server was based on Mongoose because of its small size, excellent code base and community, and its portability to real time and embedded operating systems.

This manual is written as a technical guide to the yaSSL Embedded Web Server. It includes how to obtain support as well as licensing details. It also gives an explanation of the yaSSL Embedded Web Server feature set.

Chapter 2: Obtaining Support

2.1 Where to Find Help and Information

For general product support, we maintain an online forum for the yaSSL related product family. Please post to our forums or contact us at **support@yassl.com** with any questions.

yaSSL Forums (http://www.yassl.com/forums)

For information regarding our products, questions regarding licensing, or general comments, please contact us at **info@yassl.com**. For support packages, please see Chapter 3.

2.2 Bugs Reports and Support Issues

If you are submitting a bug report or asking about a problem you are encountering, please include the following information with your submission:

1. yaSSL Embedded Web Server version number
2. Operating System version
3. Compiler version
4. The exact error you are seeing
5. A description of how we can reproduce or try to replicate this problem

With the above information, we will do our best to resolve your problems. Without this information, it is very hard to pinpoint the source of the problem. We value your feedback and make it a priority to get back to you as soon as possible.

Chapter 3: Licensing

3.1 Open Source

The founders and employees of yaSSL believe in Open Source Systems. As such, the source code for yaSSL products is available for all to use, modify, test and enjoy under the GPL. The yaSSL Embedded Web Server may be modified to the needs of the user as long as the user adheres to version two of the GPL License. The GPL license can be found on the gnu.org website (http://www.gnu.org/licenses/old-licenses/gpl-2.0.html).

We do not reserve features! As such, everything available in our commercial version is also available in our GPL version. For more information on our licensing, please see our web site or contact **info@yassl.com**.

3.2 Commercial Subscriptions

Businesses and enterprises who wish to use or incorporate the yaSSL Embedded Web Server into proprietary appliances or other commercial software products for re-distribution must license commercial versions. Commercial licenses for the yaSSL Embedded Web Server are available for $5,000 USD. Subscriptions are generally issued for one product and include unlimited distribution.

3.3 Support Packages

Support packages for the yaSSL Embedded Web Server are available on an annual basis directly from yaSSL. With three different package options, you can compare them side-by-side and choose the package that best fits your specific needs. Please see our Support Packages page for more details (http://www.yassl.com/yaSSL/Support/support_tiers.php).

Chapter 4: Installation

4.1 Web Server Download

The yaSSL Embedded Web Server may be downloaded from the yaSSL Website:

Download Page: http://yassl.com/yaSSL/Download.html

4.2 Standard Installation

To install the yaSSL Embedded Web Server, follow these simple steps:

1. Download the Source from http://www.yassl.com
2. Unzip the downloaded source folder to your desired location
3. Change directory (cd) to the yasslEWS directory
4. Run 'make' and look for your platform
5. Run 'make' again followed by your platform. e.g:

   ```
   $make
   make (linux|bsd|solaris|mac|windows|mingw|ios)
   $make mac
   ```

6. To run with SSL, please make sure the CyaSSL libraries are available. After a make install of CyaSSL, the libraries will be available in /usr/local/lib with a default configuration. If /usr/local/lib is not in your library path, please add it like so:

   Linux:
   ```
   export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cyassl/lib
   ```

   Mac:
   ```
   export DYLD_LIBRARY_PATH=$DYLD_LIBRARY_PATH:/usr/local/cyassl/lib
   ```

7. If desired, create a configuration file to meet your needs. A sample config file is included in the yasslEWS package (yasslEWS.conf).

8. Run ./yasslEWS with the desired options to start the server. Please see the yaSSL Embedded Web Server man page or Section 5 of this manual for a list of available options. An example of starting the server with basic the basic options of setting the SSL certificate, ports, and error file:

```
./yasslEWS -s server.pem -p 8080,8081s -e error.txt
```

Chapter 5: Web Server Options

The yaSSL Embedded Web Server provides users with a number of options for configuring and running the server as desired. This section provides a list of the available web server options. These options may be defined at runtime or in the web server configuration file. If used in the configuration file, you must omit the dashes before the option.

Many of the options listed below are described in more detail with usage examples Chapter 6: Feature Descriptions.

5.1 Web Server Options

Server Option	Description
-A <htpasswd_file> <domain_name> <user_name> <password>	Add to or modify the user list in a given .htpasswd file.
-C cgi_pattern	All files which match the given CGI pattern are treated as CGI, independent of any specific directory location (default: "**.cgi$\|**.pl$|**.php$").
-E cgi_environment	Comma-separated list of custom CGI environment variables in the form of X=Y pairs (ex: VARIABLE=VALUE).
-G put_delete_passwords_file	PUT and DELETE passwords file. This must be specified if PUT and DELETE methods are used.
-I cgi_interpreter	CGI interpreter to use with all CGI scripts.
-P protect_uri	Comma-separated list of URI=htpasswd mapping pairs. .htpasswd files may be modified or created using the "-A" option. The given URI will be protected using the associated htpasswd file.
-R authentication_domain	Authentication domain name (default: "mydomain.com").

-S ssi_pattern	All files which match the given list of accepted SSI patterns are treated as SSI. Unknown SSI directives are silently ignored. Currently supported SSI directives are “include” and “exec”. (default: "**.shtml$\|**.shtm$").
-a access_log_file	Access log file. By default, no logging is done.
-c ssl_chain_file	(default: “ ”)
-d enable_directory_listing	Turn on/off directory listing. Use "yes" or "no", (default: "yes").
-e error_log_file	Error log file. By default, no logging is done.
-g global_passwords_file	Global passwords file. If this is specified, individual directory htpasswd files will be ignored and all requests will be validated against the global password file.
-i index_files	A list of default index files to use. Default: "index.html, index.htm, index.cgi".
-k enable_keep_alive	(default: “no”)
-l access_control_list	Comma-separated list of IP subnets, where each subnet is prepended by either a "-" or a "+" meaning to disallow or allow, respectively. If the subnet mask is omitted, this means to allow/deny only an individual IP address.
-M max_requests_size	(default: “16384”)
-m extra_mime_types	Comma separated list of ext=mime_type pairs
-p listening_ports	Comma-separated list of ports to listen on. If the port is SSL, a letter "s" must be appended to the port number (ex: -ports 8080,8081s). An IP Address may be bound to as well such as "-ports 127.0.0.1:80".
-r document_root	Web root directory. If this option is not specified, the current working directory will be used as the web root. A comma-separated list of URI_PREFIX=DIRECTORY pairs may be appended to it to allow for aliases or let yasslEWS serve from multiple directories.

-s ssl_certificate	SSL certificate file
-t num_threads	Maximum simultaneous threads to spawn (default: "10").
-u run_as_user	Run as the specified user.

Chapter 6: Feature Descriptions

6.1 Server Side Includes Support (SSI)

Server Side Includes (SSI) is a simple interpreted server-side scripting language which is most commonly used to include the contents of a file into a web page. It can be useful when it is desirable to include a common piece of code throughout a website.

Some of the ways in which Server Side Includes may be used include:

- Including the contents of a file (html, txt, etc.) into a web page
- Include the result of running a CGI script
- Executing a program, script, or shell command on the server

In order for a web page to recognize an SSI-enabled HTML file, the filename should end with a special extension. The default for the yaSSL Embedded Web Server is either **shtml** or **shtm**. This can be customized using the SSI pattern runtime option (-S, or ssi_pattern) or by setting the appropriate option in a configuration file.

The yaSSL Embedded Web Server supports two SSI directives, "**include**" and "**exec**". The "include" directive may be used to include the contents of a file or the result of running a CGI script. When using "**include**", you must use either the "**virtual**" or "**file**" variable when specifying the path to the file or script. The "**virtual**" variable specifies the target relative to the web root. The "**file**" variable specifies the target relative to the directory of the current file. An example of including a file called "test.c" is located in a file called “ssi_include.shtml” in the web root:

```
<!--#include file="test.c" -->
```

The second directive supported, "**exec**", is used to execute a program, script, or shell command on the server. An example of executing the "**ls -l**" command:

```
<!--#exec "ls -l" -->
```

Examples can be found in the yaSSL Embedded Web Server download.

For more information on Server Side Includes, take a look at the Wikipedia entry: Server Side Includes (http://en.wikipedia.org/wiki/Server_Side_Includes)

6.2 Access Control List (ACL)

An **Access Control List** (ACL) allows restrictions to be put on the list of IP addresses which have access to the web server. In the case of the yaSSL Embedded Web Server, the ACL is a comma separated list of IP subnets, where each subnet is prepended by either a '-' or a '+' sign. A plus sign means allow, where a minus sign means deny. If a subnet mask is omitted, such as "-1.2.3.4", this means to deny only that single IP address.

Subnet masks may vary from 0 to 32, inclusive. The default setting is to allow all, and on each request the full list is traversed - where the last match wins.

The ACL can be specified either at runtime or in the config file, using the **-l option**, or by the "**access_control_list**" option. For example, to allow only the **192.168.0.0/16** subnet to connect, you would run the following command:

```
./yasslEWS -l -0.0.0.0/0,+192.168.0.0/16
```

The ACL can also be set in the web server **config file**. Using the example above, the config file line would be:

```
access_control_list       -0.0.0.0/0,+192.168.0.0/16
```

To learn more about subnet masks, see the Wikipedia page on Subnetwork (http://en.wikipedia.org/wiki/Subnetwork), or IP Subnet Masks by Transtronics (http://wiki.xtronics.com/index.php/IP_Subnet_Masks).

6.3 CGI Support

Using **CGI** (Common Gateway Interface), a web server can communicate with other types of programs running on the server. Because the yaSSL Embedded Web Server by itself is only able to deal with HTML files, it can "pass off" scripts written in other languages to their specific interpreter, thus allowing the functionality of many CGI languages to be used. Some of the possible languages include: *PHP, Perl, ASP, ASP.NET, Python, Ruby on Rails, and C.*

A. Enabling CGI Support

To configure the yaSSL Embedded Web Server to process CGI scripts in a given language, the interpreter for that language must be installed on the server. As an example, we'll walk through how you would enable PHP to be used with the yaSSL Embedded Web Server.

The first thing you would need to do would be to download PHP if it is not currently installed on your server. The PHP source can be downloaded from the following location: http://www.php.net/downloads.php. After it has been downloaded, it should be built and installed. From the php source directory, run the following commands:

```
./configure
make
sudo make install
```

On OS X, this will place the “**php-cgi**" program in the "**/usr/local/bin**" directory. OS X is just being used as an example here. Steps will be similar for other operating systems. Now that the PHP CGI interpreter is installed, we need to let the yaSSL Embedded Web Server know where it is located. This can be done in two ways (as most options can) and be set at runtime or in the config file by using the **-I** or the **-cgi_interpreter** option. We can also set the **-C** (or **-cgi_pattern**) option, which defines which extensions are treated as CGI scripts. Setting these option at runtime, you would start the yaSSL Embedded Web Server as follows:

```
./yasslEWS -C “**.cgi$|**.php$” -I /usr/local/bin/php-cgi
```

After starting the web server, you can test if PHP is working by browsing to any PHP file which is located under your web server root directory. To quickly test this, browse to the “php_test.php” file located in the root directory of the yaSSL Embedded Web Server download.

B. CGI Environment Variables

Additional CGI environment variables (in addition to the standard ones) may be defined either at runtime or through the configuration file by using the “**-E**” (or **-cgi_environment**) variable. The list must be a comma-separated list of VAR=VALUE pairs. For example, to define two additional environment variables, VAR1 and VAR2, using the configuration file would be similar to the following:

```
cgi_environment VAR1="value1 here",VAR2="value2 here"
```

6.4 Directory Listing

Directory listing can be enabled or disabled in the yaSSL Embedded Web Server either at runtime or in the config file by using the **-d** (or **-enable_directory_listing**) option. By default, directory listing is turned on. To turn off directory listing at runtime, start the yaSSL Embedded Web Server with the above option:

```
./yasslEWS -d no
```

Directory listing may be turned off through the config file using the following line:

```
enable_directory_listing   no
```

If directory listing is turned off and a user attempts to view a directory, they will see a message similar to the following:

```
Error 403: Directory Listing Denied
Directory listing denied
```

6.5 Default Index File List

The user may set the list of default index files for the yaSSL Embedded Web Server using the “**-i**” (or **-index_files**) option at runtime or in the config file. To set the list of default index files at runtime, start the yaSSL Embedded Web Server similar to the following:

```
./yasslEWS -i index.html,index.htm,index.php,index.cgi
```

To set the list from the config file, use a line similar to the one below.

```
index_files           index.html,index.htm,index.php,index.cgi
```

Since it is possible to define a list of default index pages, the yaSSL Embedded Web Server will use the first default file that it finds in the list that exists in the web directory as the index page it shows to the user.

Using the above lists as an example, if there is both an **index.html** and **index.htm** file in the root of the web server, **index.html** will be used because it comes first in the default file list and therefore has a higher priority than index.htm.

6.6 Securing URIs and Directories

The yaSSL Embedded Web Server allows you to secure URIs and directories under your server web root. This is beneficial when you desire to protect specific directories and limit access to specific users.

A. Setting the Authentication Domain

The authentication domain is the area of a website which will be secured. This is the first thing that needs to be done, and can be set using the "**-R**" (or **-authentication_domain**) option at runtime or in the config file. For example, the following sets the authentication domain to "localhost":

Set at runtime:

```
./yasslEWS -R localhost
```

Set in the config file:

```
authentication_domain localhost
```

B. Creating the .htpasswd File

The second thing that needs to be done is that a **.htpasswd** file needs to be created to hold the username and password of the users who will be allowed to enter the restricted URI or directory. To create a .htpasswd file, the **yasslEWS** application may be used with the "**-A**" option. The "**-A**" option allows usernames and passwords to be added or modified within a given .htpasswd file. Deleting users can be done through a normal text editor and functionality is similar to that of Apache's htdigest utility. To create a .htpasswd file in the root directory of the web server with the username "admin" and the password "pass", execute the following command from the web root. For more information on the format of the -A option, please see the yasslEWS man page.

```
 ./yasslEWS -A ./.htpasswd localhost admin pass
```

This will create the corresponding .htpasswd file in the root directory of the web server.

C. Protecting Specific URIs or Directories

Now that an authentication domain has been defined and a .htpasswd file has been created, use the "**-P**" (or **-protect_uri**) option at runtime or in the config file to specify which directories and URIs are to be protected. This option is a comma separated list of **URI=path** pairs specifying that the given URI must be protected with the associated password file.

To protect a directory called "secure_area", the following command would be used:

Set at runtime:

```
./yasslEWS -P /secure_area=./.htpasswd
```

Set in the config file:
```
protect_uri        /secure_area=./.htpasswd
```

D. Setting the Global Passwords File

It’s possible to set a global passwords file in the yaSSL Embedded Web Server. If this is set, all per-directory .htpasswd files will be ignored and all requests will be authorized through this global .htpasswd file. To set this, use the “**-g**” (or **-global_passwords_file**) option at runtime or in the config file:

Set at runtime:

```
./yasslEWS -g ./.htpasswd
```

Set in the config file:

```
global_passwords_file       ./.htpasswd
```

6.7 Additional Features

To see additional features not listed in this manual, please view the yaSSL Embedded Web Server man page.