HP Secure Web Server for OpenVMS (based on Apache) Version 1.3-1 Release Notes

January 2005

Version 1.3-1 for OpenVMS Alpha, based on Apache 1.3.26
CPQ-AXPVMS-CSWS-V0103-1-1.PCSI_SFX_AXPEXE

Version 1.3-1 for OpenVMS I64, based on Apache 1.3.26
HP-I64VMS-CSWS-V0103-1-1.PCSI_SFX_I64EXE

Contents

HP is pleased to provide you with Version 1.3-1 of the HP Secure Web Server for OpenVMS Alpha and OpenVMS I64. The Secure Web Server includes Secure Sockets Layer (SSL) through mod_ssl and OpenSSL. (New kits for OpenVMS Version 8.2 are built with OpenSSL 0.9.7D.)

For the latest information, see the Secure Web Server for OpenVMS web site at http://www.hp.com/products/openvms/securewebserver/.

What's New

The *Secure Web Server* Version 1.3-1 is based on Apache 1.3.26 and contains support for suEXEC and webDAV, enhancements to MOD_AUTH_OPENVMS, and new server configuration features. The Secure Web Server runs on both OpenVMS I64 and OpenVMS Alpha.

If you are upgrading to OpenVMS Alpha Version 8.2, the currently available CSWS_JAVA V2.1, CSWS_PERL V1.1, and Perl for OpenVMS V5.6-1 kits will work properly. New kits are required for the Secure Web Server and CSWS_PHP on OpenVMS Alpha, and for the Secure Web Server and all of the optional kits on OpenVMS I64.

HP Secure Web Server Documentation

See the *HP Secure Web Server for OpenVMS Installation and Configuration Guide* and the *SSL User Guide* for information about installing, configuring, and running the Secure Web Server.

See the individual CSWS_PERL, CSWS_JAVA, and CSWS_PHP *Installation Guide and Release Notes* for information about using those optional kits with the Secure Web Server.

Apache Server Documentation

Refer to the Apache HTTP Server documentation at http://httpd.apache.org/docs/ for information about the Apache server after you have completed the installation.

You can also view the online Apache server documentation on your web site at

```
http://your.domain/manual
```

Note: To view some of the Apache server documentation on your web site, you must enable MultiViews under `<Directory "/apache$common/htdocs">`.

New Features in Version 1.3-1

- Ported to OpenVMS I64

The Secure Web Server has been ported to OpenVMS I64 and is included with OpenVMS I64 and OpenVMS Alpha Version 8.2.

- Latest security fixes included

The Secure Web Server Version 1.3-1 includes all of the latest available bug and security fixes, including fixes to a mod_ssl handshake timeout denial-of-service and a mod_ssl keepalive timeout that causes server process termination. Details about these fixes are available at http://h71000.www7.hp.com/openvms/products/ips/apache/CSWS13_UPDATE-V0600_README.TXT.

New Features in Version 1.3

- Based on Apache 1.3.26 from the Apache Software Foundation.

Previous versions of the Secure Web Server were based on Apache 1.3.20, 1.3.14 and 1.3.12.

- Support for suEXEC

The suEXEC feature provides the ability to run CGI programs under user IDs other than the user ID of the web server.

See the suEXEC Support section in the *HP Secure Web Server for OpenVMS Installation and Configuration Guide* for more information.

- Support for webDAV

webDAV allows clients to perform remote web content authoring operations.

See the MOD_DAV (Distributed Authoring and Versioning) Support section in the *HP Secure Web Server for OpenVMS Installation and Configuration Guide* for information about using MOD_DAV with the Secure Web Server.

See http://www.webdav.org/mod_dav/ for information about webDAV.

- Enhancements to MOD_AUTH_OPENVMS

The MOD_AUTH_OPENVMS module supports authentication using the usernames and passwords contained in the system authorization file (SYS$SYSTEM:SYSUAF.DAT).

See the Authentication Using OpenVMS Usernames and Passwords (MOD_AUTH_OPENVMS) section in the *Secure Web Server for OpenVMS Installation and Configuration Guide* for more information.

- New server configuration features

You now can define server startup, shutdown, and tag configuration elements while configuring a server using APACHE$CONFIG.

See the Configuring a Single Server section and the Process Logical Names table in the *HP Secure Web Server for OpenVMS Installation and Configuration Guide* for more information.

- STATUS function in APACHE$CONFIG.COM

The STATUS function reads the specified configuration data file and displays status information about the running server.

See Specifying the STATUS Function in the *HP Secure Web Server for OpenVMS Installation and Configuration Guide* for more information.

HP Secure Web Server Release Notes

- Edit *.CONF files with new image filenames after upgrade (V1.3-1 only)

Beginning with the Secure Web Server Version 1.3-1, the _ALPHA suffix has been removed from image filename extensions. All images now use the .EXE file extension.

These image filenames are referenced in the *.CONF files read during Apache startup. Because you are allowed to customize the *.CONF files, the installation procedure does not replace the files during an upgrade. **You must manually edit any *.CONF procedure following the V1.3-1 upgrade and change the *.EXE_ALPHA references to *.EXE.** In particular, you will need to modify HTTPD.CONF and MOD_SSL.CONF to reflect the new image names (MOD_AUTH_OPENVMS.EXE, MOD_DAV.EXE, and MOD_SSL.EXE).

- Mixed-architecture cluster support (V1.3-1 only)

This version of the software does not support mixed-architecture clusters. Support for mixed-architecture cluster operation will be included in a future release.

- ODS-5 disk volume required in Version 2.0 and higher

Version 2.0 of the Secure Web Server (available for download from the Secure Web Server web site at http://www.hp.com/products/openvms/securewebserver/) requires an ODS-5 disk volume for installation and operation. This requirement is a result of the increasing use of extended file specifications and filename case sensitivity in Apache and related components. ODS-5 support in the DEC C Run-Time Library will make the Secure Web Server file system name-space more consistent between OpenVMS and UNIX.

- Running MOD_OSUSCRIPT with the Secure Web Server Version 1.1 for OpenVMS

The Secure Web Server provides a CGI script environment. However, it also includes MOD_OSUSCRIPT, an optional module that enables the server to run scripts that were written for the OSU http server's script environment (which is not CGI).

In the Secure Web Server Version 1.1 for OpenVMS and higher, the Apache logical names must be defined **systemwide** (not processwide) in order for MOD_OSUSCRIPT to work properly.

See Running MOD_OSUSCRIPT in the *Installation and Configuration Guide* for more information.

- Location of the Listen 80 directive changed in the Secure Web Server Version 1.1 for OpenVMS

In the Secure Web Server Version 1.1 kit, the location of the Listen 80 directive has been moved from the MOD_SSL.CONF file to the HTTPD.CONF file.

This could cause a problem in your configuration of the Secure Web Server Version 1.1 if you use one configuration file from a previous version and one new V1.1 configuration file.

If neither configuration file includes a Listen 80 directive, requests to HTTP:// will not work because the Secure Web Server is not listening on port 80. If both configuration files include the Listen 80 directive, the Secure Web Server will fail to start, and the log file will contain an entry similar to the following:

```
[Tue Mar 13 17:41:39 2003] [crit] (48)address already in use :
make_sock: could not bind to port 80
```

If you use a configuration file (MOD_SSL.CONF or HTTPD.CONF) from a previous version of the Secure Web Server, the workaround is to manually edit the files so that the Listen 80 directive is included in HTTPD.CONF and not in MOD_SSL.CONF.

- Apply CRTL patch to fix problem with root logical names

On OpenVMS Alpha Version 7.2, a problem in the C Run-Time Library prevents the Secure Web Server from correctly locating index.html files in top-level document roots with concealed logical names. An example of a top-level document root with a concealed logical name is `/web_root/000000` (where *web_root* is defined as *ddcu:[directory.]*).

The C RTL patch kit (ECO) for OpenVMS Version 7.2 (VMS72_ACRTL-V0100) corrects this problem. VMS72_ACRTL-V0100 is available from the HP support website.

- Internet Explorer forces download of script with URL ending in .COM

Microsoft Internet Explorer treats the content for a URL ending in .COM as an executable image and pops up the "File Download" dialogue box. For example, Internet Explorer treats the URL http://hostname/cgi-bin/test-cgi-vms.com as an image and does not display its contents, even though this CGI script is returning "text/plain" content.

To work around this problem, add a semi-colon (;) to the end of the URL or eliminate the .COM file extension entirely.

This problem does not occur with Mozilla/Netscape browsers.

- Access to CGI scripts

The Secure Web Server allows access to a particular CGI script by script name (without the .COM or .EXE extension) or with a fully-specified script name.

If no extension is specified, the Secure Web Server searches for a CGI script in the following order: *script-name*, *script-name.COM*, *script-name.EXE*.

- Using MOD_NEGOTIATION on OpenVMS

The Apache module MOD_NEGOTIATION allows you to specify language variants of HTML files. For example, `filename.html.fr` is the French variant of `filename.html`.

To specify language variants using the Secure Web Server, use an underscore instead of a period before the language tag, as follows:

```
filename.ext_tag
```

For example, use `INDEX.HTML_EN` instead of `INDEX.HTML.EN`.

- Error `%IMGACT-F-SYMVECMIS`

If you receive this error, apply the most current DEC C RTL ECO from the HP support website.

- Large binary data transfer using CGI

In previous versions of the Secure Web Server, a CGI application could not transmit binary data larger than 32K bytes because of a problem with embedded Carriage Return / Line Feed pairs, even if the application attempted to transmit the data in multiple segments containing fewer than 32K bytes.

Note: In Version 1.0-1 and higher, the maximum amount of data you can transfer *in each write operation* is approximately 32K bytes. If your binary file is larger than 32K bytes, transfer the file using multiple write operations of 32K bytes each.

The Secure Web Server supports the transfer of binary files larger than 32K bytes in multiple write operations if the device characteristics of SYS$OUTPUT (or SYS$INPUT) are set correctly, and your CGI application opens SYS$OUTPUT (or SYS$INPUT) using the proper parameters.

There are two ways to properly set the device characteristics of SYS$OUTPUT, as follows:

1. Execute APACHE$FLIP_CCL to set the device characteristics of SYS$OUTPUT to support large binary file transfers.

APACHE$FLIP_CCL disables carriage control on the output device that is required for transferring binary data. APACHE$FLIP_CCL is a symbol that executes APACHE$ROOT:[000000]APACHE$FLIP_CCL.EXE_ALPHA. It accepts an optional argument that specifies the device on which to operate. The default is SYS$OUTPUT.

If your CGI application is run from within a command file, execute APACHE$FLIP_CCL before you execute your CGI application. For example:

```
$ !CGI command file
$ APACHE$FLIP_CCL
$ run MYCGIAPPLICATION
```

If you use DCL commands in your command file to write the HTTP header prior to running your CGI application to transfer binary data, execute APACHE$FLIP_CCL before you write the header. For example:

```
$ !CGI command file
$ APACHE$FLIP_CCL
$ write sys$output f$fao("!AS!/!/","Content-type: image/jpeg")
$ run MYCGIAPPLICATION
```

If you have not written a CGI application to transfer data, Compaq provides APACHE$DCL_BIN as a convenient tool for transferring binary files. (You may be able to use this image instead of writing your own.) The symbol APACHE$DCL_BIN executes the image. The command line syntax for APACHE$DCL_BIN is as follows:

```
$ APACHE$DCL_BIN [-s bin-size] bin-file
```

where *-s bin-size* is an optional parameter that specifies the size in bytes of the binary file, and *bin-file* is the name of the binary file. If you do not specify *bin-size*, APACHE$DCL_BIN computes the size of the binary file.

The following command file is an example of how to transfer binary files using APACHE$DCL_BIN.

```
$ !CGI command file using APACHE$DCL_BIN
$ APACHE$FLIP_CCL
$ write sys$output f$fao("!AS!/!/","Content-type: image/jpeg")
$ APACHE$DCL_BIN myjpegfile.jpg
```

As stated previously, APACHE$FLIP_CCL disables carriage control on the output device that is required for transferring binary data. However, carriage control is required when generating HTTP headers. To generate the headers correctly, you must provide explicit Carriage Return / Line Feed pairs after each header, as follows:

- If you are writing the headers from a command file, use the F$FA0 lexical function as shown in the preceding example.

- If you are writing the headers from your application, the new line escape sequence "\n" in your write statement generates the proper carriage control characters.

Regardless of which way you choose to set the device characteristics, your CGI application must open SYS$OUTPUT (or SYS$INPUT) in binary mode in order to successfully transfer binary files. An example of an fopen() call you can use is as follows:

```
outfile = fopen("SYS$OUTPUT", "wb", "rat=none", "rfm=stm", "ctx=bin");
```

2. Call APACHE$FIXBG() from within your CGI application to set the device characteristics of SYS$OUTPUT (or SYS$INPUT) to support large binary file transfers.

APACHE$FIXBG() is provided in the APACHE$FIXBG.EXE shareable image. The following C code is provided as an example of how to call APACHE$FIXBG().

```
#include <descrip.h>
#include <ssdef.h>
#include <starlet.h>
extern int APACHE$FIXBG(short int, int);
$DESCRIPTOR(output_file,"SYS$OUTPUT");
unsigned short stdout_sock;
int ret_stat;
ret_stat = SYS$ASSIGN(&output_file,&stdout_sock, 0, 0);
if (ret_stat == SS$_NORMAL)
ret_stat = APACHE$FIXBG(stdout_sock,1);
```

Link your CGI application against the APACHE$FIXBG shareable image using the following command in your linker options file:

```
APACHE$FIXBG/share
```

Corrected Problems and Workarounds

- APACHE$CERT_TOOL failed to process Country Name field

In the Secure Web Server Version 1.3 and earlier, the APACHE$CERT_TOOL.COM procedure failed to process the Country Name field during certificate request generation when extended DCL parsing is enabled ($ SET PROCESS/PARSE_STYLE=EXTENDED). This caused the tool to write an invalid countryName field in the OPENSSL.CONF configuration file. The invalid data results in the tool issuing an "Invalid Entry, Try again..." message.

The problem can be avoided by deleting the existing OPENSSL.CONF file and disabling extended DCL parsing ($ SET PROCESS/PARSE_STYLE=TRADITIONAL).

This problem is corrected in the Secure Web Server Version 1.3-1.

- Links to parent directory

Links to the parent directory in directory indexes are now supported.

- File ownership check enforced

In the *Secure Web Server* Version 1.3 and higher, file ownership check is enforced by default for CGI scripts. Use APACHE$CGI_BYPASS_OWNER_CHECK to override this check for compatibility with previous versions of the Secure Web Server.

- RWMBX processes during shutdown

The *Secure Web Server* Version 1.3 fixes a problem with RWMBX processes during shutdown when using a large number of server processes.

- MOD_OSUSCRIPT NET_XXX processes

The Secure Web Server Version 1.3 fixes a problem whereby MOD_OSUSCRIPT NET_XXX processes were not deleted for scripts with "Location" headers.

- Server shutdown on ODS-5 disks

The Secure Web Server Version 1.3 fixes a problem that occurred when attempting to shut down the server on ODS-5 disks.

- APACHE$CONFIG NEW corrupted log file channels

The Secure Web Server Version 1.3 fixes a problem with the APACHE$CONFIG NEW command sometimes corrupting log file I/O channels.

Known Problems and Restrictions

- mod_auth_openvms does not recognize the PWDMIX flag in the SYSUAF.DAT file

The module mod_auth_openvms does not recognize the PWDMIX flag in the SYSUAF.DAT file. A user with an account that has PWDMIX set cannot be authenticated correctly unless the password is all uppercase.

- MOD_DAV record attributes other than STREAM-LF may cause data corruption

Only STREAM-LF files are supported by DAV. If you have an existing file that is not in STREAM-LF format, a GET/PUT will create a new version of the file with the previous version's file attributes. This may cause data corruption.

- AuthOpenVMSAuthoritative directive fails

In Version 1.3 and higher, MOD_AUTH_OPENVMS is always "authoritative," which is the default behavior in previous releases of the Secure Web Server. A problem with the AuthOpenVMSAuthoritative directive prevents it from being used turn off "authoritative" mode. See the Authentication Using OpenVMS Usernames and Passwords (MOD_AUTH_OPENVMS) section in the *HP Secure Web Server for OpenVMS Installation and Configuration Guide* for more information.

- CGI scripts may fail when running TCPware from Process Software

If you are running TCPware from Process Software, CGI scripts may fail with the following entry in the error log file:

```
18-Dec-2004 14:20:50 [000007C0] GENERIC_SOCKETPAIR_inet:
SYS$ASSIGN() -2312 [Thu Dec 18 14:20:50 2004]
[error] [client a.b.c.d] couldn't spawn child process:
/apache$common/cgi-bin/test.cgi
```

To avoid this problem, add the following line to your SYSTARTUP_VMS.COM procedure:

```
$ define/system tcpip$device ucx$device
```

- Problem with PROXY-CACHE file cleanup

The **CacheSize** directive does not work correctly when you select the caching option under MOD_PROXY. The workaround for this problem is to periodically delete the files in the CACHE directory if the size becomes unmanageable.

This problem will be corrected in a future release of the *Secure Web Server*.

- **User** directive not supported

The **User** directive cannot be used to change the user profile of the web server processes in this version of the Secure Web Server. All web server processes run under the APACHE$WWW user profile. HP may add support for the **User** directive in the future.

- **ProxyPass** directive not supported inside a virtual container

The **ProxyPass** directive in mod_proxy is not supported inside a virtual container.

- DCL SEARCH command causes fatal error in CGI scripts

The DCL SEARCH utility is not compatible with CGI scripts when the SEARCH command accesses more than one file. For example:

```
$ SEARCH *.HTML "HP"
```

In this case, the SEARCH utility closes and re-opens SYS$OUTPUT on every file access. In a CGI script, this tears down the output stream and yields the following error:

```
%SEARCH-F-WRITEERR, error writing SYS$OUTPUT:.;
-RMS-F-WER, file write error
-SYSTEM-F-BADPARAM, bad parameter value
```

The workaround for this problem is to output the search results to a temporary file, and then type the contents of the output file. For example:

```
$ pid = f$getjpi("","PID") !Get PID to generate unique name
$ search *.html "HP"/output=SRCH_'pid'.TMP
$ type SRCH_'pid'.TMP
$ delete SRCH_'pid'.TMP;*
```

- DCL extended parse style not supported

The Secure Web Server does not support the DCL extended parse style feature. Extended parse style allows additional DCL syntax for ODS-5 extended file specifications and case-preserves foreign command arguments.

If extended parse style is enabled, the *HP Secure Web Server* startup will fail with the following error:

```
%SYSTEM-F-NOLOGTAB, no logical name table name match
```

You can disable DCL extended parse style for the Secure Web Server by adding the following line to APACHE$ROOT:[000000]LOGIN.COM:

```
$ SET PROCESS/PARSE_STYLE=TRADITIONAL
```

- Limited support for ODS-5 disks

The Secure Web Server has limited support for the OpenVMS Extended File System (EFS, ODS-5). You can install the Secure Web Server on an EFS disk. ODS-2 features and deeply nested directories work properly. However, EFS-specific features such as multiple periods (dots) in file names and extended characters do not work properly.

- Overwritten ERROR_LOG. and ACCESS_LOG. entries

Because several processes write to APACHE$ROOT:[LOGS]ERROR_LOG. and ACCESS_LOG., entries from one process can occur in the middle of a line written from another.

SSL Release Notes

- Delete the temporary, self-signed certificate

After you install a real certificate, delete the temporary (30-day), self-signed certificate (APACHE$SPECIFIC:[CONF.SSL_CRT]SERVER.CRT) that was created during the installation of the Secure Web Server. This prevents the accidental use of the temporary certificate if you installed the real certificate in APACHE$COMMON:[CONF.SSL_CRT] using the same name, and your MOD_SSL.CONF directive uses APACHE$ROOT as part of the certificate file path. For example:

```
  SSLCertificateFile /apache$root/conf/ssl_crt/server.crt
```

Because APACHE$ROOT is a search-listed logical name, the server first searches APACHE$SPECIFIC:[CONF.SSL_CRT] and then searches APACHE$COMMON:[CONF.SSL_CRT] for the SERVER.CRT file. If you used the same name as the temporary certificate file, the server will find the temporary file first.

- Legal caution

SSL data transport requires encryption. Many governments, including the United States, have restrictions on the import and export of cryptographic algorithms. Please ensure that your use of SSL is in compliance with all national and international laws that apply to you.

- Changing MOD_SSL directives

Always make changes to mod_ssl directives in the MOD_SSL.CONF include file. Do not add mod_ssl directives to HTTPD.CONF. You must stop and restart the Secure Web Server for any change you make in MOD_SSL.CONF to take effect.

- <Location> container

The <Location> container statement (which provides for access control by URL) is not supported by mod_ssl directives (although its use in other contexts is permitted in HTTPD.CONF).

- Using the Certificate Tool

Completion of all fields is mandatory when using the OpenSSL Certificate Tool options. Required information that is omitted will prevent certificates from being generated or create invalid certificates.

- Common name usage

When creating server certificate requests, the common name must be the same as your server's DNS host name (or virtual host name, if name-based virtual hosting is used).

- Initializing command-line OpenSSL

Before using command-line OpenSSL, you must run the following command to initialize the environment:

```
$ @APACHE$COMMON:[OPENSSL.COM]OPENSSL_INIT_ENV.COM
```

- Signing client certificates

When signing a client certificate you must use the same pass phrase you used to create your certificate authority.

- PKCS12 export format

In order to serve PKCS12 client certificates correctly to Mozilla/Netscape users, you need to define this file type in the MOD_SSL.CONF file:

```
AddType application/octet-stream .p12
```

When distributing client certificates signed by your own CA, clients must load both the client certificate and your CA certificate in their browser. (The password used when converting the client certificate to PKCS12 format is the same one used by clients to install the certificate.)

- Use semaphore for SSLMutex

When using the **SSLMutex** mod_ssl directive, use the default system semaphore-caching type (SSLMutex sem). Mutex file locking (SSLMutex file) will reduce system performance or, in some cases, result in hung HTTP or HTTPS connections.

- Shared SSLMutex and SSLSessionCache resources among servers are supported

All servers must specify the same **SSLMutex** directive. If different **SSLMutex** directives are used, access to the SSL session cache will not be synchronized properly.

- Use builtin for SSLRandomSeed

When using the **SSLRandomSeed** mod_ssl directive, use the default builtin source (SSLRandomSeed builtin). Using another source will prevent your server from starting.

- Caution changing SSL directives in .HTACCESS files

When modifying your SSL configuration in .HTACCESS files, be aware that making certain changes when using indexing via a secure connection can lead to unexpected results. For example, if you change the **SSLCipherSuite** to *high* within a directory tree viewed via indexing, the system returns an error for a *low* encryption-capable browser. This occurs even when the directory containing the .HTACCESS file is not visible at the current level of the index. The reason this occurs is that during indexing, directories are processed along with any .HTACCESS files. If the .HTACCESS file includes SSL directives that may force a renegotiation of the handshake (for example, **SSLCipherSuite**), errors may occur unexpectedly.
