Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha (*based on Apache*)

Version 1.0-1 Release Notes

January 5, 2001

Based on Apache V1.3.12
CPQ-AXPVMS-CSWS-V0100-1-1.PCSI-DCX-AXPEXE

We're pleased to provide you with an update to the **Compaq-supported, customer release version** of *Compaq Secure Web Server Version 1.0 for OpenVMS Alpha*. The Compaq Secure Web Server includes Secure Sockets Layer (SSL): mod_ssl and OpenSSL/RSA Crypto-C (BSAFE).

Version 1.0-1 includes a new optional, loadable module, MOD_AUTH_OPENVMS, and corrections to several problems. See the Release Notes for more information.

Downloading the Kit

The *Compaq Secure Web Server for OpenVMS Alpha* kit is available for the Alpha platform as a compressed self-extracting file.

To download the kit, please fill out and submit the registration form at http://www.openvms.compaq.com/openvms/products/ips/apache/csws.html.

Apache Server Documentation

Refer to http://www.apache.org/docs for information about the Apache server after you have completed the installation.

Compaq Secure Web Server Release Notes

- New feature: Support for authentication using OpenVMS usernames and passwords

  Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha includes a new module, MOD_AUTH_OPENVMS, that supports authentication using the usernames and passwords contained in the system authorization file (SYS$SYSTEM:SYSUAF.DAT). You can optionally load the new module at startup. The module is located in:

  ```
  APACHE$COMMON:[MODULES]MOD_AUTH_OPENVMS.EXE_ALPHA
  ```

  Note: External authentication is not supported in Version 1.0-1.

  To enable this feature, the server administrator must edit the HTTPD.CONF file to include the following directive:

  ```
  LoadModule auth_openvms_module
/apache$common/modules/mod_auth_openvms.exe_alpha
  ```

This module supports the new directive "AuthUserOpenVMS {On,Off}".

For example, if a directory is protected using an .HTACCESS file, the contents of that file might contain:

```
AuthType Basic

AuthName "OpenVMS authentication"

AuthUserOpenVMS On

require valid-user
```

When a user seeks to open a file in that directory, the user will be prompted for a username and password. That username and password must match entries in the SYSUAF.DAT file. Furthermore, the SYSUAF.DAT entry must allow a network login for that username at the time of the request. If an invalid authentication request occurs, an intrusion record is written.

An authentication request can be rejected for the following reasons:

- Username is blank or invalid.
- Password is blank or invalid. (Note: *Compaq Secure Web Server* does not allow blank or null passwords, even though blank or null passwords are allowed by OpenVMS.)
- Password is valid, but target account is flagged as an intruder. (Account is flagged after 5 consecutive failures in a given time period; see the DCL commands SHOW INTRUSION and DELETE INTRUSION.)
- Target account has a secondary password defined. (HTTP password protocol does not support secondary passwords.)
- Account is marked for external authentication (see AUTHORIZE /FLAGS=EXTAUTH qualifier).
- Password has expired (see AUTHORIZE /FLAGS=PWD_EXPIRED qualifier).
- Account has expired (see AUTHORIZE /EXPIRATION qualifier).
- Account is disabled (see AUTHORIZE /FLAGS=DISUSER qualifier).
- Access restrictions prevent a network access for this time of day (see AUTHORIZE /ACCESS and /PRIMEDAYS qualifiers).

- Problem corrected: Initial configuration data file not created

  The following problem has been corrected in *Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha*.

In Version 1.0 of *Compaq Secure Web Server*, if you did not previously install a beta kit of the *Compaq Secure Web Server* or *Apache Web Server for OpenVMS*, the initial configuration data file required to run the server was not created when you installed the kit.

- Problem corrected: Large binary file transfer using CGI

  The following problem has been corrected in *Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha*.

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

  Note: In Version 1.0-1, 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.

  Version 1.0-1 supports the transfer of binary files larger than 32K bytes in multiple write operations if the following conditions are met:

  - The device characteristics of SYS$OUTPUT are set correctly.
  - Your CGI application opens SYS$OUTPUT 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. 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 in binary mode in order to successfully transfer binary files. An example of an fopen() call you can use is:

```
  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 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
```

- Problem corrected: Inability to use ACLs to access CGI scripts

  The following problem has been corrected in *Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha*.

  In previous versions of *Compaq Secure Web Server* and *Apache for OpenVMS*, the server restricted CGI execution to script files owned by the server process (APACHE$WWW). The server did not honor access control list (ACL) entries that granted read access to APACHE$WWW. Any attempt to run a CGI script not owned by APACHE$WWW resulted in this browser message:

  ```
  "Forbidden

  You don't have permission to access <script-name> on this server."
  ```

  or, if the URL did not contain the file extension, the error was as follows:

  ```
  "Not Found

  The requested URL <script-name>  was not found on this server."
  ```

  Version 1.0-1 processes CGI scripts for which the server has been granted read access, either through UIC-based SOGW protection or an ACL entry. Read access applies not only to the script file, but also to any directory along the path.

  To grant read access via ACL, use the following DCL command:

  ```
  $ set security/acl=(ident=apache$www,access=read) <file-spec>
  ```

- Problem corrected: Extra blank line required after HTTP headers in CGI scripts

The following problem has been corrected in *Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha*.

In previous versions of *Compaq Secure Web Server* and *Apache for OpenVMS*, HTTP headers generated by CGI scripts sometimes required an extra blank line following the HTTP headers in order to be processed correctly. This was most often observed following a "Location" header, such as:

```
$ write sys$output "Location: http:///new.html"
$ write sys$output ""
$ write sys$output ""
```

In Version 1.0-1, the extra line is no longer necessary.

- Problem corrected: Server process falls into HIB state

  The following problem has been corrected in *Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha*.

  In previous versions of *Compaq Secure Web Server*, on heavily loaded systems running CGI scripts, occasionally a server process fell into HIB state and ceased processing.

- Workaround: Server-side include (SSI) "exec cmd"

  When writing server-side include (SSI) `exec cmd` commands, the initial write to SYS$OUTPUT inserts a <NULL><LF> character sequence at the beginning of the data stream.

  The <NULL> character may cause unpredictable behavior in some browsers. In particular, Netscape browsers ignore data between the <NULL> character and the next HTML tag or until the end of the TCP/IP data segment received from the server. The browser either displays no data at all or displays only the tail end of the data.

  There are several workarounds to this problem, but each requires that the target of the `exec cmd` command be a DCL procedure.

  Example SSI script:

  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
  <HTML>
  <HEAD>
  <TITLE>Test server-side include</TITLE>
  </HEAD>
  <BODY>
  <!--#exec cmd="$@apache$Root:[000000]test.com"--> <BR>
  ```

```
</BODY>
</HTML>
```

APACHE$ROOT:[000000]TEST.COM:

```
$! The following example eliminates the <NULL><LF> in the first
$! record sent to sys$output by temporarily turning off CCL and then
$! generating a line of output text which includes the <PRE> tag.
$!
$ apache$flip_ccl
$ write sys$output "<PRE>"
$ apache$flip_ccl
$ show system
$ write sys$output "</PRE>"
$ exit

$! This section includes the <NULL><LF>, but is followed by an
$! HTML tag which allows Netscape to resume HTML processing after
$! the <NULL> character.
$!
$ write sys$output "<PRE>"
$ show system
$ write sys$output "</PRE>"
$ exit

$! This section permanently disables carriage-control and performs its
$! own CR/LF processing by writing output to a file and reading it back
$! format it with HTML tags.
$!
$ apache$flip_ccl
$ tmpfil = "APACHE$EXEC_CMD_" + f$getjpi("","pid") + ".TMP"
$ define/user sys$output 'tmpfil'
$ show system
```

```
$ open x 'tmpfil'
$ read/end=end_it/error=end_it x y
$ write sys$output "<BOLD>"
$ write sys$output f$fao("!AS<BR>",y)
$ write sys$output "</BOLD>"
$ write sys$output "<PRE>"
$loop:
$ read/end=end_it/error=end_it x y
$ write sys$output f$fao("!AS!/",y)
$ goto loop
$! type apache$root:[000000]test.txt
$end_it:
$ close x
$ write sys$output "</PRE>"
$ delete 'tmpfil';*
$ exit
```

- Workaround: Running CGI scripts

  When you use .COM scripts with the POST method, you must read from APACHE$INPUT, not SYS$INPUT. See the file APACHE$ROOT:[CGI-BIN]TEST-CGI-VMS.COM for an example.

- Apply CRTL patch to fix problem with root logical names

  On OpenVMS Alpha Version 7.2, a problem in the Compaq C Run-Time Library prevents the *Compaq 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 Compaq support website.

- Running MOD_OSUSCRIPT with Compaq Secure Web Server

  The *Compaq Secure Web Server for OpenVMS Alpha* 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).

See the *Installation and Configuration Guide* for more information.

- 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 Netscape browsers.

- Access to CGI scripts

  The *Compaq Secure Web Server for OpenVMS* 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 *Compaq 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 *Compaq Secure Web Server for OpenVMS*, 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 Compaq support website.

SSL Release Notes

- Problem corrected: Inability to import client certificates into Netscape® Communicator or Microsoft® Internet Explorer

  The following problem has been corrected in *Compaq Secure Web Server Version 1.0-1 for OpenVMS Alpha*.

  In Version 1.0, client certificates converted from PEM to PKCS12 format with the OpenSSL utility sometimes could not be imported into Netscape and Microsoft web browsers.

Netscape Communicator issued the following error message:

```
  "Unable to import certificates.  The file specified is either corrupt
or is not a valid file."
```

Internet Explorer issued the following error message:

```
  "Invalid password error."
```

Using the OpenSSL utility to analyze the .P12 file resulted in an error message similar to the following:

```
  $ openssl pkcs12 -in apache$root:[openssl.crt]test3.p12

  Enter Import Password:

  MAC verified OK

  Error outputting keys and certificates

  172:error:06065064:digital envelope
routines:EVP_DecryptFinal:baddecrypt:EVP_ENC:248:

  172:error:23077074:PKCS12 routines:PKCS12_pbe_crypt:pkcs12
cipherfinalerror:P12_DECR:95:

  172:error:2306A075:PKCS12 routines:PKCS12_decrypt_d2i:pkcs12 pbe
crypterror:P12_DECR:121:

  $
```

- Restriction on key length for 56-bit encryption web browsers

  If you are using a version of Netscape Communicator or Microsoft Internet Explorer with 56-bit encryption, the key length of the RSA public-key in the client certificate should be no longer than 512 bits.

- 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

- Understand security issues

  Successfully implementing SSL requires more than an SSL-aware web server. Seek expert advice when setting up secure connections with clients.

- 30-day certificate expiration

  After installing *Compaq Secure Web Server*, when you run the configuration utility (APACHE$CONFIG.COM) and enable SSL, it creates and installs a self-signed server certificate. This is good for 30 days only and must be replaced, preferably with a commercial CA certificate. *Compaq Secure Web Server* will not run without a server

certificate that is valid for your system.

- 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 *Compaq 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 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 your system's performance.

- 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.

Known Software Problems

NOTE: This list is a summary and does not contain all known defects in the *Compaq Secure Web Server*.

- Limited support for ODS-5 disks

  Compaq Secure Web Server for OpenVMS has limited support for the OpenVMS Extended File System (EFS, ODS-5). You can install the *Compaq 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.

- Configuration file directive "PassEnv"

  Configuration file directive "PassEnv" does not work. No error messages are generated. The related directives SetEnv and UnsetEnv seem to work, although UnsetEnv works only when specified within a VirtualHost /VirtualHost sequence.

  -- End of file --
