Security Configuration Benchmark For

Apache HTTP Server 2.2

Version 3.0.0
May 18th, 2010

Background.

CIS provides benchmarks, scoring tools, software, data, information, suggestions, ideas, and other services and materials from the CIS website or elsewhere ("**Products**") as a public service to Internet users worldwide. Recommendations contained in the Products ("**Recommendations**") result from a consensus-building process that involves many security experts and are generally generic in nature. The Recommendations are intended to provide helpful information to organizations attempting to evaluate or improve the security of their networks, systems and devices. Proper use of the Recommendations requires careful analysis and adaptation to specific user requirements. The Recommendations are not in any way intended to be a "quick fix" for anyone's information security needs.

No representations, warranties and covenants.

CIS makes no representations, warranties or covenants whatsoever as to (i) the positive or negative effect of the Products or the Recommendations on the operation or the security of any particular network, computer system, network device, software, hardware, or any component of any of the foregoing or (ii) the accuracy, reliability, timeliness or completeness of any Product or Recommendation. CIS is providing the Products and the Recommendations "as is" and "as available" without representations, warranties or covenants of any kind.

User agreements.

By using the Products and/or the Recommendations, I and/or my organization ("**we**") agree and acknowledge that:

No network, system, device, hardware, software or component can be made fully secure;
We are using the Products and the Recommendations solely at our own risk;

We are not compensating CIS to assume any liabilities associated with our use of the Products or the Recommendations, even risks that result from CIS's negligence or failure to perform;

We have the sole responsibility to evaluate the risks and benefits of the Products and Recommendations to us and to adapt the Products and the Recommendations to our particular circumstances and requirements;

Neither CIS, nor any CIS Party (defined below) has any responsibility to make any corrections, updates, upgrades or bug fixes or to notify us if it chooses at it sole option to do so; and

Neither CIS nor any CIS Party has or will have any liability to us whatsoever (whether based in contract, tort, strict liability or otherwise) for any direct, indirect, incidental, consequential, or special damages (including without limitation loss of profits, loss of sales, loss of or damage to reputation, loss of customers, loss of software, data, information or emails, loss of privacy, loss of use of any computer or other equipment, business interruption, wasted management or other staff resources or claims of any kind against us from third parties) arising out of or in any way connected with our use of or our inability to use any of the Products or Recommendations (even if CIS has been advised of the possibility of such damages), including without limitation any liability associated with infringement of intellectual property, defects, bugs, errors, omissions, viruses, worms, backdoors, Trojan horses or other harmful items.

Grant of limited rights.

CIS hereby grants each user the following rights, but only so long as the user complies with all of the terms of these Agreed Terms of Use:

Except to the extent that we may have received additional authorization pursuant to a written agreement with CIS, each user may download, install and use each of the Products on a single computer;

Each user may print one or more copies of any Product or any component of a Product that is in a .txt, .pdf, .doc, .mcw, or .rtf format, provided that all such copies are printed in full and are kept intact, including without limitation the text of this Agreed Terms of Use in its entirety.

Retention of intellectual property rights; limitations on distribution.

The Products are protected by copyright and other intellectual property laws and by international treaties. We acknowledge and agree that we are not acquiring title to any intellectual property rights in the Products and that full title and all ownership rights to the Products will remain the exclusive property of CIS or CIS Parties. CIS reserves all rights not expressly granted to users in the preceding section entitled "Grant of limited rights." Subject to the paragraph entitled "Special Rules" (which includes a waiver, granted to some classes of CIS Members, of certain limitations in this paragraph), and except as we may have otherwise agreed in a written agreement with CIS, we agree that we will not (i) decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for any software Product that is not already in the form of source code; (ii) distribute, redistribute, encumber, sell, rent, lease, lend, sublicense, or otherwise transfer or exploit rights to any Product or any component of a Product; (iii) post any Product or any component of a Product on any website, bulletin board, ftp server, newsgroup, or other similar mechanism or device, without regard to whether such mechanism or device is internal or external, (iv) remove or alter trademark, logo, copyright or other proprietary notices, legends, symbols or labels in any Product or any component of a Product; (v) remove these Agreed Terms of Use from, or alter these Agreed Terms of Use as they appear in, any Product or any component of a Product; (vi) use any Product or any component of a Product with any derivative works based directly on a Product or any component of a Product; (vii) use any Product or any component of a Product with other products or applications that are directly and specifically dependent on such Product or any component for any part of their functionality, or (viii) represent or claim a particular level of compliance with a CIS Benchmark, scoring tool or other Product. We will not facilitate or otherwise aid other individuals or entities in any of the activities listed in this paragraph.

We hereby agree to indemnify, defend and hold CIS and all of its officers, directors, members, contributors, employees, authors, developers, agents, affiliates, licensors, information and service providers, software suppliers, hardware suppliers, and all other persons who aided CIS in the creation, development or maintenance of the Products or Recommendations ("**CIS Parties**") harmless from and against any and all liability, losses, costs and expenses (including attorneys' fees and court costs) incurred by CIS or any CIS Party in connection with any claim arising out of any violation by us of the preceding paragraph, including without limitation CIS's right, at our expense, to assume the exclusive defense and control of any matter subject to this indemnification, and in such case, we agree to cooperate with CIS in its defense of such claim. We further agree that all CIS Parties are third-party beneficiaries of our undertakings in these Agreed Terms of Use.

Special rules.

CIS has created and will from time to time create special rules for its members and for other persons and organizations with which CIS has a written contractual relationship. Those special rules will override and supersede these Agreed Terms of Use with respect to the users who are covered by the special rules. CIS hereby grants each CIS Security Consulting or Software Vendor Member and each CIS Organizational User Member, but only so long as such Member remains in good standing with CIS and complies with all of the terms of these Agreed Terms of Use, the right to distribute the Products and Recommendations within such Member's own organization, whether by manual or electronic means. Each such Member acknowledges and agrees that the foregoing grant is subject to the terms of such Member's membership arrangement with CIS and may, therefore, be modified or terminated by CIS at any time.

Choice of law; jurisdiction; venue.

We acknowledge and agree that these Agreed Terms of Use will be governed by and construed in accordance with the laws of the State of Maryland, that any action at law or in equity arising out of or relating to these Agreed Terms of Use shall be filed only in the courts located in the State of Maryland, that we hereby consent and submit to the personal jurisdiction of such courts for the purposes of litigating any such action. If any of these Agreed Terms of Use shall be determined to be unlawful, void, or for any reason unenforceable, then such terms shall be deemed severable and shall not affect the validity and enforceability of any remaining provisions. We acknowledge and agree that we have read these Agreed Terms of Use in their entirety, understand them and agree to be bound by them in all respects.

Table of Contents

Overview

This document, *Security Configuration Benchmark for Apache HTTP Server 2.2*, provides prescriptive guidance for establishing a secure configuration posture for the Apache HTTP Server versions 2.2.x running on Linux. This guide was tested against Apache Web Server 2.2.14 as built from source `httpd-2.2.14.tar.gz` from http://httpd.apache.org/ on Red Hat Enterprise Linux Server release 5.4. To obtain the latest version of this guide, please visit http://cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@cisecurity.org.

Consensus Guidance

This guide was created using a consensus review process comprised of volunteer and contract subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS benchmark undergoes two phases of consensus review. The first phase occurs during initial benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the benchmark. This discussion occurs until consensus has been reached on benchmark recommendations. The second phase begins after the benchmark has been released to the public Internet. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the CIS benchmark. If you are interested in participating in the consensus review process, please send us a note to feedback@cisecurity.org.

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel, who plan to develop, deploy, assess, or secure solutions that incorporate Apache Web Server on a Linux platform.

Acknowledgements

Typographic Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Configuration Levels

This section defines the configuration levels that are associated with each benchmark recommendation. Configuration levels represent increasing levels of security assurance.

Level-I Benchmark settings/actions

Level-I Benchmark recommendations are intended to:
- be practical and prudent;
- provide a clear security benefit; and
- do not negatively inhibit the utility of the technology beyond acceptable means

Level-II Benchmark settings/actions

Level-II Benchmark recommendations exhibit one or more of the following characteristics:
- are intended for environments or use cases where security is paramount
- acts as defense in depth measure
- may negatively inhibit the utility or performance of the technology

Scoring Status

This section defines the scoring statuses used within this document. The scoring status indicates whether compliance with the given recommendation is discernable in an automated manner.

Scorable

The platform's compliance with the given recommendation can be determined via automated means.

Not Scorable

The platform's compliance with the given recommendation cannot be determined via automated means.

1. Recommendations

1.1 Planning and Installation

Recommendations for the planning and installation of an Apache Web Server

1.1.1 Pre-installation Planning Checklist (Level 1, Not Scorable)

Review and implement the following items as appropriate:

- ✓ Reviewed and implemented my company's security policies as they relate to web security.
- ✓ Implemented a secure network infrastructure by controlling access to/from your web server by using firewalls, routers and switches.
- ✓ Harden the Underlying Operating System of the web server, by minimizing listening network services, applying proper patches and hardening the configurations as recommended in the appropriate Center for Internet Security benchmark for the platform.
- ✓ Implement central log monitoring processes.
- ✓ Implemented a disk space monitoring process and log rotation mechanism.
- ✓ Educated developers about developing secure applications. http://www.owasp.org/ http://www.webappsec.org/
- ✓ Ensure the WHOIS Domain information registered for our web presence does not reveal sensitive personnel information, which may be leveraged for Social Engineering (Individual POC Names), War Dialing (Phone Numbers) and Brute Force Attacks (Email addresses matching actual system usernames).
- ✓ Ensure your Domain Name Service (DNS) servers have been properly secured to prevent attacks, as recommended in the CIS BIND DNS benchmark.
- ✓ Implemented a Network Intrusion Detection System to monitor attacks against the web server.

1.1.2 Do not Install on a Multi-use System (Level 2, Not Scorable)

Description:
Default server configurations often expose a wide variety of services unnecessarily increasing the risk to the system. Just because a server can perform many services doesn't mean it is wise to do so. The number of services and daemons executing on the Apache Web server should be limited to those necessary, with the Web server being the only primary function of the server.

Rationale:
Maintaining a server for a single purpose increases the security of your application and system. The more services which are exposed to an attacker, the more potential vectors an attacker has to exploit the system and therefore the higher the risk for the server. A Web server should function as only a web server and if possible should not be mixed with other primary functions such as mail, DNS, database or middleware.

Remediation:

Leverage the package or services manager for your OS to uninstall or disable unneeded services. On Red Hat systems, the following will disable a given service:

```
chkconfig <servicename> off
```

Audit:
Leverage the package or services manager for your OS to list enabled services and review with document business needs of the server. On Red Hat systems, the following will produce the list of current services enabled:

```
chkconfig --list | grep ':on'
```

Default Value:
Depends on OS Platform

1.1.3 Installing Apache (Level 1, Not Scorable)

Description:

The CIS Apache Benchmark recommends using the Apache binary provided by your vendor for most situations in order to reduce the effort and increase the effectiveness of maintenance and security patches.   However to keep the benchmark as generic and applicable to all Unix/Linux platforms as possible, a default source build has been used for this benchmark.

Important Note: There is a major difference between source builds and most vendor packages that is very important to highlight. The default source build of Apache is fairly conservative and minimalist in the modules included and is therefore starts off in a fairly strong security state, while most vendor binaries are typically very well loaded with most of the functionality that one may be looking for. ***Therefore it is important that you don't assume the default value shown in the benchmark will match default values in your installation.***

You should always test any new installation in your environment before putting it into production. Also keep in mind you can install and run a new version alongside the old one by using a different Apache prefix and a different IP address or port number in the Listen directive.

Rationale:

The benefits of using the vendor supplied binaries include:

- Ease of installation as it will just work, straight out of the box.

- It is customized for your OS environment.

- It will be tested and have gone though QA procedures.

- Everything you need is likely to be included, probably including some third party modules. Many OS vendors ship Apache with `mod_ssl` and OpenSSL and PHP, `mod_perl` and `mod_security` for example.

- Your vendor will tell you about security issues so you have to look in fewer places.

- Updates to fix security issues will be easy to apply. The vendor will have already verified the problem, checked the signature on the Apache download, worked out the impact and so on.

- You may be able to get the updates automatically, reducing the window of risk.

Remediation:
Installation depends on the operating system platform. For a source build consult the Apache 2.2 documentation on compiling and installing
http://httpd.apache.org/docs/2.2/install.html for a Red Hat Enterprise Linux 5 the following yum command could be used.

```
# yum install httpd
```

Audit:
Not Applicable

Default Value:
Not Applicable

References:

1. Apache Compiling and Installation http://httpd.apache.org/docs/2.2/install.html

1.2   Minimize Apache Modules

1.2.1 Enable only necessary Authentication and Authorization Modules (Level 1, Scorable)

Description:
The Apache 2.2 modules for authentication and authorization have been refactored to provide finer granularity, more consistent and logical names and to simplify configuration. The `authn_*` modules provide authentication, while the `authz_*` modules provide authorization. Apache provides 2 types of authentication basic and digest. Review Apache AAA how-to documentation http://httpd.apache.org/docs/2.2/howto/auth.html and enable only the modules that are required.

Rationale:
Authentication and authorization are your front doors to the protected information in your web site. Most installations only need a small subset of the modules available. By minimizing the enabled modules to only those that are required, we reduce the number of

"doors" and have therefore reduced the attack surface of the web site. Likewise having fewer modules means less software that could have vulnerabilities.

Remediation:
Consult Apache module documentation for descriptions of each module in order to determine the necessary modules for the specific installation. http://httpd.apache.org/docs/2.2/mod/ The unnecessary static compiled modules are disabled through compile time configuration options as documented in http://httpd.apache.org/docs/2.2/programs/configure.html. The dynamically loaded modules are disabled by commenting out or removing the LoadModule directive from the Apache configuration files (typically httpd.conf). Some modules may be separate packages, and may be removed

Audit:

1. Use the `httpd -M` option as root to check which `auth*` modules are loaded.

```
# httpd -M | egrep 'auth._'
```

2. Also use the `httpd -M` option as root to check for any LDAP modules which don't follow the same naming convention.

```
# httpd -M | egrep 'ldap'
```

The above commands should generate a "Syntax OK' message to `stderr`, in addition to a list modules installed to `stdout`. If the "Syntax OK" message is missing then there was most likely an error in parsing the configuration files.

Default Value:
The following are the modules statically loaded for a default source build:

```
authn_file_module (static)
authn_default_module (static)
authz_host_module (static)
authz_groupfile_module (static)
authz_user_module (static)
authz_default_module (static)
auth_basic_module (static)
```

References:

1. Apache AAA how-to http://httpd.apache.org/docs/2.2/howto/auth.html

2. Apache Module Documentation http://httpd.apache.org/docs/2.2/mod/

3. Apache Source Configuration http://httpd.apache.org/docs/2.2/programs/configure.html

1.2.2 Enable the Log Config Module (Level 1, Scorable)

Description:

The `log_config` module provides for flexible logging of client requests, and provides for the configuration of the information in each log.

Rationale:
Logging is critical for monitoring usage and potential abuse of your web server. To configure the web server logging using the `log_format` directive this module is required.

Remediation:
Perform either one of the following:

a) For source builds with static modules run the Apache `./configure` script **without** including the `--disable-log-config` script options.

```
$ cd $DOWNLOAD/httpd-2.2.14
$ ./configure
```

b) For dynamically loaded modules, add or modify the `LoadModule` directive so that it is present in the apache configuration as below and not commented out :

```
LoadModule log_config_module modules/mod_log_config.so
```

Audit:
Perform the following to determine if the `log_config` has been loaded:

1. Use the `httpd -M` option as root to check the module is loaded.

```
# httpd -M | grep log_config
```

Note: If the module is correctly enabled, the output will include the module name and whether it is loaded statically or as a shared module.

Default Value:
The module is loaded by default.

References:

1. Mod Log Config http://httpd.apache.org/docs/2.2/mod/mod_log_config.html

1.2.3 Disable WebDAV modules (Level 1, Scorable)

Description:
The Apache `mod_dav` and `mod_dav_fs` modules support WebDAV ('Web-based Distributed Authoring and Versioning') functionality for Apache. WebDAV is an extension to the HTTP protocol which allows clients to create, move, and delete files and resources on the web server.

Rationale:
WebDAV is not widely used, and has serious security concerns as it may allow clients to modify unauthorized files on the web server. Therefore, the WebDav modules `mod_dav` and `mod_dav_fs` should be disabled.

Remediation:
Perform either one of the following to disable WebDAV module:

a) For source builds with static modules run the Apache `./configure` script without including the `mod_dav`, and `mod_dav_fs` in the `--enable-modules=` configure script options.

```
$ cd $DOWNLOAD/httpd-2.2.14
$ ./configure
```

b) For dynamically loaded modules, comment out or remove the `LoadModule` directive for the `mod_dav` and `mod_dav_fs` modules in the `httpd.conf` file.

```
##LoadModule dav_module modules/mod_dav.so
##LoadModule dav_fs_module modules/mod_dav_fs.so
```

Audit:
Perform the following to determine if the WebDAV modules are enabled.

1. Run the httpd server with the `-M` option to list enabled modules:

```
# httpd -M | grep ' dav_[[:print:]]+module'
```

Note: If the WebDav modules are correctly disabled, the only output should be "`Syntax OK`" when executing the above command.

Default Value:
The WebDav modules are not enabled with a default source build.

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_dav.html

1.2.4 Disable Status and Info modules (Level 1, Scorable)

Description:

The Apache `mod_info` module provides information on the server configuration via access to a `/server-info` URL location, while the `mod_status` module provides current server performance statistics.

Rationale:
While having server configuration and status information available as a web page may be convenient, it's recommended that these modules NOT be enabled:

- Once `mod_info` is loaded into the server, its handler capability is available in per-directory `.htaccess` files and can leak sensitive information from the configuration directives of other Apache modules such as system paths, usernames/passwords, database names, etc.
- If mod_`status` is loaded into the server, its handler capability is available in all configuration files, including per-directory files (e.g., `.htaccess`) and may have security-related ramifications.

Remediation:
Perform either one of the following to disable the `mod_info` and `mod_status` modules:

a) For source builds with static modules run the Apache `./configure` script without including the `mod_info`, and `mod_status` in the `--enable-modules= configure` script options.

```
$ cd $DOWNLOAD/httpd-2.2.14
$ ./configure
```

b) For dynamically loaded modules comment out or remove the LoadModule directive for `mod_info`, and `mod_status` modules the from the `httpd.conf` file.

```
##LoadModule info_module modules/mod_info.so
##LoadModule status_module modules/mod_status.so
```

Audit:
Perform the following to determine if the WebDAV modules are enabled.

1. Run the httpd server with the `-M` option to list enabled modules:

```
# /usr/sbin/httpd -M | egrep 'info_module|status_module'
```

Note: If the modules are correctly disabled, the only output should be "`Syntax OK`" when executing the above command.

Default Value:
The `mod_info` and `mod_status` modules are not enabled with a default source build.

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_info.html

2. http://httpd.apache.org/docs/2.2/mod/mod_status.html

1.2.5 Disable Autoindex module (Level 1, Scorable)

Description:
The Apache `autoindex` module automatically generates a web page that lists the contents of the requested directory.

Rationale:
Automated directory listings should not be enabled as it will reveal information helpful to an attacker such as naming conventions and directory paths. It may also reveal files that were not intended to be revealed.

Remediation:
Perform either one of the following to disable the `mod_autoindex` module:

a) For source builds with static modules run the Apache `./configure` script without including the `mod_autoindex` in the `--enable-modules= configure` script options.

```
$ cd $DOWNLOAD/httpd-2.2.14
$ ./configure
```

b) For dynamically loaded modules comment out or remove the `LoadModule` directive for `mod_autoindex` module the from the httpd.conf file.

```
## LoadModule autoindex_module modules/mod_autoindex.so
```

Audit:
Perform the following to determine if the module is enabled.

1. Run the httpd server with the `-M` option to list enabled modules:

```
# /usr/sbin/httpd -M | grep autoindex_module
```

Note: If the module is correctly disabled, the only output should be "`Syntax OK`" when executing the above command.

Default Value:
The `mod_autoindex` module IS enabled with a default source build.

References:

1. AutoIndex http://httpd.apache.org/docs/2.2/mod/mod_autoindex.html

1.2.6 Disable Proxy Modules (Level 1, Scorable)

Description:

The Apache proxy modules allow the server to act as a proxy (either forward or reverse proxy) of http and other protocols with additional proxy modules loaded. If the Apache installation is not intended to proxy requests to or from another network then the proxy module should not be loaded.

Rationale:
Proxy servers can act as an important security control when properly configured, however a secure proxy server is not within the scope of this benchmark. A web server should be primarily a web server or a proxy server but not both, for the same reasons that other

multi-use servers are not recommended.   Scanning for web servers that will also proxy requests is a very common attack, as proxy servers are useful for anonymizing attacks on other servers, or possibly proxying requests into an otherwise protected network.

Remediation:
Perform either one of the following to disable the proxy module:

a) For source builds with static modules run the Apache `./configure` script without including the `mod_proxy` in the `--enable-modules=` `configure` script options.

```
$ cd $DOWNLOAD/httpd-2.2.14
$ ./configure
```

b) For dynamically loaded modules comment out or remove the `LoadModule` directive for `mod_proxy` module and all other proxy modules the from the httpd.conf file.

```
##LoadModule proxy_module modules/mod_proxy.so
##LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
##LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
##LoadModule proxy_http_module modules/mod_proxy_http.so
##LoadModule proxy_connect_module modules/mod_proxy_connect.so
##LoadModule proxy_connect_module modules/mod_proxy_ajp.so
```

Audit:
Perform the following to determine if the modules are enabled.

1. Run the httpd server with the `-M` option to list enabled modules:

```
# httpd -M | grep proxy_
```

Note: If the modules are correctly disabled, the only output should be "`Syntax OK`" when executing the above command.

Default Value:
The `mod_proxy` module and other proxy modules are NOT enabled with a default source build.

References:

1. Mod_Proxy http://httpd.apache.org/docs/2.2/mod/mod_proxy.html

1.2.7 Disable User Directories Modules (Level 1, Scorable)

Description:
The `UserDir` directive must be disabled so that user home directories are not accessed via the web site with a tilde (~) preceding the username. The directive also sets the path name of the directory that will be accessed. For example:

- http://example.com/~ralph/ might access a `public_html` sub-directory of `ralph` user's home directory.
- The directive `UserDir ./` might map `/~root` to the root directory `/`, which of course is undesirable.

Rationale:
The user directories should not be globally enabled since it allows anonymous access to anything users may want to share with other users on the network. Also consider that every time a new account is created on the system, there is potentially new content available via the web site.

Remediation:
Perform either one of the following to disable the proxy module:

1. For source builds with static modules run the Apache `./configure` script with the `--disable-userdir configure` script options.

```
$ cd $DOWNLOAD/httpd-2.2.14
$ ./configure --disable-userdir
```

2. For dynamically loaded modules, comment out or remove the `LoadModule` directive for the `mod_userdir` module in the `httpd.conf` file.

```
##LoadModule userdir_module modules/mod_userdir.so
```

Audit:
Perform the following to determine if the modules are enabled.

1. Run the httpd server with the `-M` option to list enabled modules:

```
# httpd -M | grep userdir_
```

Note: If the modules are correctly disabled, the only output should be "`Syntax OK`" when executing the above command.

Default Value:
The `mod_userdir` modules are enabled with a default source build.

References:

1. Mod_UserDir http://httpd.apache.org/docs/2.2/mod/mod_userdir.html

1.3 Restricting OS Privileges

Security at the operating system (OS) level is the vital foundation required for a secure web server. This section will focus on OS platform permissions and privileges.

1.3.1 Run the Apache Web Server as a non-root user (Level 1, Scorable)

Description:

Although Apache typically is started with root privileges in order to listen on port 80 and 443, it can and should run as another non-root user in order to perform the web services. The Apache User and Group directives are used to designate the user and group to be used

Rationale:
One of the best ways to reduce your exposure to attack when running a web server is to create a unique, unprivileged userid and group for the web daemon to execute as. The "nobody" or "daemon" userid & group that come default on Unix variants should NOT be used to run the web server as these principals are commonly used by other daemon services. Instead, create a user and group that are exclusively used by the web service so as to not give unnecessary access to other services. Also, the userid used for the apache user should be a unique value between 1 and 499 as these lower values are reserved for the special system accounts.

A more secure alternative is to bind Apache web service to an unprivileged port so it is not necessary to start Apache as root.

Remediation:
Perform the following:

1. If the Apache user and group do not already exist, create the account and group as a unique system account:

```
# groupadd –r apache
# useradd apache –r -g apache -d /var/www -s /sbin/nologin
```

2. Configure the Apache user and group in the Apache configuration file `httpd.conf`:

```
User apache
Group apache
```

Audit:
Ensure the apache account is unique and has been created with a UID between1-499 with the apache group and configured in the `httpd.conf` file.

1. Ensure the previous lines are present in the Apache configuration and not commented out:

```
# grep –i '^User' $APACHE_PREFIX/conf/httpd.conf
# grep –i '^Group' $APACHE_PREFIX/conf/httpd.conf
```

2. Ensure the apache account is correct:

```
# id apache
```

The UID must be between 1-499, and group of apache similar to the following entries:

```
uid=48(apache) gid=48(apache) groups=48(apache)
```

Default Value:
The default Apache User and Group is configured as 'daemon':

1.3.2 Give the Apache User Account an Invalid Shell (Level 1, Scorable)

Description:
The apache account must not be used as a regular login account, and should be assigned an invalid or `nologin` shell to ensure that the account cannot be used to login.

Rationale:
Service accounts such as the apache account represent a risk if they can be used to get a login shell to the system.

Remediation:

Change the apache account to use the `nologin` shell or an invalid shell such as `/dev/null`:

```
# chsh –s /sbin/nologin apache
```

Audit:

Check the apache login shell in the `/etc/password` file:

```
# grep apache /etc/passwd
```

The apache account shell must be `/sbin/nologin` or `/dev/null` similar to the following:

```
/etc/passwd:apache:x:48:48:Apache:/var/www:/sbin/nologin
```

Default Value:
The default Apache user account is daemon with a shell of `/dev/null`

1.3.3 Lock the Apache User Account (Level 1, Scorable)

Description:

The user account under which Apache runs, should not have a valid password, but should be locked.

Rationale:
As a defense-in-depth measure the Apache user account should be locked to prevent logins, and to prevent a user from su-ing to apache using the password. In general, there shouldn't

be a need for anyone to su as apache. If a need does exist, sudo should be used instead, which would not require the apache account password.

Remediation:
Use the passwd command to lock the apache account:

```
# passwd –l apache
```

Audit:
Ensure the apache account is locked using the following:

```
# passwd –s apache
```

The results will be similar to the following:

```
apache LK 2010-01-28 0 99999 7 -1 (Password locked.)
```

Default Value:
The default user is daemon and is locked.

1.3.4 Apache Directory and File Ownership (Level 1, Scorable)

Description:
The Apache directories and files should be owned by root with the root (or root equivalent) group.   This applies to all of the Apache software directories and files installed. The only expected exception is that the Apache web document root (`$APACHE_PREFIX/htdocs`) are likely to need a designated group to allow web content to be updated (such as webupdate) through a change management process.

Rationale:
Setting the appropriate ownership and group on the Apache files and directories can help to prevent/mitigate exploitation severity and information disclosure. These changes should also be rechecked to identify any insecure settings on a continued basis through a cron job.

Remediation:
Perform the following:

1. Set ownership on the `$APACHE_PREFIX` directories such as `/usr/local/apache2`:

```
$ chown –R root:root $APACHE_PREFIX
```

Audit:

1. Identify files in the Apache directory not owned by root :

```
# find $APACHE_PREFIX \! –user root –ls
```

2. Identify files in the Apache directory with a group different from root :

```
# find $APACHE_PREFIX \! -group root -ls
```

Default Value:
Default ownership and group is a mixture of the user:group that built the software and
`root:root`.

1.3.5 Apache Directory and File Permissions (Level 1, Scorable)

Description:
The permission on the Apache directories should be `rwxr-xr-x` (`755`) and the file
permissions should be similar except not executable if executable is not appropriate. This
applies to all of the Apache software directories and files installed with the possible
exception in some cases may have a designated group with write access for the Apache web
document root (`$APACHE_PREFIX/htdocs`) are likely to need a designated group to allow
web content to be updated. In addition the `/bin` directory and executables should be set to
not be readable by other.

Rationale:
Setting the appropriate permissions on the Apache files and directories can help to
prevent/mitigate exploitation severity and information disclosure. These changes should
also be rechecked to identify any insecure settings on a continued basis through a cron job.
Also preventing reading of the Apache executables by "other" prevents non-root users from
making copies of the executables which could then be modified.

Remediation:
Perform the following to set the permissions on the `$APACHE_PREFIX` directories and then
remove other read permissions on the bin directory and its contents:

```
# chmod –R u=rwX,g=rX,o=rX $APACHE_PREFIX
# chmod –R u=rwX,g=rX,o=X $APACHE_PREFIX/bin
```

Audit:
Identify files or directories in the Apache directory with other write access, excluding
symbolic links:

```
# find –L $APACHE_PREFIX \! -type l \! -type s -perm /o=w -ls
```

Default Value:
The default permissions are mostly `rwXr-Xr-X` except for some files which have group or
other permissions which seem affected by the umask of the user performing the build.

1.3.6 Core Dump Directory Security (Level 1, Scorable)

Description:
The `CoreDumpDirectory` directive can be used to specify a directory which Apache
attempts to switch before dumping core for debugging. The default directory is the Apache
`ServerRoot` directory, however on Linux systems core dumps will be disabled by default.

Most production environments should leave core dumps disabled. In the event that core dumps are needed, the directory needs to be a writable directory by Apache, and should meet the security requirements defined below in the remediation and audit.

Rationale:
Core dumps are snapshots of memory and may contain sensitive information that should not be accessible by other accounts on the system.

Remediation:
Either remove the `CoreDumpDirectory` directive from the Apache configuration files or ensure that the configured directory meets the following requirements.

1. `CoreDumpDirectory` is not be within the Apache web document root (`$APACHE_PREFIX/htdocs`)

2. must be owned by root and have a group ownership of the Apache group (as defined via the Group directive)

   ```
   # chown root:apache /var/log/httpd
   ```

3. must have no read-write-search access permission for other users.

   ```
   # chmod o-rwx /var/log/httpd
   ```

Audit:
Verify that either the `CoreDumpDirectory` directive is not enabled in any of the Apache configuration files or that the configured directory meets the following requirements:

1. `CoreDumpDirectory` is not be within the Apache web document root (`$APACHE_PREFIX/htdocs`)

2. must be owned by root and have a group ownership of the Apache group (as defined via the `Group` directive)

3. must have no read-write-search access permission for other users. (e.g. `o=rwx`)

Default Value:
The default core dump directory is the `ServerRoot` directory, which is should not be writable. Core dumps will be disabled if the directory is not writable by the Apache user. Also on Linux systems core dumps will be disabled if the server is started as root and switches to a non-root user, as is typical

References:

1. Apache `CoreDumpDirectory` directive
   http://httpd.apache.org/docs/2.2/mod/mpm_common.html#coredumpdirectory

1.3.7 Lock File Security (Level 1, Scorable)

Description:

The `LockFile` directive sets the path to the lock file used when Apache uses `fcntl(2)` or `flock(2)` system calls to implement a mutex. Most Linux systems will default to using semaphores instead, so the directive may not apply. However in the event a lock file is used, it is important for the lock file to be in a locally directory that is not writable by other users.

Rationale:

If the `LockFile` is placed in a writable directory other accounts could create a denial of service attack and prevent the server from starting by creating a lock file with the same name.

Remediation:

1. Find the directory in which the `LockFile` would be created. The default value is the `ServerRoot/logs` directory.
2. Modify the directory if the `LockFile` if it is a directory within the Apache `DocumentRoot`
3. Change the ownership and group to be `root:root`, if not already.
4. Change the permissions so that the directory is only writable by root, or the user under which apache initially starts up (default is root),
5. Check that the lock file directory is on a locally mounted hard drive rather than an NFS mounted file system.

Audit:

1. Find the directory in which the `LockFile` would be created. The default value is the `ServerRoot/logs` directory.
2. Verify that the lock file directory is not a directory within the Apache `DocumentRoot`
3. Verify that the ownership and group of the directory is `root:root` (or the user under which apache initially starts up if not root).
4. Change the permissions so that the directory is only writable by root (or the startup user if not root),
5. Check that the lock file directory is on a locally mounted hard drive rather than an NFS mounted file system.

Default Value:

The default lock file is `logs/accept.lock`

References:

1. Apache `LockFile` directive
   http://httpd.apache.org/docs/2.2/mod/mpm_common.html#lockfile

1.3.8 Pid File Security (Level 1, Scorable)

Description:

The `PidFile` directive sets the file path to the process ID file to which the server records the process id of the server, which is useful for sending a signal to the server process or for checking on the health of the process.

Rationale:
If the `PidFile` is placed in a writable directory, other accounts could create a denial of service attack and prevent the server from starting by creating a PID file with the same name.

Remediation:
1. Find the directory in which the `PidFile` would be created. The default value is the `ServerRoot/logs` directory.
2. Modify the directory if the `PidFile` is in a directory within the Apache `DocumentRoot`
3. Change the ownership and group to be `root:root`, if not already.
4. Change the permissions so that the directory is only writable by root, or the user under which apache initially starts up (default is root),

Audit:
1. Find the directory in which the `PidFile` would be created. The default value is the `ServerRoot/logs` directory.
2. Verify that the process ID file directory is not a directory within the Apache `DocumentRoot`
3. Verify that the ownership and group of the directory is `root:root` (or the user under which apache initially starts up if not root).
4. Change the permissions so that the directory is only writable by root (or the startup user if not root).

Default Value:
The default process ID file is `logs/httpd.pid`

References:

1. Apache `PidFile` directive
   http://httpd.apache.org/docs/2.2/mod/mpm_common.html#pidfile

1.3.9 *ScoreBoard File Security (Level 1, Scorable)*

Description:
The `ScoreBoardFile` directive sets a file path which the server will use for Inter-Process Communication (IPC) among the Apache processes. On most Linux platform shared memory will be used instead of a file in the file system, so this directive is not generally needed and does need to be specified. However, if the directive is specified, then Apache will use the configured file for the inter-process communication. Therefore if it is specified it needs to be located in a secure directory.

Rationale:
If the `ScoreBoardFile` is placed in a writable directory, other accounts could create a denial of service attack and prevent the server from starting by creating a file with the same

name, and or users could monitor and disrupt the communication between the processes by reading and writing to the file.

Remediation:

1. Check to see if the `ScoreBoardFile` is specified in any of the Apache configuration files. If it is not present no changes are required.
2. If the directive is present, find the directory in which the `ScoreBoardFile` would be created. The default value is the `ServerRoot/logs` directory.
3. Modify the directory if the `ScoreBoardFile` is in a directory within the Apache `DocumentRoot`
4. Change the ownership and group to be `root:root`, if not already.
5. Change the permissions so that the directory is only writable by root, or the user under which apache initially starts up (default is root),
6. Check that the scoreboard file directory is on a locally mounted hard drive rather than an NFS mounted file system.

Audit:

1. Check to see if the `ScoreBoardFile` is specified in any of the Apache configuration files. If it is not present the configuration is compliant.
2. Find the directory in which the `ScoreBoardFile` would be created. The default value is the `ServerRoot/logs` directory.
3. Verify that the scoreboard file directory is not a directory within the Apache `DocumentRoot`
4. Verify that the ownership and group of the directory is `root:root` (or the user under which Apache initially starts up if not root).
5. Change the permissions so that the directory is only writable by root (or the startup user if not root).
6. Check that the scoreboard file directory is on a locally mounted hard drive rather than an NFS mounted file system.

Default Value:
The default scoreboard file is `logs/apache_status`

References:

1. Apache `ScoreBoardFile` directive
   http://httpd.apache.org/docs/2.2/mod/mpm_common.html#scoreboardfile

1.4   Apache Access Control

1.4.1 *Deny Access to OS Root Directory (Level 1, Scorable)*

Description:
The Apache `Directory` directive allows for directory specific configuration of access controls and many other features and options. One important usage is to create a default

deny policy that does not allow access to Operating system directories and files, except for those specifically allowed. This is done, with denying access to the OS root directory.

Rationale:
One aspect of Apache, which is occasionally misunderstood, is the feature of default access. That is, unless you take steps to change it, if the server can find its way to a file through normal URL mapping rules, it can and will serve it to clients. Having a default deny is a predominate security principal, and then helps prevent the unintended access, and we do that in this case by denying access to the OS root directory. The Order directive is important as it provides for other Allow directives to override the default deny.

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find a root `<Directory>` element.
2. Ensure there is a single `Order` directive and set the value to `deny, allow`
3. Ensure there is a `Deny` directive, and set the value `to from all`.
4. Remove any `Allow` directives from the root `<Directory>` element.

```
<Directory />
    . . .
    Order deny,allow
    Deny from all
    . . .
</Directory>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find a root `<Directory>` element.
2. Ensure there is a single `Order` directive with the value of `deny, allow`
3. Ensure there is a `Deny` directive, and with the value of `from all`.
4. Ensure there are no `Allow` directives in the root `<Directory>` element.

The following may be useful in extracting root directory elements from the Apache configuration for auditing.

```
$ perl -ne 'print if /^ *<Directory  *\//i .. /<\/Directory/i'
$APACHE_PREFIX /conf/httpd.conf
```

Default Value:
The following is the default root directory configuration:

```
<Directory />
    . . .
```

```
      Order deny,allow
      Deny from all
</Directory>
```

References:

1. `Directory` Directive http://httpd.apache.org/docs/2.2/mod/core.html#directory

2. Mod Authz http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html

1.4.2 Allow Appropriate Access to Web Content (Level 1, Not Scorable)

Description:
In order to serve Web content the Apache `Allow` directive will be need to be used to allow for appropriate access to directories, locations and virtual hosts that contains web content.

Rationale:
The `Allow` directive is used within a directory, a location or other context to allow appropriate access. Access may be allowed to all, or to specific networks, or specific domain names as appropriate. Refer to the Apache documentation http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html for details.

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (httpd.conf and any included configuration files) to find all `<Directory>` and `<Location>` elements.   There should be one for the document root and any special purpose directories or locations. There are likely to be other access control directives in other contexts, such as virtual hosts or special elements like `<Proxy>`.
2. Add a single `Order` directive and set the value to `deny, allow`
3. Include the appropriate `Allow` and `Deny` directives, with values that are appropriate for the purposes of the directory.

The configurations below are just a few possible examples.

```
<Directory "/var/www/html/">
Order deny,allow
deny from all
allow from 192.169.
</Directory>
```

```
<Directory "/var/www/html/">
Order allow,deny
allow from all
</Directory>
```

```
<Location /usage>
    Order deny,allow
```

```
    Deny from all
    Allow from 127.0.0.1
    Allow from ::1
</Location>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find all `<Directory>` elements.
2. Ensure there is a single **Order** directive with the value of `deny, allow` for each.
3. Ensure the `Allow` and `Deny` directives have values that are appropriate for the purposes of the directory.

The following command may be useful to extract `<Directory>` and `<Location>` elements and `Allow` directives from the apache configuration files.

```
# perl -ne 'print if /^ *<Directory  */i .. /<\/Directory/i'
$APACHE_PREFIX/conf/httpd.conf $APACHE_PREFIX/conf.d/*.conf

# perl -ne 'print if /^ *<Location  */i .. /<\/Location/i'
$APACHE_PREFIX/conf/httpd.conf $APACHE_PREFIX/conf.d/*.conf

# grep -i -C 6 -i 'Allow[[:space:]]from' $APACHE_PREFIX/conf/httpd.conf
$APACHE_PREFIX/conf.d/*.conf
```

Default Value:
The following is the default Web root directory configuration:

```
<Directory "/usr/local/apache2/htdocs">
    . . .
    Order deny,allow
    Allow from all
</Directory>
```

References:

1. `Directory` Directive http://httpd.apache.org/docs/2.2/mod/core.html#directory

2. Mod Authz http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html

1.4.3 Restrict OverRide for the OS Root Directory (Level 1, Scorable)

Description:
The Apache `OverRide` directive allows for `.htaccess` files to be used to override much of the configuration, including authentication, handling of document types, auto generated indexes, access control, and options. When the server finds an `.htaccess` file (as specified by `AccessFileName`) it needs to know which directives declared in that file can override earlier access information. When this directive is set to `None`, then `.htaccess` files are

completely ignored. In this case, the server will not even attempt to read `.htaccess` files in the file system. When this directive is set to `All`, then any directive which has the `.htaccess` Context is allowed in `.htaccess` files.

Refer to the Apache 2.2 documentation for details
http://httpd.apache.org/docs/2.2/mod/core.html#allowoverride

Rationale:
While the functionality of `.htaccess` files is sometimes convenient, usage decentralizes access controls and increases the risk of configurations being changed or viewed inappropriately by an unintended or rogue `.htaccess` file. Consider also that some of the more common vulnerabilities in web servers and web applications allow the web files to be viewed or to be modified. Given this, it is wise to keep the configuration of the web server from being placed in `.htaccess` files

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find a root `<Directory>` element.
2. Add a single `AllowOverride` directive if there is none.
3. Set the value for `AllowOverride` to `None`.

```
<Directory />
    . . .
    AllowOverride None
    . . .
</Directory>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (httpd.conf and any included configuration files) to find a root `<Directory>` element.
2. Ensure there is a single `AllowOverride` directive with the value of `None`.

The following may be useful for extracting root directory elements from the Apache configuration for auditing.

```
$ perl -ne 'print if /^ *<Directory  *\//i .. /<\/Directory/i'
$APACHE_PREFIX /conf/httpd.conf
```

Default Value:
The following is the default root directory configuration:

```
<Directory />
    AllowOverride None
```

```
       . . .
    </Directory>
```

References:

1. `AllowOverride` Directive
   http://httpd.apache.org/docs/2.2/mod/core.html#allowoverride

1.4.4 Restrict OverRide for All Directories (Level 1, Scorable)

Description:
The Apache `AllowOverride` directive allows for `.htaccess` files to be used to override much of the configuration, including authentication, handling of document types, auto generated indexes, access control, and options. When the server finds an `.htaccess` file (as specified by `AccessFileName`) it needs to know which directives declared in that file can override earlier access information. When this directive is set to `None`, then `.htaccess` files are completely ignored. In this case, the server will not even attempt to read `.htaccess` files in the file system. When this directive is set to `All`, then any directive which has the `.htaccess` Context is allowed in `.htaccess` files.

Refer to the Apache 2.2 documentation for details
http://httpd.apache.org/docs/2.2/mod/core.html#allowoverride

Rationale:
While the functionality of htaccess files is sometimes convenient, usage decentralizes the access controls and increases the risk of configurations being changed or viewed inappropriately by an unintended or rogue `.htaccess` file. Consider also that some of the more common vulnerabilities in web servers and web applications allow the web files to be viewed or to be modified. Given this, it is wise to keep the configuration of the web server from being placed in `.htaccess` files

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find `AllowOverride` directives.
2. Set the value for all `AllowOverride` directives to `None`.

```
   . . .
   AllowOverride None
   . . .
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find any `AllowOverride` directives.

2. Ensure there the value for `AllowOverride` is `None`.

```
grep -i AllowOverride $APACHE_PREFIX/conf/httpd.conf
```

Default Value:
Not Applicable

References:

1. `AllowOverride` Directive
   http://httpd.apache.org/docs/2.2/mod/core.html#allowoverride

1.5   Minimize Features, Content and Options

1.5.1   Restrict Options for the OS Root Directory (Level 1, Scorable)

Description:
The Apache `Options` directive allows for specific configuration of options, including execution of CGI, following symbolic links, server side includes, and content negotiation. Refer to the Apache 2.2 documentation for details
http://httpd.apache.org/docs/2.2/mod/core.html#options

Rationale:
The `Options` directive for the root OS level is used to create a default minimal options policy that allows only the minimal options at the root directory level. Then for specific web sites or portions of the web site, options may be enabled as needed and appropriate. No options should be enabled and the value for the Options Directive should be `None`.

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (httpd.conf and any included configuration files) to find a root `<Directory>` element.
2. Add a single `Options` directive if there is none.
3. Set the value for `Options` to `None`.

```
<Directory />
    . . .
    Options None
    . . .
</Directory>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (httpd.conf and any included configuration files) to find a root `<Directory>` element.
2. Ensure there is a single `Options` directive with the value of `None`.

The following may be useful for extracting root directory elements from the Apache configuration for auditing.

```
perl -ne 'print if /^ *<Directory  */i .. /<\/Directory/i'
$APACHE_PREFIX/conf/httpd.conf
```

Default Value:
The following is the default root directory configuration:

```
<Directory />
    Options FollowSymLinks
    . . .
</Directory>
```

References:

1. `Options` Directive http://httpd.apache.org/docs/2.2/mod/core.html#options

1.5.2 Restrict Options for the Web Root Directory (Level 1, Scorable)

Description:
The Apache `Options` directive allows for specific configuration of options, including

- execution of CGI,
- following symbolic links,
- server side includes, and
- content negotiation

Refer to the Apache 2.2 documentation for details
http://httpd.apache.org/docs/2.2/mod/core.html#options

Rationale:
The `Options` directive at the web root or document root level also needs to be restricted to the minimal options required. A setting of `None` is highly recommended, however it is recognized that at this level content negotiation may be needed if multiple languages are supported. No other options should be enabled.

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find the document root `<Directory>` element.
2. Add or modify any existing `Options` directive to have a value of `None` or `Multiviews,` if multi views are needed.

```
<Directory "/usr/local/apache2/htdocs">
    . . .
```

```
     Options None
     . . .
</Directory>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (`httpd.conf` and any included configuration files) to find the document root `<Directory>` elements.
2. Ensure there is a single `Options` directive with the value of `None` or `Multiviews`.

The following may be useful in extracting root directory elements from the Apache configuration for auditing.

```
perl -ne 'print if /^ *<Directory  */i .. /<\/Directory/i'
$APACHE_PREFIX/conf/httpd.conf
```

Default Value:
The following is the default document root directory configuration:

```
<Directory "/usr/local/apache2/htdocs">
    Options Indexes FollowSymLinks
    . . .
</Directory>
```

References:

1. `Options` Directive http://httpd.apache.org/docs/2.2/mod/core.html#options

1.5.3 Minimize Options for Other Directories (Level 1, Scorable)

Description:
The Apache `Options` directive allows for specific configuration of options, including execution of CGI, following symbolic links, server side includes, and content negotiation. Refer to the Apache 2.2 documentation for details http://httpd.apache.org/docs/2.2/mod/core.html#options

Rationale:
Likewise the options for other directories and hosts needs to be restricted to the minimal options required. A setting of `None` is recommended, however it is recognized that other options may be needed in some cases:

- `MultiViews` –Is appropriate if content negotiation is required such as for multiple language are supported. Otherwise, it is recommended that this option be disabled.

- ExecCGI – Enables the execution of CGI scripts via the `mod_cgi` module. It is recommended that this option be disabled for all directories that not are specifically used for CGIs.

- FollowSymLinks & SymLinksIfOwnerMatch – The following of symbolic links is not recommended and should be disabled. The usage of symbolic links opens up additional risk for possible attacks that may use inappropriate symbolic links to access content outside of the document root of the web server. Also consider that it could be combined with a vulnerability that allowed an attacker or insider to create an inappropriate link. The option `SymLinksIfOwnerMatch` is much safer in that the ownership must match for the link to be used, however keep in mind there is additional overhead created by requiring Apache to check the ownership.

- Includes & IncludesNOEXEC – The `IncludesNOEXEC` option should only be needed when server side includes are required. The full `Includes` option should not be used as it also allows execution of arbitrary shell commands. See Apache Mod Include for details http://httpd.apache.org/docs/2.2/mod/mod_include.html. It is recommended that the `Includes` and `IncludesNOEXEC` options be disabled.

- Indexes – The `Indexes` option causes apache to display the web visitor with content listing for the requested directory if no index file is present. It is recommended that the `Indexes` directive be disabled.

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files (httpd.conf and any included configuration files) to find the all `<Directory>` elements.
2. Add or modify any existing `Options` directive to **NOT** have a value of `Includes`. Other options may be set if necessary and appropriate as described above.

Audit:
Perform the following to determine if the recommended state is implemented:

1. Search the Apache configuration files (httpd.conf and any included configuration files) to find the all **`<Directory>`** elements.
2. Ensure that the `Options` directives do not enable **`Includes`**.

The following may be useful for extracting directory elements from the Apache configuration for auditing.

```
perl -ne 'print if /^ *<Directory  */i .. /<\/Directory/i'
$APACHE_PREFIX/conf/httpd.conf

or
```

```
grep -i -A 12 '<Directory[[:space:]]/>' $APACHE_PREFIX/conf/httpd.conf
```

Default Value:
Not Applicable

References:

1. `Options` Directive http://httpd.apache.org/docs/2.2/mod/core.html#options

1.5.4 Remove Default HTML Content (Level 1, Scorable)

Description:
Most Web Servers include Apache installations have default content which is not needed or appropriate for production use. The primary function for these sample content is to provide a default web site, provide user manuals or to demonstrate special features of the web server. All content that is not needed should be removed.

Rationale:
Historically these sample content and features have been remotely exploited and can provide different levels of access to the server. In the Microsoft arena, Code Red exploited a problem with the index service provided by the Internet Information Service. Usually, these routines are not written for production use and consequently little thought was given to security in their development.

Remediation:
Review all pre-installed content and remove content which is not required. In particular look for the unnecessary content which may be found in the document root directory, a configuration directory such as `conf/extra` directory, or as a Unix/Linux package

1. Remove the default index.html or welcome page, if it is a separate package or comment out the configuration if it is part of main Apache `httpd` package such as it is on Red Hat Linux.   Removing a file such as the `welcome.conf` shown below is not recommended as it may get replaced if the package is updated.

```
#
# This configuration file enables the default "Welcome"
# page if there is no default index page present for
# the root URL.  To disable the Welcome page, comment
# out all the lines below.
#
##<LocationMatch "^/+$">
##    Options -Indexes
##    ErrorDocument 403 /error/noindex.html
##</LocationMatch>
```

2. Remove the Apache user manual content or comment out configurations referencing the manual

```
# yum erase httpd-manual
```

3. Remove or comment out any Server Status handler configuration.

```
#
# Allow server status reports generated by mod_status,
# with the URL of http://servername/server-status
# Change the ".example.com" to match your domain to enable.
#
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>
```

4. Remove or comment out any Server Information handler configuration.

```
#
# Allow remote server configuration reports, with the URL of
#  http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".example.com" to match your domain to enable.
#
#<Location /server-info>
#    SetHandler server-info
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>
```

5. Remove or comment out any other handler configuration such as perl-status.

```
# This will allow remote server configuration reports, with the URL of
#  http://servername/perl-status
# Change the ".example.com" to match your domain to enable.
#
#<Location /perl-status>
#    SetHandler perl-script
#    PerlResponseHandler Apache2::Status
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Verify the document root directory and the configuration files do not provide for default `index.html` or welcome page,

2. Ensure the Apache User Manual content is not installed by checking the configuration files for manual location directives.

3. Verify the Apache configuration files do not have the Server Status handler configured.

4. Verify that the Server Information handler is not configured.

5. Verify that any other handler configurations such as `perl-status` is not enabled.

Default Value:
The default source build extra content available in the `/usr/local/apache2/conf/extra/` directory, but the configuration of the extra content is commented out by default. The only default content is a minimal barebones index.html in the document root which contains:

```
<html><body><h1>It works!</h1></body></html>
```

1.5.5 Remove Default CGI Content (Level 1, Scorable)

Description:
Most Web Servers include Apache installations have default CGI content which is not needed or appropriate for production use. The primary function for these sample programs is to demonstrate the capabilities of the web server. All content that is not needed should be removed.

Rationale:
CGI programs have a long history of security bugs and problems associated with improperly accepting user-input. Since these programs are often targets of attackers, we need to make sure that there are no stray CGI programs that could potentially be used for malicious purposes. Usually these programs are not written for production use and consequently little thought was given to security in their development. The `printenv` and `test-cgi` script are common default CGI which disclose inappropriate information about the web server including directory paths and detailed version and configuration information.

Remediation:
Review all pre-installed CGI programs and remove programs which are not required. In particular, look for the unnecessary CGI programs which may be found in the directories configured with `ScriptAlias`, `Script` or other `Script*` directives. Often, CGI directories are named `cgi-bin`. Also, CGI `AddHandler` or `SetHandler` directives may also be in use for specific handlers such as perl, python and PHP.

1. Locate cgi-bin files and directories enabled in the Apache configuration via `Script`, `ScriptAlias` or other `Script*` directives.

2. Remove the printenv default CGI in cgi-bin directory if it is installed.

```
# rm $APACHE_PREFIX/cgi-bin/printenv
```

3. Remove the test-cgi file from the cgi-bin directory if it is installed.

```
# rm $APACHE_PREFIX/cgi-bin/test-cgi
```

4. Review and remove any other cgi-bin files which are not needed for business purposes.

Audit:
Perform the following to determine if the recommended state is implemented:

1. Locate cgi-bin files and directories enabled in the Apache configuration via `Script`, `ScriptAlias` or other `Script*` directives.

2. Ensure the `printenv` CGI is not installed in any configured `cgi-bin` directory.

3. Verify that the `test-cgi` file is not installed in any configured `cgi-bin` directory.

4. Verify that other CGI content has a necessary business function.

Default Value:
The default source build includes the following cgi programs in the `/usr/local/apache2/cgi-bin/` directory.

- printenv
- test-cgi

1.5.6 Limit HTTP Request Methods (Level 1, Scorable)

Description:
Use the Apache `<LimitExcept>` directive to restrict unnecessary HTTP request methods of the web server to only accept and process the `GET`, `HEAD`, `POST` and `OPTIONS` HTTP request methods. Refer to the Apache documentation for more details
http://httpd.apache.org/docs/2.2/mod/core.html#limitexcept

Rationale:
The HTTP 1.1 protocol supports several request methods which are rarely used and potentially high risk. For example, methods such as `PUT` and `DELETE` are rarely used and should be disabled in keeping with the primary security principal of minimize features and options. Also since the usage of these methods is typically to modify resources on the web server, they should be explicitly disallowed. For normal web server operation, you will typically need to allow only the `GET`, `HEAD` and `POST` request methods. This will allow for downloading of web pages and submitting information to web forms. The `OPTIONS` request method will also be allowed as it used to request which HTTP request methods are allowed. Unfortunately the Apache `<LimitExcept>` directive does not deny the `TRACE` request method. The `TRACE` request method will be disallowed in another benchmark recommendation with the `TraceEnable` directive.

Remediation:
Perform the following to implement the recommended state:

1. Locate the Apache configuration files and included configuration files.

2. Search for the `<Directory>` directive on the document root directory such as:

```
<Directory "/usr/local/apache2/htdocs">
. . .
</Directory>
```

3. Ensure that the access control order within the `<Directory>` directive is allow,deny

```
Order allow,deny
```

4. Add a `<LimitExcept>` directive as shown below within the group of document root `<Directory>` directives.

```
# Limit HTTP methods to HEAD, GET and POST, but it does not limit TRACE
    <LimitExcept GET POST OPTIONS>
        deny from all
    </LimitExcept>
```

5. Search for other `<Directory>` directives in the Apache configuration files other than the OS root directory, and add the same `<LimitExcept>` directives to each. It is very important to understand that the `<Directory>` directive is based on the OS file system hierarchy as accessed by Apache and not the hierarchy of the locations within web site URLs.

```
<Directory "/usr/local/apache2/cgi-bin">
    . . .
    Order allow,deny
    . . .
  # Limit HTTP methods
   <LimitExcept GET POST OPTIONS>
       deny from all
    </LimitExcept>

</Directory>
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Locate the Apache configuration files and included configuration files.

2. Search for all `<Directory>` directives other than the on the OS root directory.

3. Ensure that group contains a single `Order` directive within the `<Directory>` directive with a value of `allow, deny`

4. Verify the `<LimitExcept>` directive does not include any HTTP methods other than `GET`, `POST`, and `OPTIONS`. (It may contain fewer methods.).

Default Value:
No Limits on HTTP methods.

References:

1. Apache LimitExcept Directive
http://httpd.apache.org/docs/2.2/mod/core.html#limitexcept

2. HTTP 1.1 RFC http://www.ietf.org/rfc/rfc2616.txt

1.5.7 Disable HTTP TRACE Method (Level 1, Scorable)

Description:
Use the Apache `TraceEnable` directive to disable the HTTP `TRACE` request method. Refer to the Apache documentation for more details
http://httpd.apache.org/docs/2.2/mod/core.html#traceenable

Rationale:
The HTTP 1.1 protocol requires support for the `TRACE` request method which reflects the request back as a response and was intended for diagnostics purposes. The `TRACE` method is not needed and is easily subject to abuse and should be disabled.

Remediation:
Perform the following to implement the recommended state:

1. Locate the main Apache configuration file such as httpd.conf.

2. Add a `TraceEnable` directive to the server level configuration with a value of `off`. Server level configuration is the top level configuration, not nested within any other directives like `<Directory>` or `<Location>`.

```
TraceEnable off
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Locate the Apache configuration files and included configuration files.

2. Verify there is a single `TraceEnable` directive configured with a value of `off`.

Default Value:
The default value is for the `TRACE` method to be enabled.

```
TraceEnable on
```

References:

1. Apache `TraceEnable` Directive
   http://httpd.apache.org/docs/2.2/mod/core.html#traceenable

2. HTTP 1.1 RFC http://www.ietf.org/rfc/rfc2616.txt

1.5.8 Restrict HTTP Protocol Versions (Level 1, Scorable)

Description:
The Apache modules `mod_rewrite` or `mod_security` can be used to disallow old and invalid HTTP protocols versions.   The HTTP version 1.1 RFC is dated June 1999, and has been supported by Apache since version 1.2. It should no longer be necessary to allow ancient versions of HTTP such as 1.0 and prior. Refer to the Apache documentation on `mod_rewrite` for more details http://httpd.apache.org/docs/2.2/mod/mod_rewrite.html

Rationale:
Many malicious automated programs, vulnerability scanners and fingerprinting tools will send abnormal HTTP protocol versions to see how the web server responds. These requests are usually part of the attacker's enumeration process and therefore it is important that we respond by denying these requests.

Remediation:
Perform the following to implement the recommended state:

1. Load the `mod_rewrite` module for Apache by doing either one of the following:

   a. Build Apache with `mod_rewrite` statically loaded during the build, by adding the `--enable-rewrite` option to the `./configure` script.

   ```
   ./configure --enable-rewrite
   ```

   b. Or dynamically loading the module with the `LoadModule` directive in the httpd.conf configuration file.

   ```
   LoadModule rewrite_module modules/mod_rewrite.so
   ```

2. Add the `RewriteEngine` directive to the configuration with the value of `on` so that the rewrite engine is enabled.

   ```
   RewriteEngine On
   ```

3. Locate the main Apache configuration file such as `httpd.conf` and add the following rewrite condition to match HTTP/1.1 and the rewrite rule to the top server level configuration to disallow other protocol versions.

   ```
   RewriteCond %{THE_REQUEST} !HTTP/1\.1$
   RewriteRule .* - [F]
   ```

4. By default, `mod_rewrite` configuration settings from the main server context are not inherited by virtual hosts. Therefore it is also necessary to add the following directives in each `<VirtualHost>` section to inherit the main server settings.

```
RewriteEngine On
RewriteOptions Inherit
```

Audit:
Perform the following to determine if the recommended state is implemented:

1. Locate the Apache configuration files and included configuration files.

2. Verify there is a rewrite condition that disallows requests that do not include the HTTP/1.1 header as shown below.

```
RewriteEngine On
RewriteCond %{THE_REQUEST} !HTTP/1\.1$
```

3. Verify the following directives are included in each `<VirtualHost>` section so that the main server settings will be inherited.

```
RewriteEngine On
RewriteOptions Inherit
```

Default Value:
The default value is for the RewiteEngine:

```
RewriteEngine off
```

References:

1. Apache `Rewrite` Module http://httpd.apache.org/docs/2.2/mod/mod_rewrite.html

1.5.9 Restrict Access to .ht* files (Level 1, Scorable)

Description:
Restrict access to any files beginning with `.ht` using the `FileMatch` directive.

Rationale:
The default name for access filename which allows files in web directories to override the Apache configuration is `.htaccess`. The usage of access files should not be allowed, but as a defense in depth a `FilesMatch` directive is recommended to prevent web clients from viewing those files in case they are created. Also a common name for web password and group files is `.htpasswd` and `.htgroup`. Neither of these files should be placed in the document root, but in the event they are, the `FilesMatch` directive can be used to prevent them from being viewed by web clients.

Remediation:

Perform the following to implement the recommended state:

1. Add or modify the following lines in the apache configuration at the server configuration level.

```
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</FilesMatch>
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that a `FileMatch` directive similar to the one above is present in the apache configuration and not commented out.

Default Value:
The default source build has the `FilesMatch` directive as shown,

```
#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</FilesMatch>
```

References:

1. `FilesMatch` directive http://httpd.apache.org/docs/2.2/mod/core.html#filesmatch

1.5.10 Restrict File Extensions (Level 2, Scorable)

Description:
Restrict access to inappropriate file extensions that are not expected to be a legitimate part of web sites using the `FileMatch` directive.

Rationale:
There are many files that are often left within the web server document root that could provide an attacker with sensitive information. Most often these files are mistakenly left behind after installation, trouble-shooting, or backing up files before editing. Regardless of the reason for their creation, these files can still be served by Apache even when there is no hyperlink pointing to them. The web administrators should use the `FilesMatch` directive to restrict access to only those file extensions that are appropriate for the web server. Rather than create a list of potentially inappropriate file extensions such as `.bak`, `.config`, `.old`, etc, it is recommend instead that a white list of the appropriate and expected file

extensions for the web server be created, reviewed and restricted with a `FileMatch` directive.

Remediation:
Perform the following to implement the recommended state:

1. Compile a list of existing file extension on the web server. The following `find`/`awk` command may be useful, but is likely to need some customization according to the appropriate webroot directories for your web server. Please note that the find skips over any files without a dot (.) in the file name, as these are not expected to be appropriate web content.

```
find */htdocs -type f -name '*.*' | awk -F. '{print $NF }' | sort -u
```

2. Review the list of existing file extensions, for appropriate content for the web server, remove those that are inappropriate and add any additional file extensions expected to be added to the web server in the near future.

3. Add the `FileMatch` directive below which denies access to all files by default.

```
# Block all files by default, unless specifically allowed.
<FilesMatch "^.*$">
Order Deny,Allow
Deny from all
</FilesMatch>
```

4. Add another `FileMatch` directive that allows access to those file extensions specifically allowed from the review process in step 2. An example `FileMatch` directive is below. The file extensions in the regular expression should match your approved list, and not necessarily the expression below.

```
# Allow files with specifically approved file extensions
# Such as  (css, htm; html; js; pdf; txt; xml; xsl; ...),
# images (gif; ico; jpeg; jpg; png; ...), multimedia
<FilesMatch "^.*\.(css|html?|js|pdf|txt|xml|xsl|gif|ico|jpe?g|png)$">
Order Deny,Allow
Allow from all
</FilesMatch>
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that the `FileMatch` directive that denies access to all files is present as shown in step 3 of the remediation with the order of `Deny,Allow`.

2. Verify that there is another `FileMatch` directive similar to the one in step 4 of the remediation, with an expression that matches the approved file extensions.

Default Value:

There is no restriction on file extensions in the default configuration.

```
#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by Web clients.
#
<FilesMatch "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</FilesMatch>
```

References:

1. `FilesMatch` directive http://httpd.apache.org/docs/2.2/mod/core.html#filesmatch

1.6   *Operations -* Logging, Monitoring and Maintenance

Operational procedures of logging, monitoring and maintenance are vital to protecting your web servers as well as the rest of the infrastructure.

1.6.1 Configure the Error Log (Level 1, Scorable)

Description:

The `LogLevel` directive is used to configure the severity level for the error logs. While the `ErrorLog` directive configures the error log file name. The log level values are the standard syslog levels of `emerg, alert, crit, error, warn, notice, info` and `debug`. The recommended level is notice, so that all errors from the `emerg` level through `notice` level will be logged.

Rationale:

The server error logs are invaluable because they can also be used to spot any potential problems before they become serious. Most importantly, they can be used to watch for anomalous behavior such as a lot of "not found" or "unauthorized" errors may be an indication that an attack is pending or has occurred.

Remediation:

Perform the following to implement the recommended state:

1. Add or modify the `LogLevel` in the apache configuration to have a value of `notice` or lower. Note that is it is compliant to have a value of `info` or `debug` if there is a need for a more verbose log and the storage and monitoring processes are capable of handling the extra load. The recommended value is `notice`.

```
LogLevel notice
```

2. Add an `ErrorLog` directive if not already configured. The file path may be relative, absolute, or the logs may be configured to be sent to a syslog server.

```
ErrorLog "logs/error_log"
```

3. Add a similar `ErrorLog` directive for each virtual host configured if the virtual host will have different people responsible for the web site. Each responsible individual or organization needs access to their own web logs, and needs the skills/training/tools for monitor the logs.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify the `LogLevel` in the apache server configuration has a value of `notice` or lower. Note that it is also compliant to have a value of `info` or `debug` if there is a need for a more verbose log and the storage and monitoring processes are capable of handling the extra load. The recommended value is `notice`.

2. Verify the `ErrorLog` directive is configured to an appropriate log file or syslog facility.

3. Verify there is a similar `ErrorLog` directive for each virtual host configured if the virtual host will have different people responsible for the web site.

Default Value:
The following is the default configuration:

```
LogLevel warn
ErrorLog "logs/error_log"
```

References:

1. Apache Log Files http://httpd.apache.org/docs/2.2/logs.html
2. LogLevel http://httpd.apache.org/docs/2.2/mod/core.html#loglevel
3. ErrorLog http://httpd.apache.org/docs/2.2/mod/core.html#errorlog

1.6.2 Configure the Access Log (Level 1, Scorable)

Description:
The `LogFormat` directive defines the format and information to be included in the access log entries. The `CustomLog` directive specifies the log file, syslog facility or piped logging utility.

Rationale:
The server access logs are also invaluable for a variety of reasons. They can be used to determine what resources are being used most.   Most importantly, they can be used to investigate anomalous behavior that may be an indication that an attack is pending or has occurred. If the server only logs errors, and does not log successful access, then it is very difficult to investigate incidents. You may see that the errors stop, and wonder if the attacker gave up, or was the attack successful.

Remediation:

Perform the following to implement the recommended state:

1. Add or modify the `LogFormat` directives in the Apache configuration to use the standard and recommended `combined` format show as shown below.

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\"" combined
```

2. Add or modify the `CustomLog` directives in the Apache configuration to use the `combined` format with an appropriate log file, syslog facility or piped logging utility.

```
CustomLog log/access_log combined
```

3. Add a similar `CustomLog` directives for each virtual host configured if the virtual host will have different people responsible for the web site. Each responsible individual or organization needs access to their own web logs, and needs the skills/training/tools for monitor the logs.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify the `LogFormat` directive in the Apache server configuration has the recommended information parameters.

2. Verify the `CustomLog` directive is configured to an appropriate log file, syslog facility, or piped logging utility and uses the combined format.

3. Verify there is a similar `CustomLog` directives for each virtual host configured if the virtual host will have different people responsible for the web site.

Default Value:
The following are the default log configuration:

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog "logs/access_log" common
```

1.6.3 Log Monitoring (Level 1, Scorable)

Description:
Collecting the Apache logs is an important start, but there must be a monitoring process that will review the logs for indications of potential attack or abuse.   The monitoring process should be at least on a daily process, and more frequently for high risk environment or large installations. Log collection and analysis tools are important to making the monitoring process effective. Some of the most popular and recommended log monitoring tools include:

- LogWatch provides host based log monitoring and is installed by default on many Linux systems,
- Syslog and syslog-ng provides central collection is very popular and recommended for medium and larger installations and would feed into a centralized log monitoring process,
- OSSEC is an open source HIDS which optionally includes log collection and monitoring and is useful for all size organizations.

At a minimum a processed summary of the web request that resulted in errors (Status codes 400's – 599) should be reviewed daily. Detection of other anomalies like spikes in traffic to specific URL's and/or from individual IP addresses are also desirable, as common attack patterns such as password guessing or attempted blind SQL injection may be detected.

Rationale:
Even with sophisticated tools, log monitoring when done correctly is necessarily one of the more expensive security controls, as it requires human effort in the review process as well as investigating of anomalies and handling of incidents. It is a way too common mistake to underestimate the tremendous value to the organization in proper monitoring, and hence resources for monitoring are not sufficient and/or the log filtering and anomaly detection is turned up to the point where attacks and abuse are not detected. Increases in web application attacks has been one of the primary security trends on this past decade and is expected to continue for the foreseeable future.

Remediation:
Perform the following to implement the recommended state:

1. Decide on host based log monitoring and/or central log collection, and perform on either or both of the following:

   a. Install suitable host based log monitoring tools such as LogWatch or OSSEC and configure to send appropriate reports and alerts to an individual or a team monitoring the logs.

   b. Configure Apache to send logs to the syslog daemon and configure syslog to send to a central collection and monitoring system(s).

2. Develop a log monitoring and incident response process and assign appropriate staff with the training and system resources to implement.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that either:

a. A suitable host based log monitoring tools such as LogWatch or OSSEC has been installed and is configured to send appropriate reports and alerts to an individual or a team monitoring the logs.

b. Apache is configured to send logs to the syslog daemon and the syslog is configured to send to another system(s).

2. If possible review the log monitoring and incident response process, to ensure they are assigned to appropriate staff with sufficient training and resources.

Default Value:
LogWatch is installed by default and it is configured by default to send email to the local root. However, LogWatch will not monitor the log files of the default source build, (`/usr/local/apache2/logs`) unless it is configured. Likewise, OSSEC needs to be configured to know where the apache logs are found. Syslog is installed by default, but is not configured to send logs to another host.

References:

1. Logwatch – http://www.logwatch.org/

2. OSSEC HIDS – http://www.ossec.net/

3. Syslog-NG http://www.balabit.com/network-security/syslog-ng/

1.6.4 Log Storage and Rotation (Level 1, Scorable)

Description:
It is important that there is adequate disk space on the partition that will hold all the log files, and that log rotation is configured to retain at least 3 months or 13 weeks if central logging is not used for storage.

Rationale:
Keep in mind that the generation of logs is potentially under an attacker's control. So do not hold any Apache log files on the root partition of the OS. This could result in a denial of service against your web server host by filling up the root partition and causing the system to crash. For this reason it is recommended that the log files should be stored on a dedicated partition. Likewise consider that attackers sometimes put information into your logs which is intended to attack your log collection or log analysis processing software. So it is important that they are not vulnerable. Investigation of incidents often require access to several months or more of logs, which is why it is important to keep at least 3 months available. Two common log rotation utilities include `rotatelogs(8)`, which is bundled with Apache, and `logrotate(8)`, commonly bundled on Linux distributions are described in the remediation section.

Remediation:
To implement the recommended state do either option a) if using the Linux `logrotate` utility or option b) if using a piped logging utility such as the Apache `rotatelogs`:

a) **File Logging with Logrotate**:

1. Add or modify the web log rotation configuration to match your configured log files in `/etc/logrotate.d/httpd` to be similar to the following.

```
/var/log/httpd/*log {
    missingok
    notifempty
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/httpd.pid 2>/dev/null` 2>
/dev/null || true
    endscript
}
```

2. Modify the rotation period and number of logs to keep so that at least 13 weeks or 3 months of logs are retained. This may be done as the default value for all logs in `/etc/logrotate.conf` or in the web specific log rotation configuration in `/etc/logrotate.d/httpd` to be similar to the following.

```
# rotate log files weekly
weekly

# keep 1 years of backlogs
rotate 52
```

3. For each virtual host configured with its own log files ensure that those log files are also included in a similar log rotation.

b) **Piped Logging:**

1. Configure the log rotation interval and log file names to a suitable interval such as daily.

```
CustomLog "|bin/rotatelogs -l /var/logs/logfile.%Y.%m.%d 86400"
combined
```

2. Ensure the log file naming and any rotation scripts provide for retaining at least 3 months or 13 weeks of log files.

3. For each virtual host configured with its own log files ensure that those log files are also included in a similar log rotation.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify the web log rotation configuration matches the Apache configured log files.

2. Verify the rotation period and number of logs to retain is at least 13 weeks or 3 months.

3. For each virtual host configured with its own log files ensure that those log files are also included in a similar log rotation.

Default Value:
The following is the default httpd log rotation configuration in `/etc/logrotate.d/httpd`:

```
/var/log/httpd/*log {
    missingok
    notifempty
    sharedscripts
    postrotate
        /bin/kill -HUP `cat /var/run/httpd.pid 2>/dev/null` 2>
/dev/null || true
    endscript
}
```

The default log retention is configured in `/etc/logrotate.conf`

```
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4
```

1.6.5 Monitor Vulnerability Lists (Level 1, Not Scorable)

Description:
Subscribe to an appropriate security advisory list.

Rationale:
One of the most frustrating aspects of web attacks is that most can be prevented if the appropriate patches are applied. Both OS and web server vendors are constantly issuing patches in response to flaws found within their application's code. Keeping abreast of new patches can be a daunting task to say the least. To keep abreast of issues specific to Apache software and the operating system platform, the individuals responsible for security and/or administration of the server should subscribe to a notification service such as those listed below that will alert them to newly discovered security issues.

Remediation:
Subscribe to one or more of the following to stay abreast of new vulnerabilities.

1. Apache httpd mailing list - http://httpd.apache.org/lists.htmlThe main announcement mailing list is going to tell you whenever a new release of Apache comes out and about security fixes but doesn't usually contain much information

about the actual issues. Serious vulnerabilities tend to get their own advisories written up which also get posted to the announce list.

2. OS Vendor security lists such as Red Hat
   https://www.redhat.com/mailman/listinfo/enterprise-watch-list

3. CERT CC http://www.cert.org/ The Computer Emergency Response Team Co-ordination Centre monitors security incidents – mostly focused on those that have a significant impact. CERT advisories are well researched and a good source of information, especially when CERT was notified of an issue in advance. Not all issues are notified to CERT so it cannot be relied upon as a sole source of information, and since CERT deal with issues across all products and operating systems they are not always able to give immediate updates. Even so, it is well worth subscribing to their alert lists.

4. Security service provider lists – Many security service providers now provide security notifications which can be customized to report only on software and systems that you actual deploy.   If you already subscribe to one of these, then be sure Apache is included in the customization.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Identify personnel responsible updating the Apache software.

2. Interview responsible personnel with regard to how they find out about vulnerabilities and security patches available.

Default Value:
Not Applicable.

1.6.6 Apply Applicable Patches (Level 1, Scorable)

Description:
Apply available Apache patches within 1 month of availability.

Rationale:
Knowing about newly discovered vulnerabilities is only part of the solution. There needs to be a process in place where patches are tested and installed. These patches fix diverse problems, including security issues.

Important Notice: The Apache 2.2.15 was released March 6, 2010 during the consensus process for this benchmark and contains security patches for CVE-2010-0425, CVE-2010-0434 and CVE-2010-0408.

Remediation:
Update to the latest Apache release available according to either of the following:

1. When building from source:

    a. Read release notes and related security patch information

    b. Download latest source and any dependent modules such as `mod_security`.

    c. Build new Apache software according to your build process with the same configuration options.

    d. Install and Test the new software according to your organizations testing process.

    e. Move to production according to your organizations deployment process.

2. When using platform packages such as Red Hat.

    a. Read release notes and related security patch information

    b. Download and install latest available Apache package and any dependent software.

    ```
    # yum update httpd
    ```

    c. Test the new software according to your organizations testing process.

    d. Move to production according to your organizations deployment process.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. When Apache was built from source:

    a. Check the Apache web site for latest versions, date of releases and any security patches. http://httpd.apache.org/security/vulnerabilities_22.html Apache patches are available http://www.apache.org/dist/httpd/patches/

    b. If newer versions with security patches more than 1 month old and are not installed, then the installation is not sufficiently up-to-date.

2. When using platform packages such as Red Hat.

    a. Check for vendor supplied updates such as the yum repository or the vendor web site such as https://www.redhat.com/security/updates/.

    ```
    # yum check-update httpd
    ```

b. If newer versions with security patches more than 1 month old are not installed, then the installation is not sufficiently up-to-date.

Default Value:
Not Applicable

References:

1. Apache vulnerabilities http://httpd.apache.org/security/vulnerabilities_22.html

2. Red Hat Network http://rhn.redhat.com/

3. Red Hat Security Updates https://www.redhat.com/security/updates/

1.7   Use SSL / TLS

1.7.1 Install mod_ssl and/or mod_nss (Level 1, Scorable)

Description:
Secure Sockets Layer (SSL) was developed by Netscape and turned into an open standard, and was renamed Transport Layer Security (TLS) as part of the process. TLS is important for protecting communication and can provide authentication of the server and even the client. However contrary to vendor claims, implementing SSL does NOT directly make your web server more secure! SSL is used to encrypt traffic and therefore does provide confidentiality of private information and users credentials. However, just because you have encrypted the data in transit does not mean that the data provided by the client is secure while it is on the server. Also, SSL does not protect the web server, as attackers will easily target SSL-Enabled web servers, and the attack will be hidden in the encrypted channel. The mod_ssl module is the standard, most used module that implements SSL/TLS for Apache. A newer module found on Red Hat systems can be a compliment or replacement for mod_ssl, and provides the same functionality plus additional security services. The mod_nss is an Apache module implementation of the Network Security Services (NSS) software from Mozilla, which implements a wide range of cryptographic functions in addition to TLS.

Rationale:
It is best to plan for SSL/TLS implementation from the beginning of any new web server. As most web servers have some need for SSL/TLS due to:
- non-public information submitted that should be protected as it's transmitted to the web server.
- non-public information that is downloaded from the web server.
- users are going to be authenticated to some portion of the web server
- there is a need to authenticate the web server to ensure users that they have reached the real web server, and have not been phished or redirected to a bogus site.

Remediation:
Perform either of the following to implement the recommended state:

1. For Apache installations built from the source, use the `--enable-ssl` configure option to add the SSL modules to the build. The `--with-included-apr` configure option may be necessary if there are conflicts with the platform version. See the Apache documentation on building from source http://httpd.apache.org/docs/2.2/install.html for details.

```
# ./configure --with-included-apr  --enable-ssl
```

2. For installations using OS packages, it is typically just a matter of ensuring the `mod_ssl` package is installed. The `mod_nss` package might also be installed. The following yum commands are suitable for Red Hat Linux.

```
# yum install mod_ssl
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Ensure the mod_ssl and/or mod_nss is loaded in the Apache configuration:

```
# httpd -M | egrep 'ssl_module|nss_module'
```

Results should show "Syntax OK" along with either or both of the modules.

Default Value:
The SSL is not enabled by default.

References:

1. Mod_SSL http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

2. Mod_NSS http://directory.fedoraproject.org/wiki/Mod_nss

3. NSS http://www.mozilla.org/projects/security/pki/nss/

1.7.2 Install a valid trusted certificate (Level 1, Scorable)

Description:
The default SSL certificate is self-signed and is not trusted. Install a valid certificate signed by a commonly trusted certificate authority. To be valid, the certificate must be

- Signed by a trusted certificate authority
- not be expired, and
- have a common name that matches the host name of the web server, such as www.example.com.

Rationale:
A digital certificate on your server automatically communicates your site's authenticity to visitors' web browsers. If a trusted authority signs your certificate, it confirms for the

visitor they are actually communicating with you, and not with a fraudulent site stealing credit card numbers or personal information.

Remediation:
Perform the following to implement the recommended state:

1. Decide on the host name to be used for the certificate. It is important to remember that the browser will compare the host name in the URL to the common name in the certificate, so that it is important that all https: URL's match the correct host name. Specifically the host name `www.example.com` is neither the same as example.com nor the same as `ssl.example.com`.

2. Generate a private key using OpenSSL. Although certificate key lengths of 1024 have been common in the past, a key length of 2048 is now recommended for strong authentication. The key must be kept confidential and will be encrypted with a passphrase by default. Follow the steps below and respond to the prompts for a passphrase. See the Apache or OpenSSL documentation for details:

   http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html#realcert
   http://www.openssl.org/docs/HOWTO/certificates.txt

   ```
   # cd /etc/pki/tls/certs
   # umask 077
   # openssl genrsa -aes128 2048 > example.com.key

   Generating RSA private key, 2048 bit long modulus
   ...+++
   ...........+++
   e is 65537 (0x10001)
   Enter pass phrase:
   Verifying - Enter pass phrase:
   ```

3. Generate the certificate signing request (CSR) to be signed by a certificate authority. It is important that common name exactly make the web host name.

   ```
   # openssl req -utf8 -new -key example.com.key -out www.example.com.csr

   Enter pass phrase for example.com.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) [GB]:US
   State or Province Name (full name) [Berkshire]:New York
   Locality Name (eg, city) [Newbury]:Lima
   ```

```
Organization Name (eg, company) [My Company Ltd]:Durkee Consulting
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname)
[]:www.example.com
Email Address []:ralph@example.com

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

# mv www.example.com.key /etc/pki/tls/private/
```

4. Send the certificate signing request (CSR) to a certificate signing authority to be signed, and follow their instructions for submission and validation. The CSR and the final signed certificate are just encoded text, and need to be protected for integrity, but not confidentiality. This certificate will be given out for every SSL connection made.

5. The resulting signed certificate may be named `www.example.com.crt` and placed in `/etc/pki/tls/certs/` as readable by all (mode `0444`). Please note that the certificate authority does not need the private key (`example.com.key`) and this file must be carefully protected. With a decrypted copy of the private key, it would be possible to decrypt all conversations with the server.

6. Do not forget the passphrase used to encrypt the private key. It will be required every time the server is started in https mode. If it is necessary to avoid requiring an administrator having to type the passphrase every time the httpd service is started, the private key may be stored in clear text. Storing the private key in clear text increases the convenience while increasing the risk of disclosure of the key, but may be appropriate for the sake of being able to restart, if the risks are well managed. Be sure that the key file is only readable by root. To decrypt the private key and store it in clear text file the following `openssl` command may be used. You can tell by the private key headers whether it is encrypted or clear text.

```
# cd /etc/pki/tls/private/
# umask 077
# openssl rsa -in example.com.key -out example.com.key.clear
```

7. Locate the Apache configuration file for mod_ssl and add or modify the `SSLCertificateFile` and `SSLCertificateKeyFile` directives to have the correct path for the private key and signed certificate files. If a clear text key is referenced then a passphrase will not be required. You can use the CA's certificate that signed your certificate instead of the CA bundle, to speed up the initial SSL connection as fewer certificates will need to be transmitted.

```
SSLCertificateFile /etc/pki/tls/certs/example.com.crt
SSLCertificateKeyFile /etc/pki/tls/private/example.com.key
```

```
# Default CA file, can be replaced with your CA's certificate.
SSLCACertificateFile /etc/pki/tls/certs/ca-bundle.crt
```

8. Lastly, start or restart the `httpd` service and verify correct functioning with your favorite browser.

Audit:
Perform either or both of the following steps to determine if the recommended state is implemented:

1. OpenSSL can also be used to validate a certificate as a valid trusted certificate, using a trusted bundle of CA certificate. It is important that the CA bundle of certificates be an already validated and trusted file in order for the test to be valid.

```
$ openssl verify -CAfile /etc/pki/tls/certs/ca-bundle.crt -purpose
        sslserver /etc/pki/tls/certs/example.com.crt

/etc/pki/tls/certs/example.com.crt: OK
```

A specific error message and code will be reported **in addition to** the `OK` if the certificate is not valid, For example:

```
error 10 at 0 depth lookup:certificate has expired
OK
```

2. Testing can also be done by connecting to a running web server. This may be done with your favorite browser, a command line web client or with openssl s_client. Of course it is important here as well to be sure of the integrity of the trusted certificate authorities used by the web client. Check out the OWASP testing SSL web page for additional suggestions:

   http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29

Default Value:
Default is an invalid self-signed certificate.

References:

1. OWASP SSL Testing http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29

2. SSL FAQ http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html#realcert

3. OpenSSL How-to http://www.openssl.org/docs/HOWTO/certificates.txt

4. NIST Special Publication 800-52 http://csrc.nist.gov/publications/nistpubs/800-52/SP800-52.pdf

1.7.3 Protect the Servers Private Key (Level 1, Scorable)

Description:
It is critical to protect the server's private key. The server private key is be encrypted by default as a means of protecting it, however having it encrypted means that the passphrase is required each time the server is started up, and now it is necessary to protect the passphrase as well. The passphrase may be typed into when it is manually started up, or provided by an automated program. See
http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslpassphrasedialog for details. To summarize, the options are:

a) Use `SSLPassPhraseDialog builtin` to require a passphrase be manually entered.
b) Use `SSLPassPhraseDialog |/path/to/program` to provide the passphrase.
c) Use `SSLPassPhraseDialog exec:/path/to/program` to provide the passphrase,
d) Store the private key in clear text so that a passphrase is not required.

Any of the above options are acceptable for as long as the key and passphrase are protected as described below. Option a) has the additional security benefit of not storing the passphrase, but is not generally acceptable for most production web servers, since it requires the web server to be manually started. Options b) and c) can provide additional security if the programs providing them are secure. Option d) is the simplest, is widely used and is acceptable as long as the private key is appropriately protected.

Rationale:
If the private key were to be disclosed, it could be used to decrypt all of the SSL communications with the web server, and could also be used to impersonate the web server.

Remediation:
Perform the following to implement the recommended state:

1. All private keys must be stored separately from the public certificates. Find all `SSLCertificateFile` directives in the Apache configuration files. For any `SSLCertificateFile` directives that does not have a corresponding separate `SSLCertificateKeyFile` directive, move the key to a separate file from the certificate, and add the `SSLCertificateKeyFile` directive for the key file.

2. For each the `SSLCertificateKeyFile` directives, change the ownership and permissions on the server's private key to be owned by `root:root` with permission 0400.

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. For each certificate file referenced in the Apache configuration files with the `SSLCertificateFile` directive, examine the file for a private key, clearly identified by the sting "`PRIVATE KEY---`"

2. For each file referenced in the Apache configuration files with the `SSLCertificateKeyFile` directive, verify the ownership is root:root and the permission 0400.

Default Value:
Not applicable.

References:

1. Apache SSL Module http://httpd.apache.org/docs/2.2/mod/mod_ssl.html

2. SSLPassphraseDialog
http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslpassphrasedialog

1.7.4 Restrict weak SSL Protocols and Ciphers (Level 1, Scorable)

Description:
Disable weak SSL protocols and weak ciphers using the `SSLProtocol` and `SSLCipherSuite` directives.

Rationale:
The SSLv2 protocol is flawed and shouldn't be used, as it is subject to man-in-the-middle attacks and other cryptographic attacks. The SSLv3 and TLSv1 protocols should be used instead.

*IMPORTANT NOTICE: There is also a fairly recent (Nov 2009) man-in-the-middle renegotiation attack discovered in SSLv3 and TLSv1.
http://www.phonefactor.com/sslgap/ssl-tls-authentication-patches First a work around and then a fix has been made recently approved as an Internet Standard as RFC 574, Feb 2010.
The work around which removes the renegotiation is available from OpenSSL as version 0.9.8l and 0.9.8m is the current version for the fix at the time of this publication. Test and upgrade to one of these versions when it's available for your platform or download the source from openssl.org. http://www.openssl.org/news/secadv_20091111.txt*

Remediation:
Perform the following to implement the recommended state:

1. Add or modify the following line in the Apache server level configuration and every virtual host that is SSL enabled:

```
SSLProtocol all -SSLv2
```

2. Add or modify the following line in the Apache server level configuration and every virtual host that is SSL enabled:

```
SSLCipherSuite ALL:!EXP:!NULL:!ADH:!LOW:!SSLv2
```

FIPS Compliance: For servers that fall under FIPS 140-2 compliance requirements, SP800-52 provides guidelines for the TLS ciphers. To eliminate usage of the RC4 cipher and MD5

hash which are not deemed FIPS compliant, therefore the cipher suite should disallow MD5 and RC4. Such as:

```
# enable only FIPS 140-2 compliant cyphers
SSLCipherSuite ALL:!EXP:!NULL:!ADH:!LOW:!SSLv2:!MD5:!RC4
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify the `SSLProtocol` directive disables SSLv2 is present in the Apache server level configuration and every virtual host that is SSL enabled. Also verify the `SSLCipherSuite` directive disables weak ciphers in the Apache server level configuration and every virtual host that is SSL enabled.

2. Alternately the SSL protocols and ciphers supported can be easily tested by connecting to a running web server with `openssl s_client` such as shown in http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29

Default Value:
The following are the default modules loaded:

References:

1. SSLProtocol http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslprotocol

2. SSLCipherSuite http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslciphersuite

3. OpenSSL http://www.openssl.org/

4. Testing SSL http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29

1.7.5 Restrict Insecure SSL Renegotiation (Level 1, Scorable)

Description:
There is a somewhat recent (Nov 2009) man-in-the-middle renegotiation attack discovered in SSLv3 and TLSv1. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555 http://www.phonefactor.com/sslgap/ssl-tls-authentication-patches First a work around and then a fix has been made recently approved as an Internet Standard as RFC 574, Feb 2010. The work around which removes the renegotiation is available from OpenSSL as version 0.9.8l and 0.9.8m is the current version for the fix at the time of this publication. Test and upgrade to one of these versions when it's available for your platform, or download the source from openssl.org.
http://www.openssl.org/news/secadv_20091111.txt

The `SSLInsecureRenegotiation` directive was recently added in Apache 2.2.15 for web servers linked with OpenSSL version 0.9.8m or later, to allow the insecure renegotiation to provide backward compatibility to clients with the older unpatched SSL implementations.

While providing backward compatibility, enabling the `SSLInsecureRenegotiation` directive also leaves the server vulnerable to man-in-the-middle renegotiation attack CVE-2009-3555. Therefore the `SSLInsecureRenegotiation` directive should not be enabled.

Rationale:
The seriousness and ramification of this attack warrant that servers and clients be upgraded to support the improved SSL/TLS protocols. Therefore the recommendation is to not enable the insecure renegotiation.

Remediation:
Perform the following to implement the recommended state:

1. Search the Apache configuration files for the `SSLInsecureRenegotiation` directive. If the directive is present modify the value to be `off`. If the directive is not present then no action is required.

```
SSLInsecureRenegotiation off
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Search the Apache configuration files for the `SSLInsecureRenegotiation` directive and verify that the directive is either not present or has a value of `off`.

Default Value:
The default value is `off`:

```
SSLInsecureRenegotiation off
```

References:

1. `SSLInsecureRenegotiation` directive
   http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslinsecurerenegotiation

2. Cert CVE http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555

1.8   Information Leakage

1.8.1   Limit Information in the Server Token (Level 1, Scorable)

Description:
Configure the Apache `ServerTokens` directive to provide minimal information. This is accomplished by setting the value to `Prod` or `ProductOnly`. The only version information given in the server HTTP response header will be "`Apache`" rather than providing detailed on modules and versions installed,

Rationale:
Information is power, and identifying web server details greatly increases the efficiency of any attack, as security vulnerabilities are extremely dependent upon specific software

versions and configurations. Excessive probing and requests may cause too much "noise" being generated and may tip off an administrator. If an attacker can accurately target their exploits, the chances of successful compromise prior to detection increase dramatically. Script Kiddies are constantly scanning the Internet and documenting the version information openly provided by web servers. The purpose of this scanning is to accumulate a database of software installed on those hosts, which can then be used when new vulnerabilities are released.

Remediation:
Perform the following to implement the recommended state:

1. Add or modify the `ServerTokens` directive as shown below to have the value of `Prod` or `ProductOnly`:

```
ServerTokens Prod
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify the `ServerTokens` directive is present in the apache configuration and has a value of `Prod` or `ProductOnly`.

Default Value:
The default value is `Full` which provides the most detailed information.

```
ServerTokens Full
```

Sample Server Header:

```
Server: Apache/2.2.14 (Unix) mod_ssl/2.2.14 OpenSSL/0.9.8e-fips-rhel5
```

References:

1. ServerTokens http://httpd.apache.org/docs/2.2/mod/core.html#servertokens

1.8.2 Limit Information in the Server Signature (Level 1, Scorable)

Description:
Disable the server signatures which generate a signature line as a trailing footer at the bottom of server generated documents such as error pages.

Rationale:
Server signatures are helpful when the server is acting as a proxy, since it helps the user distinguish errors from the proxy rather than the destination server, however in this context there is no need for the additional information and we want to limit leakage of unnecessary information.

Remediation:
Perform the following to implement the recommended state:

1. Add or modify the `ServerSignature` directive as shown below to have the value of `Off`:

```
ServerSignature Off
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify the `ServerSignature` directive is either NOT present in the apache configuration or has a value of `Off`:

Default Value:
The default value is `Off` for `ServerSignature`.

References:

1. `ServerSignature`
   http://httpd.apache.org/docs/2.2/mod/core.html#serversignature

1.8.3 Information Leakage via Default Apache Content (Level 2, Scorable)

Description:
In previous recommendations we have removed default content such as the Apache manuals and default CGI programs. However if you want to further restrict information leakage about the web server, it is important that default content such as icons are left on the web server.

Rationale:
To identify the type of web servers and versions software installed it is common for attackers to scan for icons or special content specific to the server type and version. A simple request like http://example.com/icons/apache_pb2.png may tell the attacker that the server is Apache 2.2 as shown below. The many icons are used primary for auto indexing, which is recommended to be disabled.

Remediation:
Perform either of the following to implement the recommended state:

1. The default source build places the auto-index and icon configurations in the `extra/httpd-autoindex.conf` file, so it can be disabled by leaving the include line commented out in the main httpd.conf file as shown below.

```
# Fancy directory listings
#Include conf/extra/httpd-autoindex.conf
```

2. Alternatively the icon `alias` directive and the directory access control configuration can be commented out as shown:

```
# We include the /icons/ alias for FancyIndexed directory listings.  If
# you do not use FancyIndexing, you may comment this out.
#
#Alias /icons/ "/var/www/icons/"

#<Directory "/var/www/icons">
#    Options Indexes MultiViews FollowSymLinks
#    AllowOverride None
#    Order allow,deny
#    Allow from all
#</Directory>
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that there is no alias or directory access to the apache icons directory in any of the Apache configuration files.

Default Value:
The default source build does not enable access to the Apache icons.

1.9 Miscellaneous Configuration Settings

1.9.1 Denial of Service Mitigation (Level 1, Scorable)

Description:
Denial of Service (DoS) is an attack technique with the intent of preventing a web site from serving normal user activity. DoS attacks, which are normally applied to the network layer, are also possible at the application layer. These malicious attacks can succeed by starving a system of critical resources, vulnerability exploit, or abuse of functionality. Although there is no 100% solution for preventing DoS attacks, the following recommendations use the `Timeout`, `KeepAlive`, and `KeepAliveTimeout` directives to mitigate some of the risk, by requiring more effort for a successful DoS attack. Of course DoS attacks can happen in rather unintentional ways as well as intentional and these directives will help in many of those situations as well.

Rationale:
One common technique for DoS is to initiate many connections to the server. By decreasing the timeout for old connections and we allow the server to free up resources more quickly and be more responsive. In addition we will enable `KeepAlives` which allows for multiple HTTP requests to be sent while keeping the same TCP connection alive. This reduces the overhead of having to setup and tear down new TCP connections. By making the server more efficient will be more resilient to DoS conditions. The `Timeout` directive affects several timeout values for Apache, so review the Apache document carefully.
http://httpd.apache.org/docs/2.2/mod/core.html#timeout

Important Notice: There is a slow form of DoS attack not adequately mitigated by these control, such as the Slow Loris DoS attack of June 2009 http://ha.ckers.org/slowloris/
A new module `mod_reqtimeout` has been introduced in the recent 2.2.15 release, to provide tools for mitigating these forms of attack; however the module is marked experimental at this time. http://httpd.apache.org/docs/2.2/mod/mod_reqtimeout.html

Remediation:
Perform the following to implement the recommended state:

1. Add or modify the `Timeout` directive in the Apache configuration to have a value of `10` seconds or shorter.

   ```
   Timeout 10
   ```

2. Add or modify the `KeepAlive` directive in the Apache configuration to have a value of `On`, so that `KeepAlive` connections are enabled.

   ```
   KeepAlive On
   ```

3. Add or modify the `MaxKeepAliveRequests` directive in the Apache configuration to have a value of 100 or more.

   ```
   MaxKeepAliveRequests 100
   ```

4. Add or modify the `KeepAliveTimeout` directive in the Apache configuration to have a value of 15 or less.

   ```
   KeepAliveTimeout 15
   ```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that the `Timeout` directive is specified in the Apache configuration files to have a value of 30 seconds or shorter.

2. Verify that the `KeepAlive` directive in the Apache configuration to have a value of `On`, or is not present. If the directive is not present the default value is `On`.

3. Verify that the `MaxKeepAliveRequests` directive in the Apache configuration to have a value of 100 or more. If the directive is not present the default value is 100.

4. Verify that the `KeepAliveTimeout` directive in the Apache configuration to have a value of 15 or less. . If the directive is not present the default value is 15 seconds.

Default Value:
The following are the default values for each directive:

```
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
```

References:

1. Timeout http://httpd.apache.org/docs/2.2/mod/core.html#timeout

2. KeepAlive http://httpd.apache.org/docs/2.2/mod/core.html#keepalive

3. KeepAliveTimeout
   http://httpd.apache.org/docs/2.2/mod/core.html#keepalivetimeout

4. MaxKeepAliveRequests
   http://httpd.apache.org/docs/2.2/mod/core.html#maxkeepaliverequests

1.9.2 Buffer Overflow Mitigation (Level 2, Scorable)

Description:
Buffer Overflow attacks attempt to exploit an application by providing more data than the
application buffer can contain. If the application allows copying data to the buffer to
overflow the boundaries of the buffer, then the application is vulnerable to a buffer
overflow. The results of Buffer overflow vulnerabilities varies, and may result in the
application crashing, or may allow the attacker to execute instructions provided in the data.
The Apache `LimitRequest*` directives allow the Apache web server to limit the sizes of
requests and request fields and can be used to help protect programs and applications
processing those requests.

Rationale:
The limiting the sizes of requests is helpful so that the web server can prevent an
unexpectedly long or large requests from being passed to a potentially vulnerable CGI
program, module or application that would have attempted to process the request. Of
course the underlying dependency is that we need to set the limits high enough to not
interfere with any one application on the server, while setting them low enough to be of
value in protecting the applications. Since the configuration directives are available only at
the server configuration level, it is not possible to tune the value for different portions of
the same web server. Please read the Apache documentation carefully, as these requests
may interfere with the expected functionality of some web applications

Remediation:
Perform the following to implement the recommended state:

1. Add or modify the `LimitRequestline` directive in the Apache configuration to have a
   value less than or equal to `512`.

```
LimitRequestline 512
```

2. Add or modify the `LimitRequestFields` directive in the Apache configuration to have a value of `100` or less. If the directive is not present the default depends on a compile time configuration, but defaults to a value of `100`.

```
LimitRequestFields 100
```

3. Add or modify the `LimitRequestFieldsize` directive in the Apache configuration to have a value of `1024` or less.

```
LimitRequestFieldsize 1024
```

4. Add or modify the `LimitRequestBody` directive in the Apache configuration to have a value of `102400` (100K) or less. Please read the Apache documentation so that it is understood that this directive will limit the size of file up-loads to the web server.

```
LimitRequestBody 102400
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that the `LimitRequestline` directive is in the Apache configuration and has a value of 512 or less.

2. Verify that the `LimitRequestFields` directive is in the Apache configuration and has a value of 100 or less.

3. Verify that the `LimitRequestFieldsize` directive is in the Apache configuration and has a value of 1024 or less.

4. Verify that the `LimitRequestBody` directive in the Apache configuration to have a value of 102400 (100K) or less.

Default Value:
The following are the default values:

```
LimitRequestline 8190
LimitRequestFields 100
LimitRequestFieldsize 8190
LimitRequestBody 0   (unlimited)
```

References:

1. LimitRequestline
   http://httpd.apache.org/docs/2.2/mod/core.html#limitrequestline

2. LimitRequestFields
   http://httpd.apache.org/docs/2.2/mod/core.html#limitrequestfields

3. LimitRequestFieldsize
http://httpd.apache.org/docs/2.2/mod/core.html#limitrequestfieldsize

4. LimitRequestBody
http://httpd.apache.org/docs/2.2/mod/core.html#limitrequestbody

1.9.3 Restrict Listen Directive (Level 2, Scorable)

Description:
The Apache `Listen` directive specifies the IP addresses and port numbers the Apache web server will listen for requests. Rather than be unrestricted to listen on all IP addresses available to the system, the specific IP address or addresses intended should be explicitly specified. Specifically a Listen directive with no IP address specified, or with an IP address of zero's should not be used.

Rationale:
Having multiple interfaces on web servers is fairly common, and without explicit `Listen` directives, the web server is likely to be listening on an inappropriate IP address / interface that was not intended for the web server. Single homed system with a single IP addressed are also required to have an explicit IP address in the Listen directive, in case additional interfaces are added to the system at a later date.

Remediation:
Perform the following to implement the recommended state:

1. Find any `Listen` directives in the Apache configuration file with no IP address specified, or with an IP address of all zero's similar to the examples below. Keep in mind there may be both IPv4 and IPv6 addresses on the system.

```
Listen 80
Listen 0.0.0.0:80
Listen [::ffff:0.0.0.0]:80
```

2. Modify the `Listen` directives in the Apache configuration file to have explicit IP addresses according to the intended usage. Multiple `Listen` directives may be specified for each IP address & Port.

```
Listen 10.1.2.3:80
Listen 192.168.4.5:80
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

Audit:
Perform the following steps to determine if the recommended state is implemented:

1. Verify that no `Listen` directives are in the Apache configuration file with no IP address specified, or with an IP address of all zero's.

Default Value:

The following is the default value:

```
Listen 80
```

References:

1. Listen Directive http://httpd.apache.org/docs/2.2/mod/mpm_common.html#listen

Appendix A: References

1. Apache Software Foundation (2009). *Apache HTTP Server Version 2.2 Documentation*. Available http://httpd.apache.org/docs/2.2/ Last accessed Feb 2010.

2. National Institute of Standards and Technology. (2009). *Checklist Details for Web Apache Checklist Version 6, Release 1.11*. Available: http://web.nvd.nist.gov/view/ncp/repository/checklistDetail?id=94. Last accessed Feb 2010.

Appendix B: Further Reading

1. The Apache `ChrootDir` Directive provides a means to restrict access for the web server running as the apache user by either running the web server in a chroot "jailed" environment and would have been included, except that it is still fairly new, and not yet widely used in production environment. Adding the directive should be considered for future updates. Document is available http://httpd.apache.org/docs/2.2/mod/mpm_common.html#chrootdir

2. A good alternative chroot solution is the `mod_security` chroot approach which is commonly used, but will now be covered as a separate benchmark. With documentation available http://www.modsecurity.org/documentation/apache-internal-chroot.html

3. A second alternative to the `ChrootDir` considered was the SELinux which takes a different approach to implement similar restrictions through mandatory access controls and is commonly used, but not a native part of Apache. Additional information can be found here:

    a. Main Project http://selinuxproject.org/
    b. Fedora http://fedoraproject.org/wiki/SELinux
    c. RHEL5 http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Deployment_Guide-en-US/ch-selinux.html
    d. Debian http://wiki.debian.org/SELinux
    e. Gentoo www.gentoo.org/proj/en/hardened/selinux/

Appendix C: DISA Web SRR Checklist Mapping

The following table maps the requirements articulated in
`SEC_5_Apache_2_x_Instance_V6R1.11.pdf` as obtained from References [2]

Item	DISA	CIS
Timeout	WA000-WWA020 (300 or less)	1.9.1 (10 or less)
KeepAlive	WA000-WWA022 (On)	1.9.1 (On)
KeepAliveTimeout	WA000-WWA024 (15 or less)	1.9.1 (15 or less)
StartServers	WA000-WWA026 (Between 5 and 10)	NA
MinSpareServers	WA000-WWA028 (Between 5 and 10)	NA
MaxSpareServers	WA000-WWA030 (10 or less)	NA
MaxClients	WA000-WWA032 (256 or less)	NA
Restrict ExecCGI	WA000-WWA050	1.5.3
Restrict FollowSymLinks	WA000-WWA052	1.5.3
Restrict IncludesNOEXEC	WA000-WWA054	1.5.3
Restrict MultiViews	WA000-WWA056	1.5.3
Restrict Indexes	WA000-WWA058	1.5.3
LimitRequestBody	WA000-WWA060 (Not 0)	1.9.2 (102400 or less)
LimitRequestFields	WA000-WWA062 (Not 0)	1.9.2 (100 or less)
LimitRequestFieldsize	WA000-WWA064 (8190 or less)	1.9.2 (1024 or less)
LimitRequestline	WA000-WWA066 (8190 or less)	1.9.2 (512 or less)
Trained Staff	WA050	NA
Public Server Isolation	WA060	NA
Private Server Isolation	WA070	NA
Log Archive	WA110	1.6.1-1.6.4
User/Group Documentation	WA120	NA
Configuration Backup	WA140	NA
Physically Secure Classified Systems	WA155	NA
Base OS Not Secured	WA160	NA
Incident Response Procedure	WA170	NA
Software Upgrade Plan	WA200	NA
Patch Apache	WA230	1.6.6
Public/Private Server Connection	WG040	NA
Web admin has service password	WG050	NA
Change web service password annually	WG060	NA
Remove Compilers	WG080	NA
Remove unneeded programs	WG130	NA
Remove unneeded services	WG135	1.1.2
Use DoD certificate validation	WG145	1.7.2,1.7.3
Unsupported Apache Version	WG190	1.1.3
Only Administrators Have Shell Access	WG200	NA
Single Use System	WG204	1.1.2

Only Admins Access Web Administration	WG220	NA
Anti Virus Used for Uploads	WG237	NA
Secure .htpasswd Files	WG270	1.5.9
Run as non-privileged account	WG275	1.3.1
Configuration file ownership	WG280	1.3.4
Configuration file permissions	WG300	1.3.5
Disable external mail facilities	WG330	1.1.2
Server's Trusted Roots link to DoD CA	WG335	NA
Remove handlers for CSH and SH	WG370	NA
Remove Vulnerable Programs	WG380	NA
Remove Sample Code and Docs	WG385	1.5.4,1.5.5
Remove Backup Files	WG420	1.5.10
CGIs are Monitored	WG440	NA
Users Access Win32 Scripting Hosts	WG470	NA
Web Server or OS Information Leaked	WG520	1.8.1-1.8.3
Limit AuthN and AuthZ Modules	NA	1.2.1
Enable Log Config Module	NA	1.2.2
Disable WebDAV Modules	NA	1.2.3
Disable Status and Info Modules	NA	1.2.4
Disable Autoindex Module	NA	1.2.5
Disable Proxy Modules	NA	1.2.6
Disable Userdir Module	NA	1.2.7
Invalidate Apache User's Shell	NA	1.3.2
Lock Apache User	NA	1.3.3
Secure Core Dumps	NA	1.3.6
Secure LockFile	NA	1.3.7
Secure PidFile	NA	1.3.8
Secure ScoreBoardFile	NA	1.3.9
Protect OS Root	NA	1.4.1
Restrict Overrides for OS Root	NA	1.4.3
Restrict Directory Overrides	NA	1.4.4
Restrict Options for OS Root	NA	1.5.1
Restrict Options for Web Root	NA	1.5.2
Minimize Options for other Directories	NA	1.5.3
Limit HTTP Request Methods	NA	1.5.6
Disable HTTP Trace Method	NA	1.5.7
Restrict HTTP Protocol Version	NA	1.5.8
Restrict File Extensions	NA	1.5.10
Restrict weak SSL Protocols and Ciphers	NA	1.7.4
Restrict Insecure SSL Renegotiation	NA	1.7.5
Restrict Listen Directive	NA	1.9.3

Appendix D: Change History

Date	Version	Changes for this version
May 17th, 2010	3.0.0	Release Major Update with new format with Auditable items by Ralph Durkee Summary of Changes: New Items: 1.1.2 Do not Install on a Multi-use System (Level 2, Not Scorable) 1.2.1 Enable only necessary Authentication and Authorization Modules (Level 1, Scorable) 1.2.2 Enable the Log Config Module (Level 1, Scorable) 1.2.3 Disable WebDAV modules (Level 1, Scorable) 1.2.4 Disable Status and Info modules (Level 1, Scorable) 1.2.5 Disable Autoindex module (Level 1, Scorable) 1.2.6 Disable Proxy Modules (Level 1, Scorable) 1.3.3 Lock the Apache User Account (Level 1, Scorable) 1.3.6 Core Dump Directory Security (Level 1, Scorable) 1.3.7 Lock File Security (Level 1, Scorable) 1.3.8 Pid File Security (Level 1, Scorable) 1.3.9 ScoreBoard File Security (Level 1, Scorable) 1.4.4 Restrict OverRide for All Directories (Level 1, Scorable) 1.5.1 Restrict Options for the OS Root Directory (Level 1, Scorable) 1.5.7 Disable HTTP TRACE Method (Level 1, Scorable) 1.5.9 Restrict Access to .ht* files (Level 1, Scorable) 1.6.3 Log Monitoring (Level 1, Scorable) 1.7.3 Protect the Servers Private Key (Level 1, Scorable) 1.7.5 Restrict Insecure SSL Renegotiation (Level 1, Scorable) 1.8.3 Information Leakage via Default Apache Content (Level 2, Scorable) 1.9.3 Restrict Listen Directive (Level 2, Scorable) Removed Content: - Usage of mod_security is moved to a separate benchmark 2.2 Mod_log_Forensic 2.3 Denial of Service and Brute Force Identification and

		Response • 2.4 Buffer Overflow Protection Tuning • 2.6 Virtual Patching with ModSecurity • 2.7 Additional Software Information Leakage Protection Major Changes: 2. Mod_Security alternate implementation for 1.5.6 Limit HTTP Request Methods and 1.5.10 Restrict File Extensions (Level 2, Scorable) and chroot are not included.
October, 2008	2.2.0	• Updated 1.19 to refer to /usr/local/apache2 instead of /usr/local/apache. • Updated 2.1 to refer to /usr/local/apache2 instead of /usr/local/apache. • Updated 2.2 to refer to /usr/local/apache2 instead of /usr/local/apache. • Updated 2.5 to refer to /usr/local/apache2 instead of /usr/local/apache. • Formatted inline 'verbatim' text for readability. • Added new cover page, Change History, and TOU. • Added footer with page numbers • Updated 1.14 example to match explanation text. • Updated 1.17 example to reference /var/log/httpd for logging. This aligns with the recommendation to log off the server root. • Updated 1.21 to state users should use the update mechanism provided by their OS to update Apache. • Updated 1.15 ssl config to reference /var/log/httpd for logging location vie logs/. • Updated 1.15 to disallow SSLv2 Ciphers • Added Acknowledgement section • Moved the SecServerSignature and ErrorDocument recommendations within 1.16 to new level 2 recommendation – 2.7.