CENTER FOR
INTERNET SECURITY

CIS Apache HTTP Server 2.2 Benchmark

v3.3.1 - 04-23-2015

http://benchmarks.cisecurity.org

The CIS Security Benchmarks division provides consensus-oriented information security products, services, tools, metrics, suggestions, and recommendations (the "SB Products") as a public service to Internet users worldwide. Downloading or using SB Products in any way signifies and confirms your acceptance of and your binding agreement to these CIS Security Benchmarks Terms of Use.

CIS SECURITY BENCHMARKS TERMS OF USE

BOTH CIS SECURITY BENCHMARKS DIVISION MEMBERS AND NON-MEMBERS MAY:

- Download, install, and use each of the SB Products on a single computer, and/or
- Print one or more copies of any SB Product that is in a .txt, .pdf, .doc, .mcw, or .rtf format, but only if each such copy is printed in its entirety and is kept intact, including without limitation the text of these CIS Security Benchmarks Terms of Use.

UNDER THE FOLLOWING TERMS AND CONDITIONS:

- **SB Products Provided As Is**. CIS is providing the SB Products "as is" and "as available" without: (1) any representations, warranties, or covenants of any kind whatsoever (including the absence of any warranty regarding: (a) the effect or lack of effect of any SB Product on the operation or the security of any network, system, software, hardware, or any component of any of them, and (b) the accuracy, utility, reliability, timeliness, or completeness of any SB Product); or (2) the responsibility to make or notify you of any corrections, updates, upgrades, or fixes.
- **Intellectual Property and Rights Reserved**. You are not acquiring any title or ownership rights in or to any SB Product, and full title and all ownership rights to the SB Products remain the exclusive property of CIS. All rights to the SB Products not expressly granted in these Terms of Use are hereby reserved.
- **Restrictions**. You acknowledge and agree that you may not: (1) decompile, dis-assemble, alter, reverse engineer, or otherwise attempt to derive the source code for any software SB Product that is not already in the form of source code; (2) distribute, redistribute, sell, rent, lease, sublicense or otherwise transfer or exploit any rights to any SB Product in any way or for any purpose; (3) post any SB Product on any website, bulletin board, ftp server, newsgroup, or other similar mechanism or device; (4) remove from or alter these CIS Security Benchmarks Terms of Use on any SB Product; (5) remove or alter any proprietary notices on any SB Product; (6) use any SB Product or any component of an SB Product with any derivative works based directly on an SB Product or any component of an SB Product; (7) use any SB Product or any component of an SB Product with other products or applications that are directly and specifically dependent on such SB Product or any component for any part of their functionality; (8) represent or claim a particular level of compliance or consistency with any SB Product; or (9) facilitate or otherwise aid other individuals or entities in violating these CIS Security Benchmarks Terms of Use.
- **Your Responsibility to Evaluate Risks**. You acknowledge and agree that: (1) no network, system, device, hardware, software, or component can be made fully secure; (2) you have the sole responsibility to evaluate the risks and benefits of the SB Products to your particular circumstances and requirements; and (3) CIS is not assuming any of the liabilities associated with your use of any or all of the SB Products.
- **CIS Liability**. You acknowledge and agree that neither CIS nor any of its employees, officers, directors, agents or other service providers has or will have any liability to you whatsoever (whether based in contract, tort, strict liability or otherwise) for any direct, indirect, incidental, consequential, or special damages that arise out of or are connected in any way with your use of any SB Product.
- **Indemnification**. You agree to indemnify, defend, and hold CIS and all of CIS's employees, officers, directors, agents and other service providers harmless from and against any liabilities, costs and expenses incurred by any of them in connection with your violation of these CIS Security Benchmarks Terms of Use.
- **Jurisdiction**. You acknowledge and agree that: (1) these CIS Security Benchmarks Terms of Use will be governed by and construed in accordance with the laws of the State of Maryland; (2) any action at law or in equity arising out of or relating to these CIS Security Benchmarks Terms of Use shall be filed only in the courts located in the State of Maryland; and (3) you hereby consent and submit to the personal jurisdiction of such courts for the purposes of litigating any such action.
- **U.S. Export Control and Sanctions laws**. Regarding your use of the SB Products with any non-U.S. entity or country, you acknowledge that it is your responsibility to understand and abide by all U.S. sanctions and export control laws as set from time to time by the U.S. Bureau of Industry and Security (BIS) and the U.S. Office of Foreign Assets Control (OFAC).

SPECIAL RULES FOR CIS MEMBER ORGANIZATIONS: CIS reserves the right to create special rules for: (1) CIS Members; and (2) Non-Member organizations and individuals with which CIS has a written contractual relationship. CIS hereby grants to each CIS Member Organization in good standing the right to distribute the SB Products within such Member's own organization, whether by manual or electronic means. Each such Member Organization acknowledges and agrees that the foregoing grants in this paragraph are subject to the terms of such Member's membership arrangement with CIS and may, therefore, be modified or terminated by CIS at any time.

Table of Contents

Overview

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 HTTP Server 2.2 running on Linux.

Intended Audience

This document, CIS Apache 2.2 Benchmark, provides prescriptive guidance for establishing a secure configuration posture for Apache Web Server versions 2.2 running on Linux. This guide was tested against Apache Web Server 2.2.29 as built from source httpd-2.2.29.tar.gz from http://httpd.apache.org/ on Linux. To obtain the latest version of this guide, please visit http://benchmarks.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 benchmark was created using a consensus review process comprised 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 published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the benchmark. If you are interested in participating in the consensus process, please visit https://community.cisecurity.org.

Typographical 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

Scoring Information

A scoring status indicates whether compliance with the given recommendation impacts the assessed target's benchmark score. The following scoring statuses are used in this benchmark:

Scored

Failure to comply with "Scored" recommendations will decrease the final benchmark score. Compliance with "Scored" recommendations will increase the final benchmark score.

Not Scored

Failure to comply with "Not Scored" recommendations will not decrease the final benchmark score. Compliance with "Not Scored" recommendations will not increase the final benchmark score.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Items in this profile intend to:

  - be practical and prudent;
  - provide a clear security benefit; and
  - not inhibit the utility of the technology beyond acceptable means.

- **Level 2**

  This profile extends the "Level 1" profile. Items in this profile 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.

Acknowledgements

This benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Author
Ralph Durkee CISSP, GSEC, GCIH, GSNA, GPEN, C|EH, *Durkee Consulting, Inc.*

Contributor
Ahmed Adel GSEC, GCIH, GCFW, GWAN
Ryan Barnett
Quan Bui
Lawrence Grim
Blake Frantz , *Center for Internet Security*
Peter Morin CISA, CGEIT, GCFA , *Bell Canada*
Mihai Nitulescu
Eduardo Petazze
Art Stricek CompTIA Security+
Vytautas Vysniauskas PhD, *Consumer Direct*
Roger Kennedy Linux Systems Engineer
Adam Montville

Recommendations

1 Recommendations

1.1 Planning and Installation

This section contains recommendations for the planning and installation of an Apache HTTP Server.

1.1.1 Pre-Installation Planning Checklist (Not Scored)

Profile Applicability:

- Level 1

Description:

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.

Rationale:

N/A

Audit:

N/A

Remediation:

N/A

1.1.2 Do Not Install a Multi-use System (Not Scored)

Profile Applicability:

- Level 2

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.

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'
```

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
```

1.1.3 Installing Apache (Not Scored)

Profile Applicability:

- Level 1

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 through 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 less 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.

Audit:

N/A

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
```

References:

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

1.2 Minimize Apache Modules

It's crucially important to have a minimal and compact Apache installation based on documented business requirements. The remaining of this section covers specific modules that should be reviewed and disabled if not required for business purposes. However it's very important that the review and analysis of which modules are required for business purpose not be limited to the modules explicitly listed.

1.2.1 Enable only necessary Authentication and Authorization Modules (Not Scored)

Profile Applicability:

- Level 1

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. Enable only the modules that are required.

Rationale:

Authentication and authorization are your front doors to the protected information in your web site. Most installation only need a small subset of the modules available. By minimizing the enabled modules to those that are actually used, we reduce the number of "doors" and have therefore reduce the attack surface of the web site. Likewise having fewer modules means less software that could have vulnerabilities.

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.

Remediation:

Consult Apache module documentation for descriptions of each module in order to determine the necessary modules for the specific installation. The unnecessary static compiled modules are disabled through compile time configuration options. 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.

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 (Scored)

Profile Applicability:

- Level 1

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

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

Remediation:

Perform either one of the following:

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

```
$ cd $DOWNLOAD/httpd-2.2.22
$ ./configure
```

- 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
```

References:

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

1.2.3 Disable WebDAV Modules (Scored)

Profile Applicability:

- Level 1

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.

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.

Remediation:

Perform either one of the following to disable WebDAV module:

1. 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-2.2.22
$ ./configure
```

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

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

References:

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

1.2.4 Disable Status Module (Scored)

Profile Applicability:

- Level 1

Description:

The Apache `mod_status` module provides current server performance statistics.

Rationale:

While having server performance status information available as a web page may be convenient, it's recommended that this module be disabled:

- When `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.

Audit:

Perform the following to determine if the Status module is enabled.

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

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

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

Remediation:

Perform either one of the following to disable the mod_status module:

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

```
$ cd $DOWNLOAD/httpd-2.2.22
$ ./configure --disable-status
```

2. For dynamically loaded modules comment out or remove the `LoadModule` directive for the `mod_status` module from the `httpd.conf` file.

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

References:

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

1.2.5 Disable Autoindex Module (Scored)

Profile Applicability:

- Level 1

Description:

The Apache `autoindex` module automatically generates web page listing the contents of directories on the server, typically used so that an `index.html` does not have to generated

Rationale:

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

Audit:

Perform the following to determine if the module is enabled.

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

```
# httpd -M | grep autoindex_module
```

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

Remediation:

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

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

```
$ cd $DOWNLOAD/httpd-2.2.22
$ ./configure -disable-autoindex
```

2. 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
```

References:

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

1.2.6 Disable Proxy Modules (Scored)

Profile Applicability:

- Level 1

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.

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

Remediation:

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

1. 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-2.2.22
$ ./configure
```

2. 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
```

References:

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

1.2.7 Disable User Directories Modules (Scored)

Profile Applicability:

- Level 1

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 (`/`).

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.

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.

Remediation:

Perform either one of the following to disable the user directories module:

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

```
$ cd $DOWNLOAD/httpd-2.2.22
$ ./configure --disable-userdir
```

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

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

References:

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

1.2.8 Disable Info Module (Scored)

Profile Applicability:

- Level 1

Description:

The Apache `mod_info` module provides information on the server configuration via access to a `/server-info` URL location.

Rationale:

While having server configuration information available as a web page may be convenient it's recommended that this module 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.

Audit:

Perform the following to determine if the Info module is enabled.

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

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

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

Remediation:

Perform either one of the following to disable the mod_info module:

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

```
$ cd $DOWNLOAD/httpd-2.2.22
$ ./configure
```

2. For dynamically loaded modules comment out or remove the `LoadModule` directive for the `mod_info` module from the `httpd.conf` file.

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

References:

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

1.3 Principles, Permissions, and Ownership

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 (Scored)

Profile Applicability:

- Level 1

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 user and group for the server application. The "nobody" or "daemon" user and group that comes default on Unix variants should NOT be used to run the web server, since the account is commonly used for other separate daemon services. Instead, an account used only by the apache software so as to not give unnecessary access to other services. Also the user 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 not used by regular users, such as discussed in User Accounts section of the CIS Red Hat benchmark. As an even more secure alternative, if the Apache web server can be run on high unprivileged ports, then it is not necessary to start Apache as root, and all of the Apache processes may be run as the Apache specific user as described below.

Audit:

Ensure the apache account is unique and has been created with a UID between 1-499 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)
```

3. While the web server is running check the user id for the httpd processes. The user name should match the configuration file.

```
# ps axu | grep httpd | grep -v '^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
```

1.3.2 Give the Apache User Account an Invalid Shell (Scored)

Profile Applicability:

- Level 1

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.

Audit:

Check the apache login shell in the `/etc/passwd` 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`

Remediation:

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

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

1.3.3 Lock the Apache User Account (Scored)

Profile Applicability:

- Level 1

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 have to `su` as apache, and when there is a need, then `sudo` should be used instead, which would not require the apache account password.

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.)
```

Remediation:

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

```
# passwd -l apache
```

1.3.4 Set Ownership on Apache Directories and Files (Scored)

Profile Applicability:

- Level 1

Description:

The Apache directories and files should be owned by root. This applies to all of the Apache software directories and files installed.

Rationale:

Restricting ownership of the Apache files and directories will reduce the probability of unauthorized modifications to those resources.

Audit:

- Identify files in the Apache directory not owned by `root` :

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

Remediation:

Perform the following:

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

```
$ chown -R root $APACHE_PREFIX
```

Default Value:

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

1.3.5 Set Group Id on Apache Directories and Files (Scored)

Profile Applicability:

- Level 1

Description:

The Apache directories and files should be set to have a group Id of root, (or a 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`) is likely to need a designated group to allow web content to be updated (such as webupdate) through a change management process.

Rationale:

Securing Apache files and directories will reduce the probability of unauthorized modifications to those resources.

Audit:

Identify files in the Apache directories other than htdocs with a group other than root:

```
# find $APACHE_PREFIX -path $APACHE_PREFIX/htdocs -prune -o \! -group root -ls
```

Remediation:

Perform the following:

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

```
$ chgrp -R root $APACHE_PREFIX
```

Default Value:

Default group is a mixture of the user group that built the software and root.

1.3.6 Restrict Other Write Access on Apache Directories and Files (Scored)

Profile Applicability:

- Level 1

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:

None of the Apache files and directories, including the Web document root must allow other write access. Other write access is likely to be very useful for unauthorized modification of web content, configuration files or software for malicious attacks.

Audit:

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

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

Remediation:

Perform the following to remove other write access on the `$APACHE_PREFIX` directories.

```
# chmod -R o-w $APACHE_PREFIX
```

1.3.7 Secure the Core Dump Directory (Scored)

Profile Applicability:

- Level 1

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.

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`)

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 to 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
```

References:

1. http://httpd.apache.org/docs/2.2/mod/mpm_common.html#coredumpdirectory

1.3.8 Secure the Lock File (Scored)

Profile Applicability:

- Level 1

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 mounted 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.

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. Verify the permissions on the directory are 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

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.

References:

1. http://httpd.apache.org/docs/2.2/mod/mpm_common.html#lockfile

1.3.9 Secure the Pid File (Scored)

Profile Applicability:

- Level 1

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.

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. Verify the permissions on the directory are only writable by root (or the startup user if not root).

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).

References:

1. http://httpd.apache.org/docs/2.2/mod/mpm_common.html#pidfile

1.3.10 Secure the ScoreBoard File (Scored)

Profile Applicability:

- Level 1

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 platforms shared memory will be used instead of a file in the file system, so this directive is not generally needed and does not 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.

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.

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.

References:

1. http://httpd.apache.org/docs/2.2/mod/mpm_common.html#scoreboardfile

1.3.11 Restrict Group Write Access for the Apache Directories and Files (Scored)

Profile Applicability:

- Level 1

Description:

Group permissions on Apache directories should generally be r-x and 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 of the web

document root $DOCROOT defined by Apache DocumentRoot and defaults to $APACHE_PREFIX/htdocs. The directories and files in the web document root may have a designated web development group with write access to allow web content to be updated.

Rationale:

Restricting write permissions on the Apache files and directories can help mitigate attacks that modify web content to provide unauthorized access, or to attack web clients.

Audit:

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

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

Remediation:

Perform the following to remove group write access on the $APACHE_PREFIX directories.

```
# chmod -R g-w $APACHE_PREFIX
```

1.3.12 Restrict Group Write Access for the Document Root Directories and Files (Scored)

Profile Applicability:

- Level 1

Description:

Group permissions on Apache Document Root directories $DOCROOT may need to be writeable by an authorized group such as development, support, or a production content management tool. However it is important that the Apache group used to run the server does not have write access to any directories or files in the document root.

Rationale:

Preventing Apache from writing to the web document root helps mitigate risk associated with web application vulnerabilities associated with file uploads or command execution.

Typically, if an application hosted by Apache needs to write to directory, it is best practice to have that directory live outside the web root.

Audit:

Identify files or directories in the Apache Document Root directory with Apache group write access.

```
## Define $GRP to be the Apache group configured
# GRP=$(grep '^Group' $APACHE_PREFIX/conf/httpd.conf | cut -d' ' -f2)

find -L $DOCROOT -group $GRP -perm /g=w -ls
```

Remediation:

Perform the following to remove group write access on the $DOCROOT directories and files with the apache group.

```
# find -L $DOCROOT -group $GRP -perm /g=w -print | xargs chmod g-w
```

1.4 Apache Access Control

Recommendations in this section pertain to configurable access control mechanisms that are available in Apache HTTP server.

1.4.1 Deny Access to OS Root Directory (Scored)

Profile Applicability:

- Level 1

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.

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
```

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>
```

References:

1. http://httpd.apache.org/docs/2.2/mod/core.html#directory
2. http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html

1.4.2 Allow Appropriate Access to Web Content (Not Scored)

Profile Applicability:

- Level 1

Description:

In order to serve Web content, the Apache `Allow` directive will 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.

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
```

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>
```

References:

1. http://httpd.apache.org/docs/2.2/mod/core.html#directory
2. http://httpd.apache.org/docs/2.2/mod/mod_authz_host.html

1.4.3 Restrict OverRide for the OS Root Directory (Scored)

Profile Applicability:

- Level 1

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 filesystem. 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, then it is wise to keep the configuration out of the web server from being placed in `.htaccess` files.

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 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
```

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>
```

References:

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

1.4.4 Restrict OverRide for All Directories (Scored)

Profile Applicability:

- Level 1

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 filesystem. 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, then it is wise to keep the configuration out of the web server from being placed in `.htaccess` files

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
```

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
. . .
```

References:

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

1.5 Minimize Features, Content and Options

Recommendations in this section intend to reduce the effective attack surface of Apache HTTP server.

1.5.1 Restrict Options for the OS Root Directory (Scored)

Profile Applicability:

- Level 1

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`.

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
```

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>
```

References:

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

1.5.2 Restrict Options for the Web Root Directory (Scored)

Profile Applicability:

- Level 1

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.

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
```

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 multiviews are needed.

```
<Directory "/usr/local/apache2/htdocs">
    . . .
    Options None
    . . .
</Directory>
```

References:

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

1.5.3 Minimize Options for Other Directories (Scored)

Profile Applicability:

- Level 1

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.
- `ExecCGI` - Is only appropriate for special directories dedicated to executable content such as a `cgi-bin/` directory. That way you will know what is executed on the server. It is possible to enable CGI script execution based on file extension or permission settings however this makes script control and management almost impossible as developers may install scripts without your knowledge. This may become a factor in a hosting environment.
- `FollowSymLinks` & `SymLinksIfOwnerMatch` - The following of symbolic links is not recommended and should be disabled if possible. 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 in order 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
- `Indexes` - The `Indexes` option causes automatic generation of indexes, if the default index page is missing, and should be disabled unless required.

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
```

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>` 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.

References:

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

1.5.4 Remove Default HTML Content (Scored)

Profile Applicability:

- Level 1

Description:

Apache installations have default content that 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.

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.

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>
```

1.5.5 Remove Default CGI Content printenv (Scored)

Profile Applicability:

- Level 1

Description:

Most Web Servers, including 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. One common default CGI content for apache installations is the script `printenv`. This script will print back to the requester all of the CGI environment variables which includes many server configuration details and system paths.

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 unnecessary 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` script in particular will disclose inappropriate information about the web server including directory paths and detailed version and configuration information.

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

Remediation:

Perform the following to implement the recommended state:

1. Locate cgi-bin files and directories enabled in the Apache configuration via `Script`, `ScriptAlias`, `ScriptAliasMatch`, or `ScriptInterpreterSource` directives.
2. Remove the `printenv` default CGI in cgi-bin directory if it is installed.

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

1.5.6 Remove Default CGI Content test-cgi (Scored)

Profile Applicability:

- Level 1

Description:

Most Web Servers, including 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. A common default CGI content for apache installations is the script `test-cgi`. This script will print back to the requester CGI environment variables which includes many server configuration details.

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 unnecessary 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 `test-cgi`

script in particular will disclose inappropriate information about the web server including directory paths and detailed version and configuration information.

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 `ScriptAliasMatch` other `ScriptInterpreterSource` directives.
2. Ensure the `test-cgi` script is not installed in any configured `cgi-bin` directory.

Remediation:

Perform the following to implement the recommended state:

1. Locate cgi-bin files and directories enabled in the Apache configuration via `Script`, `ScriptAlias`, `ScriptAliasMatch`, or `ScriptInterpreterSource` directives.
2. Remove the `test-cgi` default CGI in cgi-bin directory if it is installed.

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

1.5.7 Limit HTTP Request Methods (Scored)

Profile Applicability:

- Level 1

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.

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.)

Remediation:

Perform the following to implement the recommended state:

1. Locate the Apache configuration files and included configuration files.
2. Search for the directive on the document root directory such as:

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

3. Ensure that the access control order within the directive is allow, deny.

```
Order allow,deny
```

4. Add a directive as shown below within the group of document root directives.

```
# Limit HTTP methods to standard methods. Note: Does not limit TRACE
<LimitExcept GET POST OPTIONS>
    deny from all
</LimitExcept>
```

5. Search for other directives in the Apache configuration files other than the OS root directory, and add the same directives to each. It is very important to understand that the directives are 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>
```

References:

1. http://httpd.apache.org/docs/2.2/mod/core.html#limitexcept
2. http://www.ietf.org/rfc/rfc2616.txt

1.5.8 Disable HTTP TRACE Method (Scored)

Profile Applicability:

- Level 1

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 subjected to abuse and should be disabled.

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`

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
```

References:

1. http://httpd.apache.org/docs/2.2/mod/core.html#traceenable
2. http://www.ietf.org/rfc/rfc2616.txt

1.5.9 Restrict HTTP Protocol Versions (Scored)

Profile Applicability:

- Level 1

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.

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 within the global server context that disallows requests that do not include the HTTP/1.1 header as shown below .

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

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

```
RewriteEngine On
RewriteOptions Inherit
```

Remediation:

Perform the following to implement the recommended state:

1. Load the `mod_rewrite`module for Apache by doing either one of the following:
    1. Build Apache with `mod_rewrite` statically loaded during the build, by adding the `--enable-rewrite` option to the `./configure` script.

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

    2. 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 within the global server context 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.

```
RewriteEngine On
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 section to inherit the main server settings.

```
RewriteEngine On
RewriteOptions Inherit
```

References:

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

1.5.10 Restrict Access to .ht files (Scored)*

Profile Applicability:

- Level 1

Description:

Restrict access to any files beginning with `.ht` using the `FilesMatch` 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.

Audit:

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

1. Verify that a `FilesMatch` directive similar to the one below is present in the Apache configuration and not commented out

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>
```

References:

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

1.5.11 Restrict File Extensions (Scored)

Profile Applicability:

- Level 2

Description:

Restrict access to inappropriate file extensions that are not expected to be a legitimate part of web sites using the `FilesMatch` 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 recommended instead that a white list of the appropriate and expected file extensions for the web server be created, reviewed and restricted with a `FilesMatch` directive.

Audit:

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

1. Verify that the `FilesMatch` 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 `FilesMatch` directive similar to the one in step 4 of the remediation, with an expression that matches the approved file extensions.

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 command 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 `FilesMatch` 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 a `FilesMatch` directive that allows access to those file extensions specifically allowed from the review process in step 2. An example `FilesMatch` 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>
```

References:

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

1.5.12 Deny IP Address Based Requests (Scored)

Profile Applicability:

- Level 2

Description:

The Apache module `mod_rewrite` can be used to disallow access for requests that use an IP address instead of a host name for the URL. Most normal access to the website from browsers and automated software will use a host name, and will therefore include the host name in the HTTP HOST header.

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

Rationale:

A common malware propagation and automated network scanning technique is to use IP addresses rather than host names for web requests, since it's much simpler to automate. By

denying IP based web requests, these automated techniques will be denied access to the website. Of course malicious web scanning techniques continue to evolve, and many are now using hostnames, however denying access to the IP based requests is still a worthwhile defense.

Audit:

Perform the following steps 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 within the global server context that disallows IP based requests by requiring a HTTP HOST header similar to the example shown below.

```
RewriteCond %{HTTP_HOST} !^www\.example\.com [NC]
RewriteCond %{REQUEST_URI} !^/error [NC]
RewriteRule ^.(.*) - [L,F]
```

Remediation:

Perform the following to implement the recommended state:

1. Load the `mod_rewrite` module for Apache by doing either one of the following:
    1. Build Apache with `mod_rewrite` statically loaded during the build, by adding the `--enable-rewrite` option to the `./configure` script.

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

    2. 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 within the global server context with the value of `on` so that the rewrite engine is enabled.

```
RewriteEngine On
```

3. Locate the Apache configuration file such as `httpd.conf` and add the following rewrite condition to match the expected host name of the top server level configuration.

```
RewriteCond %{HTTP_HOST} !^www\.example\.com [NC]
RewriteCond %{REQUEST_URI} !^/error [NC]
RewriteRule ^.(.*) - [L,F]
```

References:

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

1.5.13 Restrict Listen Directive (Scored)

Profile Applicability:

- Level 2

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 zeros 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.

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.

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 zeros 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
```

References:

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

1.5.14 Restrict Browser Frame Options (Scored)

Profile Applicability:

- Level 2

Description:

The Header directive allows server HTTP response headers to be added, replaced or merged. We will use the directive to add an server HTTP response header to tell browsers to restrict all of the web pages from being framed by other web sites.

Rationale:

Using iframes and regular web frames to embed malicious content along with expected web content has been a favored attack vector for attacking web clients for a long time. This can happen when the attacker lures the victim to a malicious web site, which using frames to include the expected content from the legitimate site. The attack can also be performed via XSS (either reflected, DOM or stored XSS) to add the malicious content to the legitimate web site.

To combat this vector, an HTTP Response header, X-Frame-Options, has been introduced that allows a server to specify whether a web page may be loaded in any frame (`DENY`) or those frames that share the pages origin (`SAMEORIGIN`).

Audit:

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

1. Ensure the previous line is present in the Apache configuration and not commented out :

```
# grep -i X-Frame-Options $APACHE_PREFIX/conf/httpd.conf
```

Remediation:

Perform the following to implement the recommended state:

1. Add or modify the `Header` directive for the `X-Frames-Options` header in the Apache configuration to have the condition `always`, an action of `append` and a value of `SAMEORIGIN` or `DENY`, as shown below.

```
Header always append X-Frame-Options SAMEORIGIN
```

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_headers.html#header
2. https://developer.mozilla.org/en/The_X-FRAME-OPTIONS_response_header
3. http://blogs.msdn.com/b/ie/archive/2009/01/27/ie8-security-part-vii-clickjacking-defenses.aspx

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 (Scored)

Profile Applicability:

- Level 1

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.

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.

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 or 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.

References:

1. http://httpd.apache.org/docs/2.2/logs.html
2. http://httpd.apache.org/docs/2.2/mod/core.html#loglevel
3. http://httpd.apache.org/docs/2.2/mod/core.html#errorlog

1.6.2 Configure a Syslog Facility for Error Logging (Scored)

Profile Applicability:

- Level 2

Description:

The `ErrorLog` directive should be configured to send logs to a `syslog` facility so that the logs can be processed and monitored along with the system logs.

Rationale:

It is easy for the web server error logs to be overlooked in the log monitoring process, and yet the application level attacks have become the most common and are extremely important for detecting attacks early, as well as detecting non-malicious problems such as a broken link,or internal errors. By including the Apache error logs with the system logging facility, the application logs are more likely to be included in the established log monitoring process.

Audit:

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

1. Verify that the `ErrorLog` in the Apache server configuration has a value of `syslog:facility` where `facility` can be any of the `syslog` facility values such as `local1`.
2. Verify there is a similar `ErrorLog` directive is either configured or inherited for each virtual host.

Remediation:

Perform the following to implement the recommended state:

1. Add an `ErrorLog` directive if not already configured. Any appropriate `syslog` facility may be used in place of `local1`.

```
ErrorLog "syslog:local1"
```

1. Add a similar `ErrorLog` directive for each virtual host if necessary.

Default Value:

The following is the default configuration:

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

References:

1. http://httpd.apache.org/docs/2.2/logs.html
2. http://httpd.apache.org/docs/2.2/mod/core.html#loglevel
3. http://httpd.apache.org/docs/2.2/mod/core.html#errorlog

1.6.3 Configure the Access Log (Scored)

Profile Applicability:

- Level 1

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.

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.

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.

1.6.4 Log Storage and Rotation (Scored)

Profile Applicability:

- Level 1

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 under a potential 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.

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.

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`:

1. 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 it's own log files ensure that those log files are also included in a similar log rotation.
2. 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.

1.6.5 Apply Applicable Patches (Scored)

Profile Applicability:

- Level 1

Description:

Apply available Apache patches within 1 month of availability.

Rationale:

Obviously 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. It is recommended to use the Apache packages and updates provide by the your Linux platform vendor rather than building from source when possible, in order to minimize the disruption and the work of keeping the software up-to-date.

Audit:

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

1. When Apache was built from source:
    1. 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
    2. 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:
    1. Check for vendor supplied updates on the vendor web site.
    2. If newer versions with security patches more than 1 month old are not installed, then the installation is not sufficiently up-to-date.

Remediation:

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

1. When building from source:
    1. Read release notes and related security patch information
    2. Download latest source and any dependent modules such as `mod_security`.
    3. Build new Apache software according to your build process with the same configuration options.
    4. Install and Test the new software according to your organizations testing process.
    5. Move to production according to your organizations deployment process.
2. When using platform packages
    1. Read release notes and related security patch information.
    2. Download and install latest available Apache package and any dependent software.
    3. Test the new software according to your organizations testing process.
    4. Move to production according to your organizations deployment process.

References:

1. http://httpd.apache.org/security/vulnerabilities_22.html
2.

1.7 Use SSL/TLS

Recommendations in this section pertain to the configuration of SSL/TLS-related aspects of Apache HTTP server.

1.7.1 Install mod_ssl and/or mod_nss (Scored)

Profile Applicability:

- Level 1

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. Keep in mind, however that 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.

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.

Remediation:

Perform either of the following to implement the recommended state:

1. For Apache installations built from the source, use the option `--with-ssl=` to specify the openssl path, and 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 --with-ssl=$OPENSSL_DIR --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
```

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html
2. http://directory.fedoraproject.org/wiki/Mod_nss

1.7.2 Install a Valid Trusted Certificate (Scored)

Profile Applicability:

- Level 1

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.

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 certificates. 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. Visit the OWASP testing SSL web page for additional suggestions: http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29

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 not 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 the common name exactly matches 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.

References:

1. http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29
2. http://httpd.apache.org/docs/2.2/ssl/ssl_faq.html#realcert
3. http://www.openssl.org/docs/HOWTO/certificates.txt

1.7.3 Protect the Servers Private Key (Scored)

Profile Applicability:

- Level 1

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 in 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:

1. Use SSLPassPhraseDialog builtin, - Requires a passphrase to be manually entered.
2. Use SSLPassPhraseDialog |/path/to/program to provide the passphrase.
3. Use SSLPassPhraseDialog exec:/path/to/program to provide the passphrase,
4. Store the private key in clear text so that a passphrase is not required.

Any of the above options 1-4 are acceptable as long as the key and passphrase are protected as described below. Option 1 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 2 and 3 can provide additional security if the programs providing them are secure. Option 4 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.

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 string "`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.

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 do 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` directive , change the ownership and permissions on the server private key to owned by `root:root` with permission 0400.

References:

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

1.7.4 Disable Weak SSL Protocols (Scored)

Profile Applicability:

- Level 1

Description:

The Apache SSLProtocol directive specifies the SSL and TLS protocols allowed. Both the SSLv2 and the SSLv3 protocols should be disabled in this directive as they are out-dated and vulnerable to information disclosure. Only TLS protocols should be enabled.

Rationale:

The SSLv2 and SSLv3 protocols are flawed and shouldn't be used, as they are subject to man-in-the-middle attacks and other cryptographic attacks. The TLSv1 protocols should be used instead, and the newer TLS protocols should be preferred.

The SSLv3 protocol was discovered to be vulnerable to the POODLE attack (Padding Oracle On Downgraded Legacy Encryption) in October 2014. The attack allows decryption and extraction of information from the server's memory. Due to this vulnerability disabling the SSLv3 protocol is highly recommended.

Audit:

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

Verify the `SSLProtocol` directive is present in the Apache server level configuration and every virtual host that is SSL enabled. For each directive verify that either:

- a minus "-SSLv2" and a minus "-SSLv3" are included
- an explicit list of only TLS protocols without any plus (+) or minus (-) symbols

Alternately the SSL protocols 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

Remediation:

Perform the following to implement the recommended state:

Search the Apache configuration files for the SSLProtocol directive; add the directive if not present, or change the value to match one of the following values. The first setting "TLSv1.1 TLS1.2" is preferred when it it acceptable to also disable the TLSv1.0 protocol. See the level 2 recommendation "Disable the TLS v1.0 Protocol" for details.

```
SSLProtocol TLSv1.1 TLSv1.2
```

```
SSLProtocol TLSv1
```

Default Value:

Default value is:
`SSLProtocol all -SSLv2`

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslprotocol
2. http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29
3. https://www.us-cert.gov/ncas/alerts/TA14-290A
4. https://www.openssl.org/~bodo/ssl-poodle.pdf

1.7.5 Restrict Weak SSL Ciphers (Scored)

Profile Applicability:

- Level 1

Description:

Disable weak SSL ciphers using the SSLCipherSuite, and SSLHonorCipherOrder directives. The SSLCipherSuite directive specifies which ciphers are allowed in the negotiation with the client. While the SSLHonorCipherOrder causes the servers preferred ciphers to be used instead of the clients specified preferences.

Rationale:

The SSL/TLS protocols support a large number of encryption ciphers including many weak ciphers that are subject to man-in-the middle attacks and information disclosure. Some implementations even support the NULL cipher which allows a TLS connection without any encryption! Therefore it is critical to ensure the configuration only allows strong ciphers greater than or equal to 128 bit to be negotiated with the client. Stronger 256-bit ciphers should be allowed and preferred. In addition enabling the `SSLHonorCipherOrder` further protects the client from man-in-the-middle downgrade attacks by ensuring the servers preferred ciphers will be used rather than the clients preferences.

In addition, the RC4 ciphers are stream ciphers that are widely used and have even been recommended in previous Apache benchmarks as a means of mitigating attacks based on CBC cipher vulnerabilities. However the RC4 ciphers also have known cryptographic weaknesses and are no longer recommended, and should be disabled. The IETF is working on a new draft proposed standard [4] that would disallow RC4 negotiation for all TLS versions. While the document is not yet an RFC (i.e. it's not a standard yet), It is expect it to become one soon, and the RC4 cipher suites will begin to disappear from options in TLS

deployments. In the meantime, it is important to ensure that RC4-based cipher suites as disabled in the configuration.

Audit:

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

- Verify the SSLCipherSuite directive disables weak ciphers in the Apache server level configuration and every virtual host that is SSL enabled.
- Alternately the SSL ciphers supported can be easily tested by connecting to a running web server with openssl s_client such as shown in https://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29

Remediation:

Perform the following to implement the recommended state:

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

```
SSLHonorCipherOrder On
SSLCipherSuite ALL:!EXP:!NULL:!ADH:!LOW:!SSLv2:!MD5:!RC4
```

FIPS Compliance:The above cipher suite specification may be used for servers that fall under FIPS 140-2 compliance requirements, SP800-52 provides guidelines for the TLS ciphers, because it eliminates the usage of the RC4 cipher and MD5 hash which are not deemed FIPS compliant.

Disable SSLv3 Ciphers: If the SSLv3 protocol has also been disabled as recommended, then the SSLv3 related ciphers will not be used, and could be removed from the cipher suite specification.

```
SSLCipherSuite ALL:!EXP:!NULL:!ADH:!LOW:!SSLv2:!SSLv3:!MD5:!RC4
```

Default Value:

The following are the default values:

```
SSLCipherSuite default depends on OpenSSL version.
SSLHonorCipherOrder Off
```

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslciphersuite
2. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslhonorcipherorder
3. http://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29
4. https://datatracker.ietf.org/doc/draft-ietf-tls-prohibiting-rc4/

1.7.6 Restrict Insecure SSL Renegotiation (Scored)

Profile Applicability:

- Level 1

Description:

There was a man-in-the-middle renegotiation attack discovered in SSLv3 and TLSv1 in Nov 2009 (CVE-2009-3555). 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 was approved as an Internet Standard as RFC 574, Feb 2010. The work around which removes the renegotiation is available from OpenSSL as of version 0.9.8l and newer versions. For details: http://www.openssl.org/news/secadv_20091111.txt

The `SSLInsecureRenegotiation` directive was 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 warrants that servers and clients be upgraded to support the improved SSL/TLS protocols. Therefore the recommendation is to not enable the insecure renegotiation.

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`.

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
```

Default Value:

The default value is off:

```
SSLInsecureRenegotiation off
```

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslinsecurerenegotiation
2. http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555

1.7.7 Ensure SSL Compression is Not Enabled (Scored)

Profile Applicability:

- Level 1

Description:

The SSLCompression directive controls whether SSL compression is used by Apache when serving content over HTTPS. It is recommended that the SSLCompression directive be set to off.

Rationale:

if SSL compression is enabled, HTTPS communication between the client and the server may be at increased risk to the CRIME attack. The CRIME attack increases a malicious actor's ability to derive the value of a session cookie, which commonly contains an authenticator. If the authenticator in a session cookie is derived, it can be used to impersonate the account associated with the authenticator.

Audit:

For Apache 2.2.26 and later, perform the following steps to determine if the recommended state is implemented:

1. Search the Apache configuration files for the SSLCompression directive.
2. Verify that the directive either does not exist or exists and is set to off.

For Apache 2.2.24 and 2.2.25 perform the following steps to determine if the recommended state is implemented:

1. Search the Apache configuration files for the SSLCompression directive.
2. Verify that the directive exists and is set to off. (The default value is on)

Apache versions prior to 2.2.24 do not support disabling SSL compression and are not compliant.

Remediation:

Perform the following to implement the recommended state:

1. Verify the Apache version is 2.2.24 or later, with the command "`httpd -v`".
2. Search the Apache configuration files for the SSLCompression directive.
3. Add or update the directive to have a value of `off`.

Default Value:

The SSLCompression directive was available in httpd 2.2.24 and later, if using OpenSSL 0.9.8 or later; virtual host scope is available if using OpenSSL 1.0.0 or later. The default used to be `ON` in versions 2.2.24 to 2.2.25, and is `OFF` for 2.2.26 and later.

References:

1. http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslcompression
2. http://en.wikipedia.org/wiki/CRIME_(security_exploit)

1.7.8 Disable the TLS v1.0 Protocol (Scored)

Profile Applicability:

- Level 2

Description:

The TLSv1.0 protocol should be disabled via the SSLProtocol directive, if possible, as it has been shown to be vulnerable to information disclosure.

Rationale:

The TLSv1.0 protocol is vulnerable to the BEAST attack when used in CBC mode (October 2011). Unfortunately the TLSv1.0 uses CBC modes for all of the block mode ciphers, which only leaves the RC4 streaming cipher. The RC4 cipher is not vulnerable to the BEAST attack; however there is research that indicates it is also weak and is not recommended. Therefore it is recommended that the TLSv1.0 protocol be disabled if all TLS clients support the newer TLS protocols. All major up-to-date browsers support TLSv1.1 and TLSv1.2; however, some older IE browsers (8,9,10) may still have TLSv1.1 and TLSv1.2 disabled for some strange reason. While Safari 6 does not support the newer TLS protocols. Review the Wikipedia reference for browser support details. Ensuring that all user's browsers are configured to allow TLSv1.1 and TLS1.2 is necessary before disabling TLSv1.0 on the Apache web server; therefore this recommendation is a level 2 rather than a level 1. Disabling TLSv1.0 on internal only websites is more easily accomplished when access is limited to clients with browsers controlled by the organization policies and procedures to allow and prefer TLSv1.1 and higher.

The NIST SP 800-52r1 guidelines for TLS configuration state that servers that support government-only applications shall not support TLSv1.0 or any of the SSL protocols. While Servers that support citizen or business-facing applications may be configured to support TLS version 1.0 in order to enable interaction with citizens and businesses.
Also it is important to note that Microsoft support for all older versions of IE ends January 12, 2016, and Apple ends support for Safari 6 with the fall release if OS X 10.11. So it is wise to plan for usage of TLSv1.0 to be eliminated in 2016.

Some organizations may find it helpful to implement a phased transitional plan where TLS1.0 is not disabled, but the web server will detect browsers which do not have TLS1.1 or newer enabled and redirect them to a web site that explains how to enabled the newer TLS protocols. The redirect can be implemented using the mod_rewrite which can detect the protocol used, and rewrite the URL to the helpful website.

Audit:

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

Search the Apache configuration files for the SSLProtocol directive and ensure it has the value of "`TLSv1.1 TLS1.2`".

Remediation:

Perform the following to implement the recommended state:

Search the Apache configuration files for the SSLProtocol directive; add the directive if not present, or change the value to "`TLSv1.1 TLS1.2`".

Default Value:

The default value is:

```
SSLProtocol all -SSLv2
```

References:

1. http://en.wikipedia.org/wiki/Transport_Layer_Security#Web_browsers- Browser support and defaults for SSL/TLS protocols
2. https://community.qualys.com/blogs/securitylabs/2013/09/10/is-beast-still-a-threat
3. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf
4. https://support.microsoft.com/en-us/gp/microsoft-internet-explorer

1.7.9 Enable HTTP Strict Transport Security (Scored)

Profile Applicability:

- Level 2

Description:

HTTP Strict Transport Security (HSTS) is an optional web server security policy mechanism specified by an HTTP Server header. The HSTS header allows a server declaration that only HTTPS communication should be used rather than clear text HTTP communication.

Rationale:

Usage of HTTP Strict Transport Security (HSTS) helps protect HSTS compliant browsers and other agents from HTTP downgrade attacks. Downgrade attacks include a variety of man-in-the-middle attacks which leave the web communication vulnerable to disclosure and modification by forcing the usage of HTTP rather than HTTPS communication. The `sslstrip` attack tool by Moxie Marlinspike released in 2009 is one such attack, which works when the server allows both HTTP and HTTPS communication. However a man-in-the-middle HTTP-to-HTTPS proxy would be effective in cases where the server required

HTTPS, but did not publish an HSTS policy to the browser. This attack would also be effective on browsers which were not compliant with HSTS. All current up-to-date browsers support HSTS except Microsoft's Internet Explorer; Internet Explorer is expected to support HSTS in the next major release **after** IE 12.

The HSTS header specifies a length of time in seconds that the browser / user agent should access the server only using HTTPS. The header may also specify if all sub-domains should also be included in the same policy. Once a compliant browser receives the HSTS Header it will not allow access to the server via HTTP. Therefore it is important that you ensure that there is no portion of the web site or web application that requires HTTP prior to enabling the HSTS protocol.

If all sub-domains are to be included via the *includeSubDomains* option, then carefully consider all various host names, web applications and third party services used including any DNS CNAME values that may be impacted. An overly broad *includeSubDomains* policy will disable access to HTTP web sites for all websites with the same domain name. Also consider that the access will be disabled for the number of seconds given in the max-age value, so in the event a mistake is made, a large value, such as a year, could create significant support issues.

An optional flag of preload may be added if the web site name is to be submitted to be preloaded in Chrome, Firefox and Safari browsers. See https://hstspreload.appspot.com/ for details.

Audit:

Perform either of the following steps to determine if the recommended state is implemented.

At the Apache server level configuration and for every virtual host that is SSL enabled, verify there is a Header directive present that sets the Strict-Transport-Security header with a max-age value of at least 480 seconds or more (8 minutes or more). For example:

```
Header always set Strict-Transport-Security "max-age=600"
```

As an alternative the configuration may be validated by connecting to the HTTPS server and verifying the presence of the header. Such as the `openssl s_client` command shown below:

```
openssl s_client -connect www.example.com:443
GET / HTTP1.1.
Host:www.example.com
```

```
HTTP/1.1 200 OK
Date: Mon, 08 Dec 2014 18:28:29 GMT
Server: Apache
X-Frame-Options: NONE
Strict-Transport-Security: max-age=600
Last-Modified: Mon, 19 Jun 2006 14:47:16 GMT
ETag: "152-41694d7a92500"
Accept-Ranges: bytes
Content-Length: 438
Connection: close
Content-Type: text/html
```

Remediation:

Perform the following to implement the recommended state:

Add a Header directive as shown below in the Apache server level configuration and every virtual host that is SSL enabled. The `includeSubDomains` and `preload` flags may be included in the header, but are not required.

```
Header always set Strict-Transport-Security "max-age=600”; includeSubDomains; preload
```

- or -

```
Header always set Strict-Transport-Security "max-age=600”
```

Default Value:

Default Value:

The Strict Transport Security header is not present by default.

References:

1. http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
2. https://www.owasp.org/index.php/HTTP_Strict_Transport_Security
3. http://www.thoughtcrime.org/software/sslstrip/
4. https://developer.mozilla.org/en-US/docs/Web/Security/HTTP_strict_transport_security
5. https://hstspreload.appspot.com/

1.8 Information Leakage

Recommendations in this section are intended to limit the disclosure of potentially sensitive information.

1.8.1 Set ServerToken to 'Prod' (Scored)

Profile Applicability:

- Level 1

Description:

Configure the Apache `ServerTokens` directive to provide minimal information. 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 detailes 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.

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`.

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
```

References:

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

1.8.2 Set ServerSignature to 'Off' (Scored)

Profile Applicability:

- Level 1

Description:

Disable the server signatures which generates 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.

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`:

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
```

References:

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

1.8.3 Information Leakage via Default Apache Content (Scored)

Profile Applicability:

- Level 2

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 not 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 primarily for auto indexing, which is recommended to be disabled.

Audit:

Perform the following step 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.

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>
```

1.9 Denial of Service Mitigations

Denial of Service (DoS) attacks intend to degrade a service's ability to process and respond to service requests. Typically, DoS attacks attempt to exhaust the service's network-, CPU-, disk-, and/or memory- related resources. Configuration states in this section may increase a server's resiliency to DoS attacks.

1.9.1 Set the TimeOut to 10 or less (Scored)

Profile Applicability:

- Level 1

Description:

The `TimeOut` directive controls the maximum time in seconds that Apache HTTP server will wait for an Input/Output call to complete. It is recommended that the `TimeOut` directive be set to 10 or less.

Rationale:

One common technique for DoS is to initiate many connections to the server. By decreasing the timeout for old connections, the server can free resources more quickly and be more responsive. By making the server more efficient, it will be more resilient to DoS conditions.

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/. Upgrading to Apache 2.4 is recommended.

Audit:

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

Verify that the `Timeout` directive is specified in the Apache configuration files to have a value of 10 seconds or shorter.

Remediation:

Perform the following to implement the recommended state:

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

```
Timeout 10
```

References:

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

1.9.2 Set the KeepAlive to On (Scored)

Profile Applicability:

- Level 1

Description:

The `KeepAlive` directive controls whether Apache will reuse the same TCP connection per client to process subsequent HTTP requests from that client. It is recommended that the `KeepAlive` directive be set to `On`.

Rationale:

Allowing per-client reuse of TCP sockets reduces the amount of system and network resources required to serve requests. This efficiency gain may improve a server's resiliency to DoS attacks.

Audit:

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

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`.

Remediation:

Perform the following to implement the recommended state:

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

```
KeepAlive On
```

References:

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

1.9.3 Set the MaxKeepAliveRequests to 100 or greater (Scored)

Profile Applicability:

- Level 1

Description:

The `MaxKeepAliveRequests` directive limits the number of requests allowed per connection when `KeepAlive` is on. If it is set to 0, unlimited requests will be allowed. It is recommended that the `MaxKeepAliveRequests` directive be set to 100 or greater.

Rationale:

Allowing per-client reuse of TCP sockets reduces the amount of system and network resources required to serve requests. This efficiency gain may improve a server's resiliency to DoS attacks.

Audit:

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

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.

Remediation:

Perform the following to implement the recommended state:

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

```
MaxKeepAliveRequests 100
```

References:

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

1.9.4 Set the KeepAliveTimeout to 15 or less (Scored)

Profile Applicability:

- Level 1

Description:

The `KeepAliveTimeout` directive specifies the number of seconds Apache will wait for a subsequent request before closing a connection that is being kept alive.

Rationale:

Reducing the number of seconds that Apache HTTP server will keep unused resources allocated for will increase the availability of resources to serve other requests. This efficiency gain may improve a server's resiliency to DoS attacks.

Audit:

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

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.

Remediation:

Perform the following to implement the recommended state:

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

```
KeepAliveTimeout 15
```

References:

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

1.9.5 Set Timeout Limits for Request Headers (Scored)

Profile Applicability:

- Level 1

Description:

The `RequestReadTimeout` directive allows configuration of timeout limits for client requests. The header portion of the directive provides for an initial timeout value, a maximum timeout and a minimum rate. The minimum rate specifies that after the initial timeout, the server will wait an additional 1 second for each N bytes received. The recommended setting is to have a maximum timeout of 40 seconds or less. Keep in mind that for SSL/TLS virtual hosts the time for the TLS handshake must fit within the timeout.

Rationale:

Setting a request header timeout is vital for mitigating Denial of Service attacks based on slow requests. The slow request attacks are particularly lethal and relative easy to perform, because they require very little bandwidth and can easily be done through anonymous proxies. Starting in June 2009 with the Slow Loris DoS attack, which used a slow GET request, was published by Robert Hansen (RSnake) on his blog http://ha.ckers.org/slowloris/. Later in November 2010 at the OWASP App Sec DC conference Wong Onn Chee demonstrated a slow POST request attack which was even more effective. See https://www.owasp.org/index.php/H.....t.....t....p.....p....o....s....t for details.

Audit:

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

1. Locate the Apache configuration files and included configuration files.
2. Locate any `RequestReadTimeout` directives and verify that they have a maximum header request timeout of 40 seconds or less.
3. If the configuration does not contain any `RequestReadTimeout` directives, and the mod_reqtimeout module is being loaded, then the default value of 40 seconds is compliant with the benchmark recommendation.

```
RequestReadTimeout header=XXX-40,MinRate=XXX body=XXXXXXXXXX
```

Remediation:

- Load the `mod_requesttimeout` module in the Apache configuration with the following configuration.

```
LoadModule reqtimeout_module modules/mod_reqtimeout.so
```

- Add a `RequestReadTimeout` directive similar to the one below with the maximum request header timeout value of 40 seconds or less.

```
RequestReadTimeout header=20-40,MinRate=500 body=20,MinRate=500
```

Default Value:

Default values are:
`header=20-40,MinRate=500`

References:

1. http://ha.ckers.org/slowloris/
2. https://www.owasp.org/index.php/H.....t.....t....p.......p....o....s....t
3. http://httpd.apache.org/docs/2.2/mod/mod_reqtimeout.html

1.9.6 Set Timeout Limits for the Request Body (Scored)

Profile Applicability:

- Level 1

Description:

The `RequestReadTimeout` directive also allows setting timeout values for the body portion of a request. The directive provides for an initial timeout value, and a maximum timeout and minimum rate. The minimum rate specifies that after the initial timeout, the server will wait an additional 1 second for each N bytes received. The recommended setting is to have a maximum timeout of 20 seconds or less. The default value is body=20,MinRate=500.

Rationale:

It is not sufficient to timeout only on the header portion of the request, as the server will still be vulnerable to attacks like the OWASP Slow POST attack, which provide the body of the request very slowly. Therefore the body portion of the request must have a timeout as well. A timeout of 20 seconds or less is recommended.

Audit:

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

- Locate the Apache configuration files and included configuration files.
- Locate any `RequestReadTimeout` directives and verify the configuration has a maximum body request timeout of 20 seconds or less.
- If the configuration does not contain any `RequestReadTimeout` directives, and the mod_reqtimeout module is being loaded, then the default value of 20 seconds is compliant with the benchmark recommendation.

```
RequestReadTimeout header=XXXXXX body=20,MinRate=XXXXXXXXXX
```

Remediation:

Load the mod_requesttimeout module in the Apache configuration with the following configuration.

```
LoadModule reqtimeout_module modules/mod_reqtimeout.so
```

Add a `RequestReadTimeout` directive similar to the one below with the maximum request body timeout value of 20 seconds or less.

```
RequestReadTimeout header=20-40,MinRate=500 body=20,MinRate=500
```

Default Value:

The default value is:

```
body=20,MinRate=500
```

References:

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

1.10 Request Limits

Recommendations in this section reduce the maximum allowed size of request parameters. Doing so increases the likelihood of negatively impacting application and/or site functionality. It is highly recommended that the configuration states described in this section be tested on test servers prior deploying them to production servers.

1.10.1 Set the LimitRequestLine directive to 512 or less (Scored)

Profile Applicability:

- Level 2

Description:

The `LimitRequestLine` directive sets the maximum number of bytes that Apache will read for each line of an HTTP request. It is recommended that the `LimitRequestLine` be set to 512 or less.

Rationale:

Limiting request line size may reduce the exposure of a buffer-related vulnerability potentially present in a code base hosted by Apache HTTP server.

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.

Remediation:

Perform the following to implement the recommended state:

1. Add or modify the `LimitRequestLine` directive in the Apache configuration to have a value of 512 or shorter.

```
LimitRequestLine 512
```

References:

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

1.10.2 Ensure the LimitRequestFields directive is set to 100 or less (Scored)

Profile Applicability:

- Level 2

Description:

The `LimitRequestFields` directive sets the maximum limit on the number of HTTP request headers allowed per request. It is recommended that the `LimitRequestFields` directive be set to 100 or less.

Rationale:

Limiting the number of headers per request may reduce the exposure of a buffer-related vulnerability potentially present in a code base hosted by Apache HTTP server.

Audit:

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

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

Remediation:

Perform the following to implement the recommended state:

1. 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
```

References:

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

1.10.3 Set the LimitRequestFieldsize directive to 1024 or less (Scored)

Profile Applicability:

- Level 2

Description:

The `LimitRequestFieldSize` directive sets the maximum size of an HTTP request header field. It is recommended that the `LimitRequestFieldSize` directive be set to 1024 or less.

Rationale:

Limiting header field size may reduce the exposure of a buffer-related vulnerability potentially present in a code base hosted by Apache HTTP server.

Audit:

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

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

Remediation:

Perform the following to implement the recommended state:

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

References:

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

1.10.4 Set the LimitRequestBody directive to 102400 or less (Scored)

Profile Applicability:

- Level 2

Description:

The `LimitRequestBody` directive sets the maximum size of an HTTP request body. It is recommended that the `LimitRequestBody` directive be set to 102400 or less.

Rationale:

Limiting request body size may reduce the exposure of a buffer-related vulnerability potentially present in a code base hosted by Apache HTTP server.

Audit:

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

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

Remediation:

Perform the following to implement the recommended state:

1. 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.
2. `LimitRequestBody 102400`

References:

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

Appendix: Change History

Date	Version	Changes for this version
09-28-2012	3.2.0	Move items 1.9.2 and 1.9.1 in to section 1.5 - Ticket #68
09-28-2012	3.2.0	1.6.6 Removed Red Hat references - Ticket #57
09-28-2012	3.2.0	1.9.1 DoS Mitigation - Broke into section distinct recommendations per directive - Ticket #58
09-28-2012	3.2.0	1.9.2 Buffer Overflow Mitigations - Broke into section with distinct recommendations per directive - Ticket #60
09-28-2012	3.2.0	1.2.1 Set to not scored
01-28-2015	3.3.0	Ticket #85: POODLE and BEAST mitigation
01-28-2015	3.3.0	Ticket #82: Error in item 1.4.2
01-28-2015	3.3.0	Ticket #72: Fix missing quotation mark
01-28-2015	3.3.0	Ticket #103: Added two recommendations for Request Header and Body
01-28-2015	3.3.0	Ticket #88: Disallow RC4 cipher suites
01-28-2015	3.3.0	Ticket #89: Recommend disabling SSL compression
01-28-2015	3.3.0	Ticket #90: HTTP Strict Transport Security Header
01-28-2015	3.3.0	Ticket #102: Added recommendation for syslog facility
01-28-2015	3.3.0	Ticket #101: Split Apache directory and file

		ownership
01-28-2015	3.3.0	Ticket #100: Split "Enable HTTP Strict Transport Security" in two
01-28-2015	3.3.0	Ticket #92: Removed socket exception from find command
04-23-2015	3.3.1	Informational update to 1.7.8 Disable the TLS v1.0 Protocol
04-23-2015	3.3.1	Informational update to 1.7.9 Enable HTTP Strict Transport Security