CIS Benchmarks

CIS Apache HTTP Server 2.4 Benchmark

v2.0.0 - 10-15-2020

Terms of Use

Please see the below link for our current terms of use:

https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/

Table of Contents

Overview

This document, CIS Apache 2.4 Benchmark, provides prescriptive guidance for establishing a secure configuration posture for Apache Web Server versions 2.4 running on Linux. This guide was tested against Apache Web Server 2.4.3 - 2.4.6 as built from source `httpd-2.4.x.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.

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate Apache HTTP Server 2.4 running on Linux.

Consensus Guidance

This benchmark was created using a consensus review process comprised of 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://workbench.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

Assessment Status

An assessment status is included for every recommendation. The assessment status indicates whether the given recommendation can be automated or requires manual steps to implement. Both statuses are equally important and are determined and supported as defined below:

Automated

Represents recommendations for which assessment of a technical control can be fully automated and validated to a pass/fail state. Recommendations will include the necessary information to implement automation.

Manual

Represents recommendations for which assessment of a technical control cannot be fully automated and requires all or some manual steps to validate that the configured state is set as expected. The expected state can vary depending on the environment.

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 GXPN, CISSP, GSEC, GCIH, GSNA, GPEN, C|EH, Durkee Consulting, Inc.

Editor
Tim Harrison, Center for Internet Security

Recommendations

1 Planning and Installation

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

1.1 Ensure the Pre-Installation Planning Checklist Has Been Implemented (Manual)

Profile Applicability:

- Level 1
- Level 2

Description:

Review and implement the following items as appropriate:

- Reviewed and implemented 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.
- Educate developers, architects and testers about developing secure applications, and integrate security into the software development lifecycle. https://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.

References:

1. Open Web Application Security Project - https://www.OWASP.org
2. Web Application Security Consortium - http://www.webappsec.org/

1.2 Ensure the Server Is Not a Multi-Use System (Manual)

Profile Applicability:

- Level 1
- 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 documented 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
```

Default Value:

Depends on OS Platform

CIS Controls:

Version 6

9.5 Operate Critical Services On Dedicated Hosts (i.e. DNS, Mail, Web, Database)

Operate critical services on separate physical or logical host machines, such as DNS, file, mail, web, and database servers.

Version 7

2.10 Physically or Logically Segregate High Risk Applications

Physically or logically segregated systems should be used to isolate and run software that is required for business operations but incur higher risk for the organization.

1.3 Ensure Apache Is Installed From the Appropriate Binaries (Manual)

Profile Applicability:

- Level 1
- Level 2

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 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. For example, many OS vendors ship Apache with `mod_ssl` and OpenSSL, PHP, `mod_perl`, and `ModSecurity`.
- Your vendor will tell you about security issues so you have to look in fewer places.
- Updates to fix security issues will be easy to apply. The vendor will have already verified the problem, checked the signature on the Apache download, worked out the impact and so on.
- You may be able to get the updates automatically, reducing the window of risk.

Audit:

Remediation:

Installation depends on the operating system platform. For a source build, consult the Apache 2.4 documentation on compiling and installing https://httpd.apache.org/docs/2.4/install.html for a Red Hat Enterprise Linux 5 or 6, the following `yum` command could be used.

```
# yum install httpd
```

References:

1. Apache Compiling and Installation https://httpd.apache.org/docs/2.4/install.html

CIS Controls:

Version 6

2 Inventory of Authorized and Unauthorized Software
Inventory of Authorized and Unauthorized Software

Version 7

2.1 Maintain Inventory of Authorized Software
Maintain an up-to-date list of all authorized software that is required in the enterprise for any business purpose on any business system.

2.2 Ensure Software is Supported by Vendor
Ensure that only software applications or operating systems currently supported by the software's vendor are added to the organization's authorized software inventory. Unsupported software should be tagged as unsupported in the inventory system.

2 Minimize Apache Modules

It's crucial to have a minimal and compact Apache installation based on documented business requirements. 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 purposes not be limited to the modules explicitly listed.

2.1 Ensure Only Necessary Authentication and Authorization Modules Are Enabled (Manual)

Profile Applicability:

- Level 1
- Level 2

Description:

The Apache 2.4 modules for authentication and authorization are grouped and named to provide both granularity and a consistent naming convention to simplify configuration. The `authn_*` modules provide authentication, while the `authz_*` modules provide authorization. Apache provides two types of authentication - basic and digest. Review the Apache Authentication and Authorization how-to documentation http://httpd.apache.org/docs/2.4/howto/auth.html and enable only the modules that are required.

Rationale:

Authentication and authorization are the front doors to the protected information in your web site. Most installations only need a small subset of the modules available. By minimizing the enabled modules to those that are actually used, we reduce the number of "doors" and 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 of 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. http://httpd.apache.org/docs/2.4/mod/ The unnecessary static compiled modules are disabled through compile time configuration options as documented in http://httpd.apache.org/docs/2.4/programs/configure.html. The dynamically loaded modules are disabled by commenting out or removing the `LoadModule` directive from the Apache configuration files (typically `httpd.conf`). Some modules may be separate packages, and may be removed.

Default Value:

The following modules are loaded by a default source build:

- `authn_file_module (shared)`
- `authn_core_module (shared)`
- `authz_host_module (shared)`
- `authz_groupfile_module (shared)`
- `authz_user_module (shared)`
- `authz_core_module (shared)`

References:

1. https://httpd.apache.org/docs/2.4/howto/auth.html
2. https://httpd.apache.org/docs/2.4/mod/
3. https://httpd.apache.org/docs/2.4/programs/configure.html

CIS Controls:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

Version 7

16.1 Maintain an Inventory of Authentication Systems

Maintain an inventory of each of the organization's authentication systems, including those located onsite or at a remote service provider.

2.2 Ensure the Log Config Module Is Enabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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. This module is required to configure web server logging using the `log_format` directive.

Audit:

Perform the following to determine if the `log_config` has been loaded:
Use the `httpd -M` option as `root` to check that 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
$ ./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
```

Default Value:

The `log_config` module is loaded by default.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_log_config.html

CIS Controls:

Version 6

6.2 Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

6.3 Enable Detailed Logging

Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

2.3 Ensure the WebDAV Modules Are Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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:

Disabling WebDAV modules will improve the security posture of the web server by reducing the amount of potentially vulnerable code paths exposed to the network and reducing potential for unauthorized access to files via misconfigured WebDAV access controls.

Audit:

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

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, there will be no output 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
$ ./configure
```

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

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

Default Value:

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

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_dav.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services
Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

2.4 Ensure the Status Module Is Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

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

Rationale:

When `mod_status` is loaded into the server, its handler capability is available in all configuration files, including per-directory files (e.g., `.htaccess`). The `mod_status` module may provide an adversary with information that can be used to refine exploits that depend on measuring server load.

Audit:

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

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

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

Note: If the modules are correctly disabled, there will be no output 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
$ ./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
```

Default Value:

The `mod_status` module IS enabled with a default source build.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_status.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services
Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

2.5 Ensure the Autoindex Module Is Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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 be 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. Directory listings may also reveal files that were not intended to be revealed.

Audit:

Perform the following to determine if the module is enabled.

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

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

Note: If the module is correctly disabled, there will be no output 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
$ ./configure -disable-autoindex
```

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

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

Default Value:

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

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_autoindex.html

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

2.6 Ensure the Proxy Modules Are Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

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

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

Note: If the modules are correctly disabled, there will be no output 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
$ ./configure
```

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

```
##LoadModule proxy_module modules/mod_proxy.so
##LoadModule proxy_connect_module modules/mod_proxy_connect.so
##LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
##LoadModule proxy_http_module modules/mod_proxy_http.so
##LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
##LoadModule proxy_scgi_module modules/mod_proxy_scgi.so
##LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
##LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
##LoadModule proxy_express_module modules/mod_proxy_express.so
##LoadModule proxy_wstunnel_module modules/mod_proxy_wstunnel.so
##LoadModule proxy_fdpass_module modules/mod_proxy_fdpass.so
```

Default Value:

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

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_proxy.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

2.7 Ensure the User Directories Module Is Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

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

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

Note: If the modules are correctly disabled, there will be no output 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
$ ./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
```

Default Value:

The `mod_userdir` module is not enabled with a default source build.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_userdir.html

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

2.8 Ensure the Info Module Is Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

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

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

Note: If the module is correctly disabled, there will be no output 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
$ ./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
```

Default Value:

The `mod_info` module is not enabled with a default source build.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_info.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services
Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

2.9 Ensure the Basic and Digest Authentication Modules are Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The Apache `mod_auth_basic` and `mod_auth_digest` modules support HTTP Basic Authentication and HTTP Digest Authentication respectively. The two authentication protocols are used to restrict access to users who provide a valid user name and password.

Rationale:

Neither HTTP Basic nor HTTP Digest authentication should be used as the protocols are out dated and no longer considered secure. Disabling the modules will improve the security posture of the web server by reducing the amount of potentially vulnerable code paths exposed to the network and reducing potential for unauthorized access to files via misconfigured access controls.

In the early days of the web, Basic HTTP Authentication was considered adequate if it was only used over HTTPS, so that the credentials would not be sent in the clear. Basic authentication uses Base64 to encode the credentials which are sent with every request. Base64 encoding is of course easily reversed, and is no more secure than clear text. The issues with using Basic Auth over HTTPS is that it does not meet current security standards for protecting the login credentials and protecting the authenticated session. The following security issues plague the Basic Authentication protocol.

- The authenticated session has an indefinite length (as long as any browser window is open) and is not timed-out on the server when the session is idle.
- Application logout is required to invalidate the session on the server to limit, but in the case of Basic Authentication, there is no server-side session that can be invalidated.
- The credentials are remembered by the browser and stored in memory.
- There is no way to disable auto-complete, where the browser offers to store the passwords. Passwords stored in the browser can be accessed if the client system or browser become compromised.
- The credentials are more likely to be exposed since they are automatically sent with every request.

- Administrators may at times have access to the HTTP headers sent in request for the purposes of diagnosing problems and detecting attacks. Having a user's credentials in the clear in the HTTP headers, may allow a user to repudiate actions performed, because the web or system administrators also had access to the user's password.

The HTTP Digest Authentication is considered even worse than Basic Authentication because it stores the password in the clear on the server, and has the same session management issues as Basic Authentication.

Audit:

Perform the following to determine if the HTTP Basic or HTTP Digest authentication modules are enabled.

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

```
# httpd -M | grep auth_basic_module
# httpd -M | grep auth_digest_module
```

Note: If the modules are correctly disabled, there will be no output when executing either of the above commands.

Remediation:

Perform either one of the following to disable the HTTP Basic or HTTP Digest authentication modules:

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

```
$ cd $DOWNLOAD_HTTPD
$ ./configure
```

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

```
##LoadModule mod_auth_basic modules/mod_auth_basic.so
##LoadModule mod_auth_digest modules/mod_auth_digest.so
```

Default Value:

The `mod_auth_basic` and `mod_auth_digest` modules are not enabled with a default source build.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_auth_basic.html
2. https://httpd.apache.org/docs/2.4/mod/mod_auth_digest.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

3 Principles, Permissions, and Ownership

This section provides recommendations for configuring identities (users and groups) that Apache leverages, permissions on Apache-related filesystem resources, and ownership of Apache-related filesystem resources.

3.1 Ensure the Apache Web Server Runs As a Non-Root User (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

Although Apache is typically 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 that the Apache worker processes will assume.

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 identifier used for the apache user should be a unique system account. System user accounts UID numbers have lower values which 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. Typically, system accounts numbers range from `1-999`, or `1-499` and are defined in the `/etc/login.defs` file.

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 less than the minimum normal user account with the Apache group and configured in the `httpd.conf` file.

1. Ensure the User and Group directives are present in the Apache configuration and not commented out:

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

2. Ensure the Apache account UID is correct:

```
# grep '^UID_MIN' /etc/login.defs
# id apache
```

   The UID must be less than the `UID_MIN` value in `/etc/login.defs`, and group of `apache` similar to the following entries:

```
UID_MIN                  1000
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
```

Default Value:

The default Apache user and group are configured as `daemon`.

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

Version 7

4.3 Ensure the Use of Dedicated Administrative Accounts

Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.

3.2 Ensure the Apache User Account Has an Invalid Shell (Automated)

Profile Applicability:

- Level 1
- Level 2

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
```

Default Value:

The default Apache user account is `daemon`. The daemon account may have a valid login shell or a shell of `/sbin/nologin` depending on the operating system distribution version.

CIS Controls:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

4.3 Ensure the Use of Dedicated Administrative Accounts

Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.

3.3 Ensure the Apache User Account Is Locked (Automated)

Profile Applicability:

- Level 1
- Level 2

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.)

- or -

apache L 07/02/2012 -1 -1 -1 -1
```

Remediation:

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

```
# passwd -l apache
```

Default Value:

The default user is `daemon` and the account is typically locked.

CIS Controls:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

Version 7

16.8 Disable Any Unassociated Accounts
Disable any account that cannot be associated with a business process or business owner.

3.4 Ensure Apache Directories and Files Are Owned By Root (Automated)

Profile Applicability:

- Level 1
- Level 2

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges
Minimize administrative privileges and only use administrative accounts when they are

required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.5 Ensure the Group Is Set Correctly on Apache Directories and Files (Automated)

Profile Applicability:

- Level 1
- Level 2

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

CIS Controls:

Version 6

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

Version 7

14.6 Protect Information through Access Control Lists
Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.6 Ensure Other Write Access on Apache Directories and Files Is Restricted (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

Permissions on Apache directories should generally be `rwxr-xr-x` (755) and file permissions should be similar except not executable unless appropriate. This applies to all of the Apache software directories and files installed with the possible exception of the web document root `$APACHE_PREFIX/htdocs`. The directories and files in the web document root may have a designated group with write access to allow web content to be updated. In summary, the minimum recommendation is to not allow write access 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
```

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the

principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.7 Ensure the Core Dump Directory Is Secured (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The `CoreDumpDirectory` directive is used to specify the directory Apache attempts to switch to before creating the core dump. Core dumps will be disabled if the directory is not writable by the Apache user. Also, core dumps will be disabled if the server is started as `root` and switches to a non-root user, as is typical. It is recommended that the `CoreDumpDirectory` directive be set to a directory that is owned by the `root` user, owned by the group the Apache HTTPD process executes as, and be inaccessible to other users.

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 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
```

Default Value:

The default core dump directory is the `ServerRoot` directory.

References:

1. https://httpd.apache.org/docs/2.4/mod/mpm_common.html#coredumpdirectory

CIS Controls:

Version 6

18.9 Sanitize Deployed Software Of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.8 Ensure the Lock File Is Secured (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The `Mutex` directive sets the locking mechanism used to serialize access to resources. It may be used to specify that a lock file is to be used as a mutex mechanism and may provide the path to the lock file to be used with the `fcntl(2)` or `flock(2)` system calls. 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 local directory that is not writable by other users.

Rationale:

If the lock file to be used as a mutex 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:

Verify the configuration does NOT include a `Mutex` directive with the mechanism of `fcntl`, `flock` or `file`.

If one of the file locking mechanisms is configured, then find the directory in which the lock file would be created. The default value is the `ServerRoot/logs` directory.

1. Verify that the lock file directory is not a directory within the Apache `DocumentRoot`
2. Verify that the ownership and group of the directory is `root:root` (or the user under which Apache initially starts up if not `root`).
3. Verify the permissions on the directory are only writable by root (or the startup user if not `root`),
4. Check that the lock file directory is on a locally mounted hard drive rather than an NFS mounted file system

Remediation:

Find the directory path in which the lock file would be created. The default value is the `ServerRoot/logs` directory.

1. Modify the directory if the path is a directory within the Apache `DocumentRoot`
2. Change the ownership and group to be `root:root`, if not already.
3. Change the permissions so that the directory is only writable by root, or the user under which Apache initially starts up (default is `root`),
4. Check that the lock file directory is on a locally mounted hard drive rather than an NFS mounted file system.

Default Value:

The default mechanism for the `Mutex` directive is platform specific and may be determined by running `httpd -V`. The default path is the `ServerRoot/logs` directory.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#mutex

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

14.6 Protect Information through Access Control Lists
Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.9 Ensure the Pid File Is Secured (Automated)

Profile Applicability:

- Level 1
- Level 2

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`).

Default Value:

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

References:

1. https://httpd.apache.org/docs/2.4/mod/mpm_common.html#pidfile

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

14.6 Protect Information through Access Control Lists
Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.10 Ensure the ScoreBoard File Is Secured (Automated)

Profile Applicability:

- Level 1
- Level 2

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 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.

Default Value:

The default scoreboard file is `logs/apache_status`.

References:

1. https://httpd.apache.org/docs/2.4/mod/mpm_common.html#scoreboardfile

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

14.6 Protect Information through Access Control Lists
Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.11 Ensure Group Write Access for the Apache Directories and Files Is Properly Restricted (Automated)

Profile Applicability:

- Level 1
- Level 2

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
```

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the

principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.12 Ensure Group Write Access for the Document Root Directories and Files Is Properly Restricted (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

Group permissions on Apache Document Root directories `$DOCROOT` may need to be writable 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
```

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists
All information stored on systems shall be protected with file system, network share,

claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

3.13 Ensure Access to Special Purpose Application Writable Directories is Properly Restricted (Manual)

Profile Applicability:

- Level 1
- Level 2

Description:

When the Apache webserver includes application software such as PHP, Java and many others, it is common for the application to require a writable directory. The writable directory may be needed for file uploads, application data, user session state information or many other purposes. It is important such directories have a single purpose, and have access properly secured to prevent a variety of possible exploits. The directory should be:

- Single Purpose Directory
- Outside the Configured Web DocumentRoot
- Owned by the root User or an Administrator Account
- Not writable by Other

Rationale:

The following provides the rationale for each requirement on the application writable directory:

- **Single Purpose Directory** - Each writable application directory should have a single purpose. For example, mixing file uploads in the same directory with session tracking information would be an obvious vulnerability, as users could create session information, to hijack or manufacturer authenticated sessions.
- **Outside the Configured Web DocumentRoot** - The directory should NOT be under the configured DocumentRoot directory as such directories are browsable by default, and might allow unintentional web read access. With web read access an attacker could upload malicious content, and then references the content in a URL exploiting the trust that users have in the website.
- **Owned by the root User or an Administrator Account** – The directory should be owned by root or a designated administrator to prevent unintended changes to the permissions.
- **Not Writable by Other** - The write access can be provided through the group permissions to the configured Apache group rather than allow write access to Other / all users. The group write access should implement the least privileges necessary in order prevent unintended access to the directory. If the application requires more complex write access, such as to specific accounts or for multiple groups, usage of an

access control lists (ACL) is recommended. ACL's are supported by most Linux file systems, and can be enabled when the file system is mounted.

Audit:

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

1. **Single Purpose Directory** - For each application writable directory review the documented purpose for the directory to confirm the directory serves a single purpose.
2. **Outside the Configured Web DocumentRoot** - For each writable directory and it's corresponding DocumentRoot perform the following. No output from the find command indicates the directory is not within the DocumentRoot.

```
# Set the WR_DIR to the writable directory such as the example shown
below
WR_DIR=/var/phptmp/sessions
# DOCROOT is the DocmentRoot directory for the web site or virtual
host.
DOCROOT=$(grep -i '^DocumentRoot' $APACHE_PREFIX/conf/httpd.conf | cut
-d' ' -f2 |
    tr -d '\"')
# Get Inode number of the writable Directory
INUM=$(stat -c '%i' $WR_DIR)
# Verify the directory is not found (No output = Not found)
find -L $DOCROOT -inum $INUM
```

3. **Owned by the root User or an Administrator Account** - For each writable directory, use the stat command to show the owner of each directory.

```
stat -c '%U' $WR_DIR/
```

4. **Not writable by Other** - For each writable directory, use the find command to identify directories writable by Other. No output indicates the directory and any sub-directories are not writable by Other.

```
find $WR_DIR/ -perm /o=w -ls
```

Remediation:

Perform the following:

1. **Single Purpose Directory** – Create separate directories of the multipurpose directory, and adjust the application configuration and directory ownership and permissions appropriately.
2. **Outside the Configured Web DocumentRoot** – Move the writable directory to a more suitable location NOT under the DocumentRoot directory. A location within the `/var/` filesystem may be a good choice for changeable data.

3. **Owned by the root User or an Administrator Account** – Change the ownership to root or an administrator.

```
chown root $WR_DIR
```

4. **Not writable by Other** – Remove the other write permissions, use group write or ACLs to provide the least privileges necessary.

```
chmod o-w $WR_DIR
```

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

4 Apache Access Control

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

4.1 Ensure Access to OS Root Directory Is Denied By Default (Automated)

Profile Applicability:

- Level 1
- Level 2

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 by 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 principle, and then helps prevent the unintended access, and we do that in this case by denying access to the OS root directory using either of two methods but not both:

1. Using the Apache `Deny` directive along with an `Order` directive.
2. Using the Apache `Require` directive.

Either method is effective. The `Order/Deny/Allow` combination are now deprecated; they provide three passes where all the directives are processed in the specified order. In contrast, the `Require` directive works on the first match similar to firewall rules. The `Require` directive is the default for Apache 2.4 and is demonstrated in the remediation procedure as it may be less likely to be misunderstood.

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 that either one of the following two methods are configured:

   Using the deprecated Order/Deny/Allow method:

   1. Ensure there is a single `Order` directive with the value of `deny, allow`
   2. Ensure there is a `Deny` directive, and with the value of `from all`.
   3. Ensure there are no `Allow` or `Require` directives in the root `<Directory>` element.

   Using the Require method:

   1. Ensure there is a single `Require` directive with the value of `all denied`
   2. Ensure there are no `Allow` or `Deny` 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. Add a single `Require` directive and set the value to `all denied`
3. Remove any `Deny` and `Allow` directives from the root `<Directory>` element.

```
<Directory>
   . . .
   Require all denied
   . . .
</Directory>
```

Default Value:

The following is the default root directory configuration:

```
<Directory>
   . . .
   Require all denied
   . . .
</Directory>
```

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#directory
2. https://httpd.apache.org/docs/2.4/mod/mod_authz_host.html

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

4.2 Ensure Appropriate Access to Web Content Is Allowed (Manual)

Profile Applicability:

- Level 1
- Level 2

Description:

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

Rationale:

Either the `Allow` or `Require` directives may be used within a directory, a location or other context to allow appropriate access. Access may be allowed to all, or to specific networks, or hosts, or users as appropriate. The `Allow/Deny/Order` directives are deprecated and should be replaced by the `Require` directive. It is also recommended that either the `Allow` directive or the `Require` directive be used, but not both in the same context.

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 that either one of the following two methods are configured:

   Use the deprecated Order/Deny/Allow method:

   1. Ensure there is a single `Order` directive with the value of `Deny,Allow` for each.
   2. Ensure the `Allow` and `Deny` directives, have values that are appropriate for the purposes of the directory.

   Use the Require method:

   1. Ensure that the `Order/Deny/Allow` directives are **NOT** used for the directory.
   2. Ensure the Require 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. Include the appropriate Require 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/">
    Require ip 192.169.
</Directory>

<Directory "/var/www/html/">
    Require all granted
</Directory>

<Location /usage>
    Require local
</Location>

<Location /portal>
    Require valid-user
</Location>
```

Default Value:

The following is the default Web root directory configuration:

```
<Directory "/usr/local/apache2/htdocs">
    . . .
    Require all granted
    . . .
</Directory>
```

References:

1. https://httpd.apache.org/docs/2.4/howto/auth.html
2. https://httpd.apache.org/docs/2.4/mod/mod_authz_host.html
3. https://httpd.apache.org/docs/2.4/mod/mod_authz_core.html
4. https://httpd.apache.org/docs/2.4/mod/mod_access_compat.html
5. https://httpd.apache.org/docs/2.4/mod/core.html#directory

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

4.3 Ensure OverRide Is Disabled for the OS Root Directory (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The Apache `AllowOverRide` directive and the new `AllowOverrideList` directive allow 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 the `.htaccess` files.

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.
3. Ensure there are no `AllowOverrideList` directives present.

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. Remove any `AllowOverrideList` directives found.
3. Add a single `AllowOverride` directive if there is none.
4. Set the value for `AllowOverride` to `None`.

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

Default Value:

The following is the default root directory configuration:

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

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#allowoverride
2. https://httpd.apache.org/docs/2.4/mod/core.html#allowoverridelist

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the

principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

4.4 Ensure OverRide Is Disabled for All Directories (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The Apache `AllowOverride` directive and the new `AllowOverrideList` directive allow 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.

Rationale:

`.htaccess` files decentralizes access control and increases the risk of server configuration being changed inappropriately.

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
```

3. Ensure there are no `AllowOverrideList` directives present.

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
. . .
```

3. Remove any `AllowOverrideList` directives found.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#allowoverride
2. https://httpd.apache.org/docs/2.4/mod/core.html#allowoverridelist

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

5 Minimize Features, Content and Options

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

5.1 Ensure Options for the OS Root Directory Are Restricted (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The Apache `Options` directive allows for specific configuration of options, including execution of CGI, following symbolic links, server side includes, and content negotiation.

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>
```

Default Value:

The default value for the root directory's `Option` directive is `Indexes FollowSymLinks`.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#options

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

5.2 Ensure Options for the Web Root Directory Are Restricted (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

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

- Execution of CGI
- Following symbolic links
- Server side includes
- Content negotiation

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 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 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>
```

Default Value:

The default value for the web root directory's `Option` directive is `FollowSymLinks`.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#options

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

5.3 Ensure Options for Other Directories Are Minimized (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The Apache `Options` directive allows for specific configuration of options, including execution of CGI, following symbolic links, server side includes, and content negotiation.

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 when multiple languages 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 https://httpd.apache.org/docs/2.4/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. https://httpd.apache.org/docs/2.4/mod/core.html#options

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

5.4 Ensure Default HTML Content Is Removed (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

Apache installations have default content that is not needed or appropriate for production use. The primary function for this 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. If it is part of main Apache `httpd` package such as it is on Red Hat Linux, then comment out

the configuration as shown below. Removing a file such as the `welcome.conf`, 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\>
```

Default Value:

The default source build provides extra content available in the `/usr/local/apache2/conf/extra/` directory, but the configuration of most of the extra content is commented out by default. **In particular, the include of `conf/extra/proxy-html.conf` is not commented out in the `httpd.conf`.**

```
# Server-pool management (MPM specific)
#Include conf/extra/httpd-mpm.conf
# Multi-language error messages
#Include conf/extra/httpd-multilang-errordoc.conf
# Fancy directory listings
#Include conf/extra/httpd-autoindex.conf
# Language settings
#Include conf/extra/httpd-languages.conf
# User home directories
#Include conf/extra/httpd-userdir.conf
# Real-time info on requests and configuration
#Include conf/extra/httpd-info.conf
# Virtual hosts
#Include conf/extra/httpd-vhosts.conf
# Local access to the Apache HTTP Server Manual
#Include conf/extra/httpd-manual.conf
# Distributed authoring and versioning (WebDAV)
#Include conf/extra/httpd-dav.conf
# Various default settings
#Include conf/extra/httpd-default.conf
# Configure mod_proxy_html to understand HTML4/XHTML1
<IfModule proxy_html_module\>
Include conf/extra/proxy-html.conf
</IfModule\>
# Secure (SSL/TLS) connections
#Include conf/extra/httpd-ssl.conf
```

Also, the only other default content is a minimal barebones `index.html` in the document root which contains.

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

CIS Controls:

Version 6

18.9 Sanitize Deployed Software Of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Version 7

5.1 Establish Secure Configurations

Maintain documented, standard security configuration standards for all authorized operating systems and software.

5.5 Ensure the Default CGI Content printenv Script Is Removed (Automated)

Profile Applicability:

- Level 1
- Level 2

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` or `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
```

Default Value:

The default source installation includes the `printenv` script. However, this script is not executable by default.

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

4.7 Limit Access to Script Tools
Limit access to scripting tools (such as Microsoft PowerShell and Python) to only administrative or development users with the need to access those capabilities.

5.6 Ensure the Default CGI Content test-cgi Script Is Removed (Automated)

Profile Applicability:

- Level 1
- Level 2

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
```

Default Value:

The default source installation includes the test-cgi script. However, this script is not executable by default.

CIS Controls:

Version 6

18.9 Sanitize Deployed Software Of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Version 7

4.7 Limit Access to Script Tools

Limit access to scripting tools (such as Microsoft PowerShell and Python) to only administrative or development users with the need to access those capabilities.

5.7 Ensure HTTP Request Methods Are Restricted (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

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 OS root directory.
3. Ensure that either one of the following two methods are configured:

   Using the deprecated Order/Deny/Allow method:

   1. Ensure that group contains a single `Order` directive within the `<Directory>` directive with a value of `deny, allow`
   2. Verify the `<LimitExcept>` directive does not include any HTTP methods other than `GET`, `POST`, and `OPTIONS`. (It may contain fewer methods.)

Using the Require method:

1. Ensure there is a single `Require` directive with the value of `all denied`
2. Ensure there are no `Allow` or `Deny` directives in the root element.

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. 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>
    Require all denied
</LimitExcept>
```

4. 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">
    . . .
    # Limit HTTP methods
    <LimitExcept GET POST OPTIONS>
        Require all denied
    </LimitExcept>
</Directory>
```

Default Value:

No Limits on HTTP methods.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#limitexcept
2. https://www.ietf.org/rfc/rfc2616.txt

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

5.8 Ensure the HTTP TRACE Method Is Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

Use the Apache `TraceEnable` directive to disable the HTTP `TRACE` request method.

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>`.

Default Value:

The `TRACE` method is enabled.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#traceenable
2. https://www.ietf.org/rfc/rfc2616.txt

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

5.9 Ensure Old HTTP Protocol Versions Are Disallowed (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

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:

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

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

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

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

2. 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 global server level configuration to disallow other protocol versions.

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

3. 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
```

Default Value:

The default value for the `RewriteEngine` directive is `off`.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

5.10 Ensure Access to .ht Files Is Restricted (Automated)*

Profile Applicability:

- Level 1
- Level 2

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 are `.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:
Verify that a `FilesMatch` directive similar to the one below is present in the apache configuration and not commented out. The deprecated `Deny from All` directive may be used instead of the Require directive.

```
<FilesMatch "^\.ht">
   Require all denied
</FilesMatch>
```

Remediation:

Perform the following to implement the recommended state:
Add or modify the following lines in the Apache configuration file at the server configuration level.

```
<FilesMatch "^\.ht">
    Require all denied
</FilesMatch>
```

Default Value:

`.ht*` files are not accessible.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#filesmatch

CIS Controls:

Version 6

18.3 Sanitize Input For In-house Software

For in-house developed software, ensure that explicit error checking is performed and documented for all input, including for size, data type, and acceptable ranges or formats.

Version 7

18.2 Ensure Explicit Error Checking is Performed for All In-house Developed Software

For in-house developed software, ensure that explicit error checking is performed and documented for all input, including for size, data type, and acceptable ranges or formats.

5.11 Ensure Access to Inappropriate File Extensions Is Restricted (Automated)

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.
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 "^.*$">
    Require all denied
</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)$">
    Require all granted
</FilesMatch>
```

Default Value:

There are no restrictions on file extensions in the default configuration.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#filesmatch

CIS Controls:

Version 6

18.3 Sanitize Input For In-house Software

For in-house developed software, ensure that explicit error checking is performed and documented for all input, including for size, data type, and acceptable ranges or formats.

Version 7

18.2 Ensure Explicit Error Checking is Performed for All In-house Developed Software

For in-house developed software, ensure that explicit error checking is performed and documented for all input, including for size, data type, and acceptable ranges or formats.

5.12 Ensure IP Address Based Requests Are Disallowed (Automated)

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 which will therefore include the host name in the HTTP HOST header.

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

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

- 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]
```

Default Value:

`RewriteEngine off`

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

5.13 Ensure the IP Addresses for Listening for Requests Are Specified (Automated)

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:
Verify that no `Listen` directives are in the Apache configuration file with no IP address specified, or with an IP address of all zeros.

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
```

Default Value:

```
Listen 80
```

References:

1. https://httpd.apache.org/docs/2.4/mod/mpm_common.html#listen

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services
Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

5.14 Ensure Browser Framing Is Restricted (Automated)

Profile Applicability:

- Level 2

Description:

To prevent Clickjacking or UI Redressing attacks, it's important for the server to include an HTTP header which instructs browsers to restrict the content from being framed. There are two headers that may be used. The Content-Security-Policy header, or the X-Frame-Options header. The Header directive allows server HTTP response headers to be added, replaced or merged. We will use the directive to add a 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 uses 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 attack vector, either an *X-Frame-Options* response header or a *Content-Security-Policy* response header may be used. The *Content-Security-Policy* header is the preferred solution. The *X-Frame-Options* header should have a value of either *DENY*, which prevents all framing, or *SAMEORIGIN* which prevents framing except via pages which share the same origin. The *Content-Security-Policy* header may also be to restrict framing with a *frame-ancestors* directive and a value of *none* or *self*

Audit:

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

- Ensure a Header directive for *Content-Security-Policy* is present in the Apache configuration and has the condition *always*, an action of *set* or *append* and a directive of *frame-ancestors* with a value of *none* or *self*, as shown below:

```
# grep -i Content-Security-Policy $APACHE_PREFIX/conf/httpd.conf
Header always append 'Content-Security-Policy frame-ancestors self'
```

- If no *Content-Security-Policy* header is found, check if a header directive for *X-Frame-Options* is present in the Apache configuration and has the condition *always*, an action of *set* or *append* and a value of *SAMEORIGIN* or *DENY*, as shown below:

```
# grep -i X-Frame-Options $APACHE_PREFIX/conf/httpd.conf
Header always set X-Frame-Options SAMEORIGIN
```

If either header configuration is present and has as a compliant value, then the server is compliant.

Remediation:

Perform the following to implement the recommended state:

Add or modify the Header directive for the *Content-Security-Policy* header in the Apache configuration to have the condition *always*, an action of *append* and a value of *frame-ancestors self*, as shown below.

```
Header always append 'Content-Security-Policy frame-ancestors self'
```

Default Value:

Neither the *Content-Security-Policy* HTTP response header nor the *X-Frame-Options* header is generated by default.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_headers.html#header
2. https://owasp.org/www-project-cheat-sheets/cheatsheets/Content_Security_Policy_Cheat_Sheet
3. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
4. https://en.wikipedia.org/wiki/Clickjacking

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

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.

6.1 Ensure the Error Log Filename and Severity Level Are Configured Correctly (Automated)

Profile Applicability:

- Level 1
- Level 2

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` for most modules, so that all errors from the `emerg` level through `notice` level will be logged. The recommended setting for the `core` module is `info` so that any `not found` requests will be included in the error logs.

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. Starting with Apache 2.4 the error log does not include the `not found` errors except at the `info` logging level. Therefore, it is important that the log level be set to `info` for the `core` module. The `not found` requests need to be included in the error log for both forensics' investigation and host intrusion detection purposes. Monitoring the access logs may not be practical for many web servers with high volume traffic.

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 `info` or lower for the core module and `notice` or lower for other modules. 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 core:info`.
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 `info` or lower for the core module and `notice` or lower for all other modules. 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 core:info`.

```
LogLevel notice core:info
```

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 monitoring the logs.

Default Value:

The following is the default configuration:

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

References:

1. https://httpd.apache.org/docs/2.4/logs.html
2. https://httpd.apache.org/docs/2.4/mod/core.html#loglevel
3. https://httpd.apache.org/docs/2.4/mod/core.html#errorlog

CIS Controls:

Version 6

6.2 Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

6.3 Enable Detailed Logging

Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

6.2 Ensure a Syslog Facility Is Configured for Error Logging (Automated)

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 which 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"
```

2. 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. https://httpd.apache.org/docs/2.4/logs.html
2. https://httpd.apache.org/docs/2.4/mod/core.html#loglevel
3. https://httpd.apache.org/docs/2.4/mod/core.html#errorlog

CIS Controls:

Version 6

6.6 Deploy A SIEM OR Log Analysis Tools For Aggregation And Correlation/Analysis

Deploy a SIEM (Security Information and Event Management) or log analytic tools for log aggregation and consolidation from multiple machines and for log correlation and analysis. Using the SIEM tool, system administrators and security personnel should devise profiles of common events from given systems so that they can tune detection to focus on unusual activity, avoid false positives, more rapidly identify anomalies, and prevent overwhelming analysts with insignificant alerts.

Version 7

6.6 Deploy SIEM or Log Analytic tool

Deploy Security Information and Event Management (SIEM) or log analytic tool for log correlation and analysis.

6.8 Regularly Tune SIEM

On a regular basis, tune your SIEM system to better identify actionable events and decrease event noise.

6.3 Ensure the Server Access Log Is Configured Correctly (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The `LogFormat` directive defines a nickname for a log 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 `CustomLog` directive is configured to an appropriate log file, syslog facility, or piped logging utility and the directive uses a log format that includes all of the format string tokens listed below. The log format string may be specified as a `LogFormat` nickname or as an explicit string. For example, either of the following two configurations are compliant:

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

CustomLog log/access_log "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"
\"%{User- agent}i\""
```

   The log format string should include the following tokens in any order. The portion *" = description text. "* describes the information to be logged.

   - `%h` = Remote hostname or IP address if HostnameLookups is set to Off, which is the default.

- `%l` =Remote logname / identity.
- `%u` =Remote user, if the request was authenticated.
- `%t` = Time the request was received,
- `%r` = First line of request.
- `%>s` = Final status.
- `%b` = Size of response in bytes.
- `%{Referer}i` = Variable value for Referer header.
- `%{User-agent}i` = Variable value for User Agent header.

2. 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 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 as well as the skills/training/tools for monitoring the logs.

Default Value:

The following are the default log configuration:

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

LogFormat "%h %l %u %t \"%r\" %>s %b" common

CustomLog "logs/access_log" common
```

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_log_config.html#customlog
2. https://httpd.apache.org/docs/2.4/mod/mod_log_config.html#formats

CIS Controls:

Version 6

6.2 Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

6.3 Enable Detailed Logging

Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

6.4 Ensure Log Storage and Rotation Is Configured Correctly (Automated)

Profile Applicability:

- Level 1
- Level 2

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`:
a) File Logging with Logrotate:

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

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

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

```
# rotate log files weekly
weekly
# keep 13 weeks of backlogs
rotate 13
```

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

b) Piped Logging:

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

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

2. Ensure the log file naming and any rotation scripts provide for retaining at least 3 months or 13 weeks of log files.
3. For each virtual host configured with its own log files ensure that those log files are also included in a similar log rotation.

Default Value:

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

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

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

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
```

CIS Controls:

Version 6

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)

Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

Version 7

6.4 Ensure adequate storage for logs

Ensure that all systems that store logs have adequate storage space for the logs generated.

6.5 Ensure Applicable Patches Are Applied (Automated)

Profile Applicability:

- Level 1
- Level 2

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 provided by the 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:
    - Check the Apache web site for latest versions, date of releases and any security patches. https://httpd.apache.org/security/vulnerabilities_24.html Apache patches are available https://www.apache.org/dist/httpd/patches
    - 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
    - Check for vendor supplied updates from the vendor web site.
    - 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:
    - Read release notes and related security patch information
    - Download latest source and any dependent modules such as `mod_security`.
    - Build new Apache software according to your build process with the same configuration options.

- Install and test the new software according to your organization's testing process.
- Move to production according to your organization's deployment process.

2. When using platform packages:
   - Read release notes and related security patch information
   - Download and install latest available Apache package and any dependent software.
   - Test the new software according to your organization's testing process.
   - Move to production according to your organization's deployment process.

Default Value:

Not Applicable

References:

1. https://httpd.apache.org/security/vulnerabilities_24.html

CIS Controls:

Version 6

4 Continuous Vulnerability Assessment and Remediation
Continuous Vulnerability Assessment and Remediation

Version 7

18.4 Only Use Up-to-date And Trusted Third-Party Components
Only use up-to-date and trusted third-party components for the software developed by the organization.

6.6 Ensure ModSecurity Is Installed and Enabled (Automated)

Profile Applicability:

- Level 2

Description:

ModSecurity is an open source web application firewall (WAF) for real-time web application monitoring, logging, and access control. It enables but does not include a powerful customizable rule set, which may be used to detect and block common web application attacks. Installation of ModSecurity without a rule set does not provide additional security for the protected web applications. Refer to the benchmark recommendation "*Install and Enable OWASP ModSecurity Core Rule Set*" for details on a recommended rule set.

Note: Like other application security/application firewall systems, ModSecurity requires a significant commitment of staff resources for initial tuning of the rules and handling alerts. In some cases, this may require additional time working with application developers/maintainers to modify applications based on analysis of the results of tuning and monitoring logs. After setup, an ongoing commitment of staff is required for monitoring logs and ongoing tuning, especially after upgrades/patches. Without this commitment to tuning and monitoring, installing ModSecurity may NOT be effective and may provide a false sense of security.

Rationale:

Installation of the ModSecurity Apache module enables a customizable web application firewall rule set which may be configured to detect and block common attack patterns as well as block outbound data leakage.

Audit:

Perform the following to determine if the `security2_module` has been loaded:
Use the httpd `-M` option as root to check that the module is loaded.

```
# httpd -M | grep security2_module
```

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:

1. Install the ModSecurity module if it is not already installed in `modules/mod_security2.so`. It may be installed via OS package installation (such as `apt-get` or `yum`) or built from the source files. See https://www.modsecurity.org/download.html for details.
2. Add or modify the `LoadModule` directive if not already present in the Apache configuration as shown below. Typically, the `LoadModule` directive is placed in file named `mod_security.conf` which is included in the Apache configuration:

```
LoadModule security2_module modules/mod_security2.so
```

Default Value:

The `ModSecurity` module is NOT loaded by default.

References:

1. https://www.modsecurity.org/

CIS Controls:

Version 6

18.2 Deploy And Configure Web Application Firewalls

Protect web applications by deploying web application firewalls (WAFs) that inspect all traffic flowing to the web application for common web application attacks, including but not limited to cross-site scripting, SQL injection, command injection, and directory traversal attacks. For applications that are not web-based, specific application firewalls should be deployed if such tools are available for the given application type. If the traffic is encrypted, the device should either sit behind the encryption or be capable of decrypting the traffic prior to analysis. If neither option is appropriate, a host-based web application firewall should be deployed.

Version 7

18.10 Deploy Web Application Firewalls (WAFs)

Protect web applications by deploying web application firewalls (WAFs) that inspect all traffic flowing to the web application for common web application attacks. For applications that are not web-based, specific application firewalls should be deployed if such tools are available for the given application type. If the traffic is encrypted, the device should either sit behind the encryption or be capable of decrypting the traffic prior to analysis. If neither option is appropriate, a host-based web application firewall should be deployed.

6.7 Ensure the OWASP ModSecurity Core Rule Set Is Installed and Enabled (Automated)

Profile Applicability:

- Level 2

Description:

The OWASP ModSecurity Core Rules Set (CRS) is a set of open source web application defensive rules for the ModSecurity web application firewall (WAF). The OWASP ModSecurity CRS provides baseline protections in the following attack/threat categories:

- HTTP Protection - detecting violations of the HTTP protocol and a locally defined usage policy.
- Real-time Blacklist Lookups - utilizes 3rd Party IP Reputation
- HTTP Denial of Service Protections - defense against HTTP Flooding and Slow HTTP DoS Attacks.
- Common Web Attacks Protection - detecting common web application security attack.
- Automation Detection - detecting bots, crawlers, scanners and other surface malicious activity.
- Integration with AV Scanning for File Uploads - detects malicious files uploaded through the web application.
- Tracking Sensitive Data - tracks credit card usage and blocks leakages.
- Trojan Protection - detecting access to trojan horses.
- Identification of Application Defects - alerts on application misconfigurations.
- Error Detection and Hiding - disguising error messages sent by the server.

Note: Like other application security/application firewall systems, ModSecurity requires a significant commitment of staff resources for initial tuning of the rules and handling alerts. In some cases, this may require additional time working with application developers/maintainers to modify applications based on analysis of the results of tuning and monitoring logs. After setup, an ongoing commitment of staff is required for monitoring logs and ongoing tuning, especially after upgrades/patches. Without this commitment to tuning and monitoring, installing ModSecurity may NOT be effective and may provide a false sense of security.

Rationale:

Installing, configuring and enabling of the OWASP ModSecurity Core Rule Set (CRS), provides additional baseline security defense, and provides a good starting point to customize the monitoring and blocking of common web application attacks.

Audit:

For the **OWASP ModSecurity CRS version 2.2.9**, perform the following to audit the configuration.

In the 2.2.9 release, the OWASP ModSecurity CRS contains 15 `base_rule` configuration files, each with rule sets. The CRS also contains 14 optional rule sets, and 17 experimental rule sets. Since it is expected that customization and testing will be necessary to implement the CRS, it is not expected that any site will implement all CRS configuration files / rule sets. Therefore, for the purpose of auditing, the OWASP ModSecurity CRS will be considered implemented if 200 or more of the security rules (`SecRule`) are active in the CRS configuration files. The default 2.2.9 installation contains 227 security rules. Perform the following to determine if 2.2.9 OWASP ModSecurity CRS is enabled:

- Set `RULE_DIR` environment variable to the directory where the active rules are included from the modsecurity configuration file. An example is shown below.

```
RULE_DIR=$APACHE_PREFIX/modsecurity.d/activated_rules/
```

- Use the following command to count the security rules in all of the active CRS configuration files.

```
find $APACHE_PREFIX/modsecurity.d/activated_rules/ -name
'modsecurity_crs_*.conf' | xargs grep '^SecRule ' | wc -l
```

- If the number of active files is 200 or greater, then OWASP ModSecurity CRS is considered active and the audit passed.

For the **OWASP ModSecurity CRS version 3.0**, perform the following to audit the configuration.

In the 3.0 release, the OWASP ModSecurity CRS contains 29 rule configuration files, each with rule sets. It is expected that customization and testing will be necessary to implement the CRS; it is not expected that any site will implement all CRS configuration files / rule sets. Therefore, for the purpose of auditing, the OWASP ModSecurity CRS v3.0 will be considered implemented if 325 or more of the security rules (SecRule) are active in the CRS configuration files. The default OWASP ModSecurity CRS 3.0 installation contains 462 security rules. In addition to the rules, there are three additional values that have to be set. The Inbound and the Outbound Anomaly Threshold and the Paranoia Mode. The Anomaly Threshold values set a limit so that traffic is not blocked until the threshold is exceeded. Any traffic that triggers enough active rules so that the additive value of each rule exceeds the threshold value will be block. The suitable paranoia level has to be defined according to the security level of the service in question. The default value of 1 should be applicable for

any online service. The Paranoia Level 2 should be chosen for online services with a need for further hardening, (such as online services with a wide attack surface or online services with known security issues and concerns). Paranoia Level 3 and Level 4 cater services with even higher security requirements but have to be considered experimental.
Perform the following to determine if OWASP ModSecurity CRS 3.0 is enabled, and is configured to meet or exceed the expected values:

- Set `RULE_DIR` environment variable to the directory where the active rules are included from the modsecurity configuration file. An example is shown below.

```
RULE_DIR=$APACHE_PREFIX/modsecurity.d/owasp-modsecurity-crs-3.0.0/
```

- Use the following command to count the security rules in all of the active CRS configuration files.

```
find $RULE_DIR -name '*.conf' | xargs grep '^SecRule ' | wc -l
```

- If the number of active rules is 325 or greater then OWASP ModSecurity CRS 3.0 is considered active.
- The Inbound Anomaly Threshold must be less than or equal to `5`, and can be checked with the following command.

```
find $RULE_DIR -name '*.conf' | xargs egrep -v '^\s*#' | grep
'setvar:tx.inbound_anomaly_score_threshold'
```

- The Outbound Anomaly Threshold must be less than or equal to `4`, and may be audited with the following command.

```
find $RULE_DIR -name '*.conf' | xargs egrep -v '^\s*#' | grep
'setvar:tx.outbound_anomaly_score_threshold'
```

- The Paranoia Level must be greater than or equal to `1`, and may be audited with the following command.

```
find $RULE_DIR -name '*.conf' | xargs egrep -v '^\s*#' | grep
'setvar:tx.paranoia_level'
```

Remediation:

Install, configure and test the OWASP ModSecurity Core Rule Set:

1. Download the OWASP ModSecurity CRS from the project page https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project
2. Unbundled the archive and follow the instructions in the `INSTALL` file.

3. Depending on the CRS version used, the `crs-setup.conf` or the `modsecurity_crs_10_setup.conf` file will be required, and rules in the `base_rules` directory are intended as a baseline useful for most applications.
4. Test the application for correct functionality after installing the CRS. Check web server error logs and the `modsec_audit.log` file for blocked requests due to false positives.
5. It is also recommended to test the application response to malicious traffic such as an automated web application scanner to ensure the rules are active. The web server error log and `modsec_audit.log` files should show logs of the attacks and the servers response codes.

Default Value:

The OWASP ModSecurity CRS is NOT installed or enabled by default.

CRS v3.0 Default Values:

- `inbound_anomaly_score_threshold = 5`
- `outbound_anomaly_score_threshold = 4`
- `paranoia_level = 1`

References:

1. https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project
2. https://www.modsecurity.org/

CIS Controls:

Version 6

18.2 Deploy And Configure Web Application Firewalls

Protect web applications by deploying web application firewalls (WAFs) that inspect all traffic flowing to the web application for common web application attacks, including but not limited to cross-site scripting, SQL injection, command injection, and directory traversal attacks. For applications that are not web-based, specific application firewalls should be deployed if such tools are available for the given application type. If the traffic is encrypted, the device should either sit behind the encryption or be capable of decrypting the traffic prior to analysis. If neither option is appropriate, a host-based web application firewall should be deployed.

Version 7

18.10 Deploy Web Application Firewalls (WAFs)

Protect web applications by deploying web application firewalls (WAFs) that inspect all

traffic flowing to the web application for common web application attacks. For applications that are not web-based, specific application firewalls should be deployed if such tools are available for the given application type. If the traffic is encrypted, the device should either sit behind the encryption or be capable of decrypting the traffic prior to analysis. If neither option is appropriate, a host-based web application firewall should be deployed.

7 SSL/TLS Configuration

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

7.1 Ensure mod_ssl and/or mod_nss Is Installed (Automated)

Profile Applicability:

- Level 1
- Level 2

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

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

Results should show 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. If a new version of Openssl is needed it may be downloaded from http://www.openssl.org/ See the Apache documentation on building from source http://httpd.apache.org/docs/2.4/install.htmlfor 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
   ```

Default Value:

SSL/TLS is not enabled by default.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html
2. https://www.centos.org/docs/5/html/5.4/technical-notes/mod_nss.html

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.2 Ensure a Valid Trusted Certificate Is Installed (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

Note: Some previously "Trusted" Certificate Authority certificates had been signed with a weak hash algorithm such as MD5, or SHA1. These signature algorithms are known to be vulnerable to collision attacks. Note that it's not the just the signature on the server's certificate, but any signature up the certificate chain. Such CA certificates are considered no longer trusted as of January 1, 2017.

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 one or more of the following steps to determine if the recommended state is implemented:

1. The Qualys SSL Labs has a website that may be used for testing external servers. https://www.ssllabs.com/ssltest/ Enter the external host name of the server and wait for an extensive tests of TLS protocols and ciphers, in addition to testing the server certificate and the entire certificate authority chain. The SSL Labs test will report any weak digital signatures of the intermediate certificate authorities. For example, the report may include a warning of:

```
Intermediate certificate has an insecure signature. Upgrade to SHA2 as
soon as possible to avoid browser warnings.
```

In addition, the weak SHA1 or MD5 signature algorithm will be highlighted with red text where the additional intermediate CA certificates are enumerated. For example, the certificate below from an SSL Labs report used SHA1 for the digital signature:

- Subject The Go Daddy Group, Inc.
- Fingerprint SHA256: 18f8a7. . .
- Pin SHA256: VjLZe. . .
- Valid until Sat, 29 Jun . . .
- Key RSA 2048 bits (e 3)
- Issuer http://www...
- Signature algorithm **SHA1withRSA INSECURE**

If a weak signature is found, then follow your certificate authority's process for having the server certificate re-issued / re-signed, in order to ensure that it is signed with a strong digital signature.

2. If the server is not an external server, or is not running on the standard port 443, a vulnerability scanner such as Nessus may be used to validate both the server certificate and the intermediate certificate chain. Custom certificate authorities may also be tested by loading the root certificate into the vulnerability scanner.
3. The testing can also be done by connecting to a running web server with your favorite browser and checking for a warning with regard to the certificate trust. However, some browsers may not warn of weak digital signatures, or other certificate issues.
4. OpenSSL can also be used to validate a certificate as a valid trusted certificate, using a trusted bundle of CA certificate. It is important that the CA bundle of certificates be an already validated and trusted file in order for the test to be valid.

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

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:
https://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:
   - https://httpd.apache.org/docs/2.4/ssl/ssl_faq.html#realcert
   - https://www.openssl.org/docs/HOWTO/certificates.txt

```
# cd /etc/ssl/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. Create a certificate specific template configuration file. It is important that common name in the certificate exactly make the web host name in the intended URL. If there are multiple host names which may be used, as is very common, then the `subjectAltName` (SAN) field should be filled with all of the alternate names. Creating a template configuration file specific to the server certificate is helpful, as it allows for multiple entries in the `subjectAltName`. Also, any typos in the CSR can be potentially costly due to the lost time, so using a file, rather than hand typing helps prevent errors. To create a template configuration file, make a local copy of the `openssl.cnf` typically found in `/etc/ssl/` or `/etc/pki/tls/`

```
# cp /etc/ssl/openssl.cnf ex1.cnf
```

4. Find the request section which follows the line "`[ req ]` ". Then add or modify the configuration file to include the appropriate values for the host names. It is recommended (but not required) that the first `subjectAltName` match the `commonName`.

```
[ req ]
. . .
distinguished_name = req_distinguished_name
req_extensions = req_ext

[ req_ext ]
subjectAltName = @alt_names

[alt_names]
DNS.1 = www.example.com
DNS.2 = example.com
DNS.3 = app.example.com
DNS.4 = service.example.com
```

5. Continue editing the configuration file under the request distinguished name section to change the existing default values in the configuration file to match the desired certificates information.

```
[ req_distinguished_name ]
countryName_default             = GB
stateOrProvinceName_default     = Scotland
localityName_default            = Glasgow
0.organizationName_default      = Example Company Ltd
organizationalUnitName_default  = ICT
commonName_default              = www.example.com
```

6. Now generate the CSR from the template file, verifying the information. If the default values were placed in the template, then just press enter to confirm the default value.

```
# openssl req -new -config ex2.cnf -out example.com.csr -key
example.com.key
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]:
State or Province Name (full name) [Scotland]:
Locality Name (eg, city) [Glasgow]:
Organization Name (eg, company) [Example Company Ltd]:
Organizational Unit Name (eg, section) [ICT]:
Common Name (e.g. server FQDN or YOUR name) [www.example.com]:
```

7. Review and verify the CSR information including the SAN by displaying the information.

```
# openssl req -in ex2.csr -text | more

Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = GB, ST = Scotland, L = Glasgow, O = Example
Company Ltd, OU = ICT, CN = www.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:cb:c2:7a:04:13:19:7a:c0:74:00:63:dd:e9:6e:
                    . . .  <snip> . . .
                    3a:9d:aa:50:09:4a:40:48:b4:e2:24:ef:fa:7b:42:
                    a4:33
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Subject Alternative Name:
              DNS:www.example.com, DNS:example.com,
DNS:app.example.com, DNS:ws.example.com
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
    Signature Algorithm: sha256WithRSAEncryption
         73:f0:e3:90:a7:ab:01:e4:7f:12:19:b7:6a:dd:be:4e:5c:f1:
         . . .
```

8. Now move the private key to its intended directory.

```
# mv www.example.com.key /etc/ssl/private/
```

9. 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.
10. The resulting signed certificate may be named `www.example.com.crt` and placed in `/etc/ssl/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.
11. 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/ssl/private/
# umask 077
# openssl rsa -in www.example.com.key -out www.example.com.key.clear
```

12. 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 may need to configure the CA's certificate along with any intermediate CA certificates that signed your certificate using the `SSLCertificateChainFile` directive. As an alternative, starting with Apache version 2.4.8 the CA and intermediate certificates may be concatenated to the server certificate configured with the `SSLCertificateFile` directive instead.

```
SSLCertificateFile /etc/ssl/certs/example.com.crt
SSLCertificateKeyFile /etc/ssl/private/example.com.key
# Default CA file, can be replaced with your CA certificate.
SSLCertificateChainFile /etc/ssl/certs/server-chain.crt
```

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

References:

1. https://www.owasp.org/index.php/Testing_for_SSL-TLS_%28OWASP-CM-001%29
2. https://httpd.apache.org/docs/2.4/ssl/ssl_faq.html#realcert
3. https://www.openssl.org/docs/HOWTO/certificates.txt
4. https://security.googleblog.com/2014/09/gradually-sunsetting-sha-1.html

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.3 Ensure the Server's Private Key Is Protected (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

It is critical to protect the server's private key. The server's private key is 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. 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 as well as 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 of the `SSLCertificateKeyFile` directives, change the ownership and permissions on the server private key to be owned by `root:root` with permission `0400`.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html
2. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslpassphrasedialog

CIS Controls:

Version 6

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

Version 7

14.6 Protect Information through Access Control Lists
Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

7.4 Ensure the TLSv1.0 and TLSv1.1 Protocols are Disabled (Automated)

Profile Applicability:

- Level 2
- Level 1

Description:

The TLSv1.0 and TLSv1.1 protocols should be disabled via the `SSLProtocol` directive. The TLSv1.0 protocol is vulnerable to information disclosure and both protocols lack support for modern cryptographic algorithms including authenticated encryption. The only SSL/TLS protocols that should be allowed is TLSv1.2 along with the newer TLSv1.3 protocol.

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 which is also weak and is not recommended. Therefore, it is recommended that the TLSv1.0 protocol be disabled. The TLSv1.1 protocol does not support Authenticated Encryption with Associated Data (AEAD) which is designed to simultaneously provide confidentiality, integrity, and authenticity. All major up-to-date browsers support TLSv1.2, and most recent versions of FireFox and Chrome support the newer TLSv1.3 protocol, since 2017.

The NIST SP 800-52r2 guidelines for TLS configuration require that TLS 1.2 is configured with FIPS-based cipher suites be supported by all government TLS servers and clients and requires support of TLS 1.3 by January 1, 2024. A September 2018 IETF draft also depreciates the usage of TLSv1.0 and TLSv1.1 as shown in the references.

As of March 2020 all major browsers will no longer support TLS 1.0 or TLS 1.1.

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 matches one of the values below.

```
SSLProtocol TLSv1.2 TLSv1.3

SSLProtocol TLSv1.2
```

Remediation:

Perform the following to implement the recommended state:

1. Check if the TLSv1.3 protocol is supported by the Apache server by either checking that the version of OpenSSL is 1.1.1 or later or place the `TLSv1.3` value in the `SSLProtocol` string of a configuration file and check the syntax with the 'httpd -t' command before using the file in production. Two examples below are shown of servers that do support the TLSv1.3 protocol.

```
$ openssl version
OpenSSL 1.1.1a  20 Nov 2018

### _(Add TLSv1.3 to the SSLProtocol directive)_
# httpd -t
Syntax OK
```

2. Search the Apache configuration files for the `SSLProtocol` directive; add the directive, if not present, or change the value to `TLSv1.2` or `TLSv1.2 TLSv1.3` if the TLSv1.3 protocol is supported.

Default Value:

`SSLProtocol all`

References:

1. https://www.godaddy.com/garage/browser-support-tls-10-11/
2. https://caniuse.com/#search=tls%201.2
3. https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/draft
4. https://en.wikipedia.org/wiki/Authenticated_encryption
5. https://tools.ietf.org/html/draft-ietf-tls-oldversions-deprecate-00

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.5 Ensure Weak SSL/TLS Ciphers Are Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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 server's 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 server's preferred ciphers will be used rather than the clients' preferences.

In addition, the RC4 stream ciphers should be disabled, even though they are widely used and have been recommended in previous Apache benchmarks as a means of mitigating attacks based on CBC cipher vulnerabilities. The RC4 ciphers have known cryptographic weaknesses and are no longer recommended. The IETF has published RFC 7465 standard [2] that would disallow RC4 negotiation for all TLS versions. While the document is somewhat new (Feb 2015) it is expected 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 are disabled in the configuration.

Audit:

Perform the following steps to determine if the recommended state is implemented:
The SSL protocols and ciphers supported can be easily tested by connecting to a running web server with an up-to-date version of the sslscan tool. The tool is available on Kali Linux https://www.kali.org/, or via github https://github.com/rbsec/sslscan The tool will color highlight the following weak ciphers.

- Red Background NULL cipher (no encryption)
- Red Broken cipher (<= 40 bit), broken protocol (SSLv2 or SSLv3)
- Yellow Weak cipher (<= 56 bit or RC4)
- Purple Anonymous cipher (ADH or AECDH)

Alternatively, the Qualys SSL Labs has a website that may be used for testing external servers. https://www.ssllabs.com/
Alternatively, verify the `SSLCipherSuite` directive is present and has the following values to disable weak ciphers in the Apache server level configuration and every virtual host that is SSL/TLS enabled.

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

Remediation:

Perform the following to implement the recommended state:

Ensure the `SSLCipherSuite` includes all of the following:

`!NULL:!SSLv2:!RC4:!aNULL` values. For example, add or modify the following line in the Apache server level configuration and every virtual host that is TLS enabled:

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

It is **not** recommended to add `!SSLv3` to the directive even if the SSLv3 protocol is not in use. Doing so disables ALL of the ciphers that may used with SSLv3, which includes the same ciphers used with the TLS protocols. The `!aNULL` will disable both the ADH and AECDH ciphers, so the `!ADH` is not required.

IMPORTANT NOTE: The above `SSLCipherSuite` value disables only the weak ciphers but allows medium strength and other ciphers which should also be disabled. Refer to the remaining TLS benchmark recommendations for stronger cipher suite values. The following cipher suite value will meet all of the level 1 and level 2 benchmark recommendations. As always, testing prior to production use is highly recommended.

```
SSLHonorCipherOrder On
SSLCipherSuite EECDH:EDH:!NULL:!SSLv2:!RC4:!aNULL:!3DES:!IDEA
```

Default Value:

The following are the default values:

`SSLCipherSuite` default depends on OpenSSL version.

`SSLHonorCipherOrder` default is Off

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslciphersuite
2. https://www.ssllabs.com/
3. https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/final
4. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf
5. https://community.qualys.com/blogs/securitylabs/2013/03/19/rc4-in-tls-is-broken-now-what
6. https://github.com/rbsec/sslscan

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.6 Ensure Insecure SSL Renegotiation Is Not Enabled (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

A man-in-the-middle renegotiation attack was discovered in SSLv3 and TLSv1 in November, 2009 (CVE-2009-3555). 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: https://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 provide backward compatibility to clients with the older, unpatched SSL implementations.

Rationale:

Enabling the `SSLInsecureRenegotiation` directive leaves the server vulnerable to man-in-the-middle renegotiation attack. Therefore, the `SSLInsecureRenegotiation` directive should not be enabled.

Audit:

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

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:

```
SSLInsecureRenegotiation off
```

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslinsecurerenegotiation
2. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555
3. https://azure.microsoft.com/en-us/services/multi-factor-authentication/

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.7 Ensure SSL Compression is not Enabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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:

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`.

Remediation:

Perform the following to implement the recommended state:

1. Search the Apache configuration files for the `SSLCompression` directive.
2. If the directive is present, set it to `off`.

Default Value:

In Apache versions >= 2.4.3, the `SSLCompression` directive is available and SSL compression is implicitly disabled. In Apache 2.4 - 2.4.2, the `SSLCompression` directive is not available and SSL compression is implicitly disabled.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslcompression
2. https://en.wikipedia.org/wiki/CRIME_(security_exploit)

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.8 Ensure Medium Strength SSL/TLS Ciphers Are Disabled (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

The SSLCipherSuite directive specifies which ciphers are allowed in the negotiation with the client. Disable the medium strength ciphers such as Triple DES (3DES) and IDEA by adding `!3DES` and `!IDEA` in the SSLCipherSuite directive.

Rationale:

Although Triple DES has been a trusted standard in the past, several vulnerabilities for it have been published over the years and it is no longer considered secure. A vulnerable against 3DES in CBC mode was nicknamed the SWEET32 attack, was published in 2016 as CVE-2016-2183. The IDEA cipher in CBC mode, is also vulnerable to the SWEET32 attack.

Audit:

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

- The SSL protocols and ciphers supported can be easily tested by connecting to a running web server with an up-to-date version of the sslscan tool. The tool is available on Kali Linux https://www.kali.org/, or via github https://github.com/rbsec/sslscan Use the command below to detect 3DES and IDEA ciphers. No output means the ciphers are not allowed.

```
$ sslscan --no-colour www.lugor.org | egrep 'IDEA|DES'
Accepted  TLSv1.2  112 bits  ECDHE-RSA-DES-CBC3-SHA      Curve P-256
DHE 256
Accepted  TLSv1.2  112 bits  EDH-RSA-DES-CBC3-SHA       DHE 2048 bits
Accepted  TLSv1.2  112 bits  DES-CBC3-SHA
Accepted  TLSv1.1  112 bits  ECDHE-RSA-DES-CBC3-SHA      Curve P-256
DHE 256
Accepted  TLSv1.1  112 bits  EDH-RSA-DES-CBC3-SHA       DHE 2048 bits
Accepted  TLSv1.1  112 bits  DES-CBC3-SHA
```

- Alternatively, the Qualys SSL Labs has a webiste that may be used for testing external servers. https://www.ssllabs.com/
- Alternatively, verify the `SSLCipherSuite` directive includes the `!3DES` and the `!IDEA` to disable the ciphers in the Apache server level configuration and every virtual host that is SSL/TLS enabled.

Remediation:

Perform the following to implement the recommended state:

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

```
SSLHonorCipherOrder On
SSLCipherSuite ALL:!EXP:!NULL:!LOW:!SSLv2:!RC4:!aNULL:!3DES:!IDEA
```

IMPORTANT NOTE: The above `SSLCipherSuite` value disables only the weak and medium ciphers but allows other ciphers which should also be disabled. Refer to the remaining TLS benchmark recommendations for more stronger cipher suite values. The following cipher suite value will meet all of the level 1 and level 2 benchmark recommendations. As always, testing prior to production use is highly recommended.

```
SSLHonorCipherOrder On
SSLCipherSuite EECDH:EDH:!NULL:!SSLv2:!RC4:!aNULL:!3DES:!IDEA
```

Default Value:

The following are the default values:

`SSLCipherSuite` default depends on OpenSSL version.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslprotocol
2. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html#sslciphersuite
3. https://sweet32.info/
4. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-2183
5. https://github.com/rbsec/sslscan
6. https://www.openssl.org/

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.9 Ensure All Web Content is Accessed via HTTPS (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

All of the website content should be served via HTTPS rather than HTTP. A redirect from the HTTP website to the HTTPS content is often useful and is recommended, but all significant content should be accessed via HTTPS so that it is authenticated and encrypted.

Rationale:

The usage of clear text HTTP prevents the client browser from authenticating the connection and ensuring the integrity of the website information. Without the HTTPS authentication, a client may be subjected to a variety of man-in-the-middle and spoofing attacks which would cause them to receive modified web content which could harm the organization's reputation. Through DNS attacks or malicious redirects, the client could arrive at a malicious website instead of the intended website. The malicious website could deliver malware, request credentials, or deliver false information.

Audit:

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

- Gather the list of listening IP addresses from the Apache configuration files. The commands below may be used to extract the relevant IP addresses from the configuration files. The `CONF_DIRS` variable needs to be set to the list of directories that contain all of the Apache configuration files.

```
## Replace the following directory list with the appropriate list.
CONF_DIRS="/etc/httpd/conf /etc/httpd/conf.d /etc/httpd/conf_dir2 . . .
"
CONFS=$(find  $CONF_DIRS -type f -name '*.conf' )
## Search for Listen directives that are not port :443 or https
IPS=$(egrep -ih '^\s*Listen ' $CONFS | egrep -iv '(:443\b)|https' | cut
-d' ' -f2)
```

- Gather the list of virtual host names from the Apache configuration files. The commands below can be used to extract the relevant virtual host names from the configuration files listed in `$CONFS`. The resulting list will include all virtual hosts not running on port :443. Although some listed virtual hosts may be TLS enabled, but on

a non-standard port. Such websites will return an error rather than HTML content, as shown in the final steps.

```
## Get host names and ports of all of the virtual hosts
VHOSTS=$(egrep -iho '^\s*<VirtualHost .*>' $CONFS | egrep -io '\s+[A-
Z:.0-9]+>$' | \
    tr -d ' >')
```

- For each of the IP address and virtual hosts name, prefix the IP address or host name with the `http://` protocol, and add the final slash as well.

```
URLS=$(for h in $LIPADDR $VHOSTS ;  do echo "http://$h/"; done)
```

- Check to ensure each URL does not deliver significate web content via the HTTP protocol. The URL's may be manually entered in a browser for testing, or may be scripted with a command line web client such as curl, as shown below.

```
## For each of the URL's test with curl, and truncate the output to 300
characters
for u in $URLS ; do echo -e "\n\n\n=== $u ==="; curl -fSs $u | head -c
300 ; done
```

Any URLs which return significant HTML document content, rather than a redirect or an error are not compliant. Two compliant examples are shown; the first one has a redirect.

```
=== http://www.cisecurity.org/ ===
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://www.cisecurity.org/">here</a>.</p>
</body></html>
```

This compliant example below returns an error, due to using HTTP on a HTTPS website.

```
=== http://www.example.com:4430/ ===
curl: (22) The requested URL returned error: 400 Bad Request
```

Remediation:

Perform the following to implement the recommended state:
Move the web content to a TLS enabled website, and add an HTTP `Redirect` directive to the Apache configuration file to redirect to the TLS enabled website similar to the example shown.

```
Redirect permanent / https://www.cisecurity.org/
```

Default Value:

The following are the default values:

TLS is not enabled by default.

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks
All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

7.10 Ensure OCSP Stapling Is Enabled (Automated)

Profile Applicability:

- Level 2

Description:

The OCSP (Online Certificate Status Protocol) provides the current revocation status of an X.509 certificate and allows for a certificate authority to revoke the validity of a signed certificate before its expiration date. The URI for the OCSP server is included in the certificate and verified by the browser. The Apache `SSLUseStapling` directive along with the `SSLStaplingCache` directive are recommended to enable OCSP Stapling by the web server. If the client requests OCSP stapling, then the web server can include the OCSP server response along with the web server's X.509 certificate.

Rationale:

The OCSP protocol is a big improvement over CRLs (certificate revocation lists) for checking if a certificate has been revoked. There are however some minor privacy and efficiency concerns with OCSP. The fact that the browser has to check a third-party CA discloses that the browser is configured for OCSP checking. Also, the already high overhead of making an SSL connection is increased by the need for the OCSP requests and responses. The OCSP stapling improves the situation by having the SSL server "staple" an OCSP response, signed by the OCSP server, to the certificate it presents to the client. This obviates the need for the client to ask the OCSP server for status information on the server certificate. However, the client will still need to make OCSP requests on any intermediate CA certificates that are typically used to sign the server's certificate.

Audit:

Perform 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 the `SSLStaplingCache` directive is present and not commented out. There are three supported cache types, any of them are considered compliant.
- Verify the `SSLUseStapling` directive is enabled with a value of `on`

Remediation:

Perform the following to implement the recommended state:
Add or modify the `SSLUseStapling` directive to have a value of `on` in the Apache server

level configuration and every virtual host that is SSL enabled. Also ensure that `SSLStaplingCache` is set to one of the three cache types similar to the examples below.

```
SSLUseStapling On
SSLStaplingCache "shmcb:logs/ssl_staple_cache(512000)"
- or-
SSLStaplingCache "dbm:logs/ssl_staple_cache.db"
- or -
SSLStaplingCache  dc:UNIX:logs/ssl_staple_socket
```

Default Value:

`SSLUseStapling Off SSLStaplingCache` *`<no default value>`*

References:

1. https://en.wikipedia.org/wiki/OCSP_stapling - OCSP Stapling
2. https://httpd.apache.org/docs/2.4/mod/mod_ssl.html- Apache SSL Directives

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.11 Ensure HTTP Strict Transport Security Is Enabled (Automated)

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.

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 to include 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:

The Strict Transport Security header is not present by default.

References:

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

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

7.12 Ensure Only Cipher Suites That Provide Forward Secrecy Are Enabled (Automated)

Profile Applicability:

- Level 2

Description:

In cryptography, *forward secrecy* (FS), which is also known as *perfect forward secrecy* (PFS), is a feature of specific key exchange protocols that give assurance that your session keys will not be compromised even if the private key of the server is compromised. Protocols such as RSA do not provide the forward secrecy, while the protocols `ECDHE` (Elliptic-Curve Diffie-Hellman Ephemeral) and the `DHE` (Diffie-Hellman Ephemeral) will provide forward secrecy. The `ECDHE` is the stronger protocol and should be preferred, while the `DHE` may be allowed for greater compatibility with older clients. The TLS ciphers should be configured to require either the `ECDHE` or the `DHE` ephemeral key exchange, while not allowing other cipher suites.

Rationale:

During the TLS handshake, after the initial client & server Hello, there is a pre-master secret generated, which is used to generate the master secret, and in turn generates the session key. When using protocols that do not provide forward secrecy, such as RSA, the pre-master secret is encrypted by the client with the server's public key and sent over the network. However, with protocols such as `ECDHE` (Elliptic-Curve Diffie-Hellman Ephemeral) the pre-master secret is not sent over the wire, even in encrypted format. The key exchange arrives at the shared secret in the clear using ephemeral keys that are not stored or used again. With FS, each session has a unique key exchange, so that future sessions are protected.

Audit:

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

- The SSL protocols and ciphers supported can be easily tested by connecting to a running web server with an up-to-date version of the `sslscan` tool. The tool is available on Kali Linux https://www.kali.org/, or via github https://github.com/rbsec/sslscan. Usage of Kali Linux for `sslscan` is highly recommended rather than other Linux distributions as it is important that the scan make use of an SSL library that still enables the old protocols. Current Linux versions often wisely eliminate support for older protocols such as SSLv3, and

therefore may be unable to properly detect the availability of older protocols on a remote system. A statically compiled sslscan with its own openssl library that supports the older protocols may be used as well.

Check the output of `sslscan`, and confirm that all accepted ciphers begin with either `ECDHE-` or `DHE-`. Any ciphers not starting with one of the ephemeral Diffie-Helman algorithms, is not implementing the recommended state. The `sslscan` command below includes regular expressions which will extract any ciphers which are not included in the recommendation. No output means that only the FS ciphers are allowed.

```
$ sslscan --no-colour --no-failed www.example.com | egrep
'(^Accepted)|(^Preferred)' | egrep -v '( ECDHE-)|( DHE-)'
```

- Alternatively, Qualys SSL Labs has a website that is very thorough and is commonly used for testing external servers. The report will show the cipher suites allowed along with many other details. https://www.ssllabs.com/ssltest/ The recommended cipher suites will start with `TLS_ECDHE_` or `TLS_DHE_` and have the initials FS at the end for forward secrecy.
- Alternatively find the specified values for the `SSLCipherSuite` directive in the Apache server level configuration and every virtual host that is SSL/TLS enabled. Then use the openssl command on the local system to verify the specified `SSLCipherSuite` directive only allows cipher suites that begin with the `ECDHE-` or `DHE-` algorithms. For example:

```
$ openssl ciphers -v
'EECDH:EDH:!NULL:!SSLv2:!RC4:!3DES:!IDEA:!aNULL:!SHA1'
ECDHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(256)
Mac=AEAD
ECDHE-ECDSA-AES256-GCM-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA
Enc=AESGCM(256) Mac=AEAD
ECDHE-RSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(256)
Mac=SHA384
ECDHE-ECDSA-AES256-SHA384 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(256)
Mac=SHA384
ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(128)
Mac=AEAD
ECDHE-ECDSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=ECDSA
Enc=AESGCM(128) Mac=AEAD
ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)
Mac=SHA256
ECDHE-ECDSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=ECDSA Enc=AES(128)
Mac=SHA256
DHE-DSS-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=DSS  Enc=AESGCM(256)
Mac=AEAD
DHE-RSA-AES256-GCM-SHA384 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(256)
Mac=AEAD
DHE-RSA-AES256-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AES(256)
Mac=SHA256
DHE-DSS-AES256-SHA256   TLSv1.2 Kx=DH       Au=DSS  Enc=AES(256)
Mac=SHA256
```

```
DHE-DSS-AES128-GCM-SHA256 TLSv1.2 Kx=DH       Au=DSS  Enc=AESGCM(128)
Mac=AEAD
DHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=DH       Au=RSA  Enc=AESGCM(128)
Mac=AEAD
DHE-RSA-AES128-SHA256   TLSv1.2 Kx=DH       Au=RSA  Enc=AES(128)
Mac=SHA256
DHE-DSS-AES128-SHA256   TLSv1.2 Kx=DH       Au=DSS  Enc=AES(128)
Mac=SHA256
```

Remediation:

Perform one of 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/TLS enabled:

```
SSLCipherSuite EECDH:EDH:!NULL:!SSLv2:!RC4:!aNULL:!3DES:!IDEA
```

- The more recent versions of openssl (such as 1.0.2 and newer) will support the usage of `ECDHE` as a synonym for `EECDH` and `DHE` as a synonym for `EDH` in the cipher specification. The usage of `ECDHE` and `DHE` are preferred so that the specification matches the expected output. So, the cipher specification could be:

```
SSLCipherSuite ECDHE:DHE:!NULL:!SSLv2:!RC4:!aNULL:!3DES:!IDEA
```

Default Value:

The default value for `SSLCipherSuite` depends on OpenSSL library version used.

References:

1. https://en.wikipedia.org/wiki/Forward_secrecy
2. https://scotthelme.co.uk/perfect-forward-secrecy/
3. https://www.owasp.org/index.php/TLS_Cipher_String_Cheat_Sheet

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

18.5 Use Only Standardized and Extensively Reviewed Encryption Algorithms
Use only standardized and extensively reviewed encryption algorithms.

8 Information Leakage

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

8.1 Ensure ServerTokens is Set to 'Prod' or 'ProductOnly' (Automated)

Profile Applicability:

- Level 1
- Level 2

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 details 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:
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:

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

```
ServerTokens Prod
```

Default Value:

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

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#servertokens

CIS Controls:

Version 6

18.9 Sanitize Deployed Software Of Development Artifacts
For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Version 7

14.7 Enforce Access Control to Data through Automated Tools
Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.

8.2 Ensure ServerSignature Is Not Enabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

Audit:

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

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

```
ServerSignature Off
```

Default Value:

The default value is `Off` for `ServerSignature`.

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#serversignature

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

13.2 Remove Sensitive Data or Systems Not Regularly Accessed by Organization
Remove sensitive data or systems not regularly accessed by the organization from the network. These systems shall only be used as stand alone systems (disconnected from the network) by the business unit needing to occasionally use the system or completely virtualized and powered off until needed.

8.3 Ensure All Default Apache Content Is Removed (Automated)

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 2.4. Many icons are used primarily for auto indexing, which is also recommended to be disabled.

Audit:

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

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 if present:

```
# 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>
```

Default Value:

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

CIS Controls:

Version 6

18.9 Sanitize Deployed Software Of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Version 7

13.2 Remove Sensitive Data or Systems Not Regularly Accessed by Organization

Remove sensitive data or systems not regularly accessed by the organization from the network. These systems shall only be used as stand alone systems (disconnected from the network) by the business unit needing to occasionally use the system or completely virtualized and powered off until needed.

8.4 Ensure ETag Response Header Fields Do Not Include Inodes (Automated)

Profile Applicability:

- Level 2

Description:

The `FileETag` directive configures the file attributes that are used to create the `ETag` (entity tag) response header field when the document is based on a static file. The `ETag` value is used in cache management to save network bandwidth. The value returned may be based on combinations of the file `inode`, the modification time, and the file size.

Rationale:

When the `FileETag` is configured to include the file `inode` number, remote attackers may be able to discern the inode number from returned values. The `inode` is considered sensitive information, as it could be useful in assisting in other attacks.

Audit:

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

For the server and all virtual host and directory configurations verify that either

1. The `FileETag` directive is not present, or
2. The configured `FileETag` value does not contain any of the values `all` or `inode` or `+inode`.

Remediation:

Perform the following to implement the recommended state:

Remove all instances of the `FileETag` directive. Alternatively, add or modify the `FileETag` directive in the server and each virtual host configuration to have either the value `None` or `MTime Size`.

Default Value:

The default value is `MTime Size`.

References:

1. http://httpd.apache.org/docs/2.4/mod/core.html#FileETag
2. https://nvd.nist.gov/vuln/detail/CVE-2003-1418

CIS Controls:

Version 6

18.9 Sanitize Deployed Software Of Development Artifacts

For in-house developed applications, ensure that development artifacts (sample data and scripts; unused libraries, components, debug code; or tools) are not included in the deployed software, or accessible in the production environment.

Version 7

13.2 Remove Sensitive Data or Systems Not Regularly Accessed by Organization

Remove sensitive data or systems not regularly accessed by the organization from the network. These systems shall only be used as stand alone systems (disconnected from the network) by the business unit needing to occasionally use the system or completely virtualized and powered off until needed.

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.

9.1 Ensure the TimeOut Is Set to 10 or Less (Automated)

Profile Applicability:

- Level 1
- Level 2

Description:

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

Rationale:

One common technique for DoS is to initiate many connections to the server. By decreasing the timeout for old connections and we allow the server to free up resources more quickly and be more responsive. By making the server more efficient, it will be more resilient to DoS conditions. The `Timeout` directive affects several timeout values for Apache, so review the Apache document carefully.
http://httpd.apache.org/docs/2.4/mod/core.html#timeout

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
```

Default Value:

```
Timeout 60
```

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#timeout

Additional Information:

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

9.2 Ensure KeepAlive Is Enabled (Automated)

Profile Applicability:

- Level 1
- Level 2

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
```

Default Value:

`KeepAlive On`

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#keepalive

Additional Information:

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

9.3 Ensure MaxKeepAliveRequests is Set to a Value of 100 or Greater (Automated)

Profile Applicability:

- Level 1
- Level 2

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.

Rationale:

The `MaxKeepAliveRequests` directive is important to be used to mitigate the risk of Denial of Service (DoS) attack technique by reducing the overhead imposed on the server. The `KeepAlive` directive must be enabled before it is effective. Enabling `KeepAlives` allows for multiple HTTP requests to be sent while keeping the same TCP connection alive. This reduces the overhead of having to setup and tear down TCP connections for each request. By making the server more efficient, it will be more resilient to DoS conditions.

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
```

Default Value:

```
MaxKeepAliveRequests 100
```

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#maxkeepaliverequests

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

9.4 Ensure KeepAliveTimeout is Set to a Value of 15 or Less (Automated)

Profile Applicability:

- Level 1
- Level 2

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:

The `KeepAliveTimeout` directive is used mitigate some of the risk, by requiring more effort for a successful DoS attack. By enabling `KeepAlive` and keeping the timeout relatively low for old connections and we allow the server to free up resources more quickly and be more responsive.

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 `5` 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
```

Default Value:

`KeepAliveTimeout 5`

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#keepalivetimeout

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

9.5 Ensure the Timeout Limits for Request Headers is Set to 40 or Less (Automated)

Profile Applicability:

- Level 1
- Level 2

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 relatively 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 as 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. For details, see: https://www.owasp.org/index.php/H.....t.....t....p.....p....o....s....t

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.

Remediation:

Perform the following to implement the recommended state:

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

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

2. 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:

`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. https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

9.6 Ensure Timeout Limits for the Request Body is Set to 20 or Less (Automated)

Profile Applicability:

- Level 1
- Level 2

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:

1. Locate the Apache configuration files and included configuration files.
2. Locate any `RequestReadTimeout` directives and verify the configuration has a maximum body request timeout of `20` 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 `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:

`body=20,MinRate=500`

References:

1. https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

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.

10.1 Ensure the LimitRequestLine directive is Set to 512 or less (Automated)

Profile Applicability:

- Level 2

Description:

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

Specifically, the `LimitRequestLine` directive limits the allowed size of a client's HTTP request-line, which consists of the HTTP method, URI, and protocol version.

Rationale:

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

Audit:

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

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:

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

```
LimitRequestline 512
```

Default Value:

`LimitRequestline 8190`

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#limitrequestline

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

10.2 Ensure the LimitRequestFields Directive is Set to 100 or Less (Automated)

Profile Applicability:

- Level 2

Description:

The `LimitRequestFields` directive limits the number of fields allowed in an HTTP request.

Rationale:

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

Audit:

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

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:

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
```

Default Value:

```
LimitRequestFields 100
```

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#limitrequestfields

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

10.3 Ensure the LimitRequestFieldsize Directive is Set to 1024 or Less (Automated)

Profile Applicability:

- Level 2

Description:

The `LimitRequestFieldSize` limits the number of bytes that will be allowed in an HTTP request header. It is recommended that the `LimitRequestFieldSize` directive be set to `1024` or less.

Rationale:

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

Audit:

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

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:

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

```
LimitRequestFieldsize 1024
```

Default Value:

```
LimitRequestFieldsize 8190
```

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#limitrequestfieldsize

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

10.4 Ensure the LimitRequestBody Directive is Set to 102400 or Less (Automated)

Profile Applicability:

- Level 2

Description:

The `LimitRequestBody` directive limits the number of bytes that are allowed in a request body. Size of requests may vary greatly; for example, during a file upload the size of the file must fit within this limit.

Rationale:

The limiting of the size of the request body is helpful so that the web server can prevent an unexpectedly long or large request from being passed to a potentially vulnerable program. Of course, the underlying dependency is that we need to set the limits high enough to not interfere with any one application on the server, while setting them low enough to be of value in protecting the applications. The `LimitRequestBody` may be configured on a per directory, or per location context. Please read the Apache documentation carefully, as these requests may interfere with the expected functionality of some web applications.

Audit:

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

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:

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

```
LimitRequestBody 102400
```

Default Value:

```
LimitRequestBody 0 (unlimited)
```

References:

1. https://httpd.apache.org/docs/2.4/mod/core.html#limitrequestbody

CIS Controls:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

11 Enable SELinux to Restrict Apache Processes

Recommendations in this section provide mandatory access controls (MAC) using the SELinux kernel module in targeted mode. SELinux provides additional enforced security which will prevent access to resources, files and directories by the httpd processes even in cases where an application or server vulnerability might allow inappropriate access. The SELinux controls are advanced security controls that require significant effort to ensure they do not negatively impact the application and/or site functionality. It is highly recommended that the configuration states described in this section be tested thoroughly on test servers prior to deploying them to production servers.

SELinux and AppArmor provide similar controls, and it is not recommended to use both SELinux and AppArmor on the same system. Depending on which Linux distribution is in use either AppArmor or SELinux are likely to be already installed or readily available as packages. AppArmor differs from SELinux in that it binds the controls to programs rather than users and uses path names rather than labeled type enforcement.

11.1 Ensure SELinux Is Enabled in Enforcing Mode (Automated)

Profile Applicability:

- Level 2

Description:

SELinux (Security-Enhanced Linux) is a Linux kernel security module that provides mandatory access control security policies with type enforcement that are checked after the traditional discretionary access controls. It was created by the US National Security Agency and can enforce rules on files and processes in a Linux system, and restrict actions, based on defined policies.

Rationale:

Web applications and services continue to be one of the leading attack vectors for black-hat criminals to gain access to information and servers. The threat is high because web servers are often externally accessible and typically have the greatest share of server-side vulnerabilities. The SELinux mandatory access controls provide a much stronger security model which can be used to implement a deny-by-default model which only allows what is explicitly permitted.

Audit:

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

Use the `sestatus` command to check that SELinux is enabled and that both the current mode and the configured mode are set to `enforcing`.

```
$ sestatus | grep -i mode
Current mode: enforcing
Mode from config file: enforcing
```

Remediation:

Perform the following to implement the recommended state:

If SELinux is not enabled in the configuration file, edit the file `/etc/selinux/config` and set the value of SELINUX as `enforcing` and reboot the system for the new configuration to be effective.

```
SELINUX=enforcing
```

If the current mode is not enforcing, and an immediate reboot is not possible, the current mode can be set to enforcing with the `setenable` command shown below.

```
# setenforce 1
```

Default Value:

SELinux is not enabled by default.

References:

1. https://en.wikipedia.org/wiki/Security-Enhanced_Linux

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

14.7 Enforce Access Control to Data through Automated Tools

Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.

11.2 Ensure Apache Processes Run in the httpd_t Confined Context (Automated)

Profile Applicability:

- Level 2

Description:

SELinux includes customizable targeted policies that may be used to confine the Apache httpd server to enforce least privileges so that the httpd server has only the minimal access to specified directories, files and network ports. Access is controlled by process types (domains) defined for the httpd process. There are over a hundred individual httpd related types defined in a default Apache SELinux policy which includes many of the common Apache add-ons and applications such as php, nagios, smokeping and many others. The default SELinux policies work well for a default Apache installation, but implementation of SELinux targeted polices on a complex or highly customized web server requires a rather significant development and testing effort which comprehends both the workings of SELinux and the detailed operations and requirements of the web application.

All directories and files to be accessed by the web server process must have security labels with appropriate types. The following types are a sample of the most commonly used:

- `http_port_t` - Network ports allowed for listening
- `httpd_sys_content_t` - Read access to directories and files with web content
- `httpd_log_t` - Directories and files to be used for writable log data
- `httpd_sys_script_exec_t` - Directories and files for executable content.

Rationale:

With the proper implementation of SELinux, vulnerabilities in the web application may be prevented from being exploited due to the additional restrictions. For example, a vulnerability that allows an attacker to read inappropriate system files may be prevented from execution by SELinux because the inappropriate files are not labeled as `httpd_sys_content_t`. Likewise writing to an unexpected directory or execution of unexpected content can be prevented by similar mandatory security labels enforced by SELinux.

Audit:

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

Check that all of the Apache httpd processes are confined to the `httpd_t` SELinux context. The type (the third colon separated field) for each process should be `httpd_t`. Note that on some platforms, such as Ubuntu, the Apache executable is named apache2 instead of httpd.

```
$ ps -eZ | grep httpd
unconfined_u:system_r:httpd_t:s0 1366 ? 00:00:00 httpd
unconfined_u:system_r:httpd_t:s0 1368 ? 00:00:00 httpd
. . .
```

Remediation:

If the running httpd processes are not confined to the `httpd_t` SELinux context. Then check the context for the `httpd` binary and the `apachectl` binary and set the httpd binary to have a context of `httpd_exec_t` and the `apachectl` executable should have a context of `initrc_exec_t` as shown below. Also note that on some platforms such as Ubuntu, the Apache executable is named apache2 instead of httpd.

```
# ls -alZ /usr/sbin/httpd /usr/sbin/httpd.* /usr/sbin/apachectl
-rwxr-xr-x. root root system_u:object_r:initrc_exec_t:s0 /usr/sbin/apachectl
-rwxr-xr-x. root root system_u:object_r:httpd_exec_t:s0 /usr/sbin/httpd
-rwxr-xr-x. root root system_u:object_r:httpd_exec_t:s0
/usr/sbin/httpd.worker
-rwxr-xr-x. root root system_u:object_r:httpd_exec_t:s0 /usr/sbin/httpd.event
```

If the executable files are not labeled correctly, they may be relabeled with the `chcon` command, as shown, however the file system labeling is based on the SELinux file context polices and the file systems will on some occasions be relabeled according to the policy.

```
# chcon -t initrc_exec_t /usr/sbin/apachectl
# chcon -t httpd_exec_t /usr/sbin/httpd /usr/sbin/httpd.*
```

Since the file system may be relabeled based on SELinux policy, it's best to check the SELinux policy with `semanage fcontext -l` option. If the policy is not present, then add the pattern to the policy using the `-a` option. The `restorecon` command shown below will restore the file context label according to the current policy, which is required if a pattern was added.

```
# ### Check the Policy
# semanage fcontext -l | fgrep 'apachectl'
/usr/sbin/apachectl regular file system_u:object_r:initrc_exec_t:s0
# semanage fcontext -l | fgrep '/usr/sbin/httpd'
/usr/sbin/httpd regular file system_u:object_r:httpd_exec_t:s0
/usr/sbin/httpd.worker regular file system_u:object_r:httpd_exec_t:s0
/usr/sbin/httpd.event regular file system_u:object_r:httpd_exec_t:s0
# ### Add to the policy, if not present
# semanage fcontext -f -- -a -t httpd_exec_t '/usr/sbin/httpd'
# semanage fcontext -f -- -a -t httpd_exec_t '/usr/sbin/httpd.worker'
# semanage fcontext -f -- -a -t httpd_exec_t '/usr/sbin/httpd.event'
```

```
# semanage fcontext -f -- -a -t initrc_exec_t /usr/sbin/apachectl

# ### Restore the file labeling accord to the SELinux policy
# restorecon -v /usr/sbin/httpd /usr/sbin/httpd.* /usr/sbin/apachectl
```

Default Value:

SELinux is not enabled by default.

References:

1. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/chap-Security-Enhanced_Linux-Targeted_Policy.html

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

11.3 Ensure the httpd_t Type is Not in Permissive Mode (Automated)

Profile Applicability:

- Level 2

Description:

In addition to setting the entire SELinux configuration in permissive mode, it is possible to set individual process types (domains) such as `httpd_t` into a permissive mode as well. The permissive mode will not prevent any access or actions, instead, any actions that would have been denied are simply logged.

Rationale:

Usage of the permissive mode is helpful for testing and ensuring that SELinux will not prevent access that is necessary for the proper function of a web application. However, all access is allowed in permissive mode by SELinux.

Audit:

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

Check that the `httpd_t` process type (domain) is not in permissive mode with the `semodule` command. There should be no output if the type is not set to permissive.

```
# semodule -l | grep permissive_httpd_t
```

Remediation:

Perform the following to implement the recommended state:

If the `httpd_t` type is in permissive mode; the customized permissive mode should be deleted with the following `semanage` command.

```
# semanage permissive -d httpd_t
```

Default Value:

The `httpd_t` type is not in permissive mode by default.

References:

1. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/sect-Security-Enhanced_Linux-Fixing_Problems-Permissive_Domains.html

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

11.4 Ensure Only the Necessary SELinux Booleans are Enabled (Manual)

Profile Applicability:

- Level 2

Description:

SELinux booleans allow or disallow behavior specific to the Apache web server. Common examples include whether CGI execution is allowed, or if the httpd server is allowed to communicate with the current terminal (`tty`). Communication with the terminal, may be necessary for entering a passphrase during start up to decrypt a private key.

Rationale:

Enabling only the necessary httpd related booleans provides a defense in depth approach, that will deny actions that are not in use or expected.

Audit:

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

Review the SELinux httpd booleans that are enabled to ensure only the necessary booleans are enabled for the current and the configured state. Due to the variety and complexity of web server usages and organizational needs, a preset recommendation of enabled booleans is not practical. Run either of the two commands below to show only the enabled httpd related booleans. The `getsebool` command is installed with the core SELinux, while the `semanage` command is an optional package; however, the `semanage` output includes descriptive text.

```
# getsebool -a | grep httpd_ | grep '> on'
httpd_builtin_scripting --> on
httpd_dbus_avahi --> on
httpd_tty_comm --> on
httpd_unified --> on
```

Alternative using the `semanage` command.

```
# semanage boolean -l | grep httpd_ | grep -v '(off , off)'
httpd_enable_cgi (on , on) Allow httpd cgi support
httpd_dbus_avahi (on , on) Allow Apache to communicate with avahi service via
dbus
httpd_unified (on , on) Unify HTTPD handling of all content files.
httpd_builtin_scripting (on , on) Allow httpd to use built in scripting
(usually php)
httpd_tty_comm (on , on) Unify HTTPD to communicate with the terminal...
```

Remediation:

Perform the following to implement the recommended state:

To disable the SELinux httpd booleans that are determined to be unnecessary, use the `setsebool` command as shown below with the `-P` option to make the change persistent.

```
# setsebool -P httpd_enable_cgi off
# getsebool httpd_enable_cgi
httpd_enable_cgi --> off
```

Default Value:

SELinux is not enabled by default.

References:

1. https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/sect-Security-Enhanced_Linux-Working_with_SELinux-Booleans.html

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

12 Enable AppArmor to Restrict Apache Processes

Recommendations in this section provide mandatory access controls (MAC) using the AppArmor kernel module. AppArmor provides additional enforced security which will prevent access to resources, files and directories by the apache2 processes even in cases where an application or server vulnerability might allow inappropriate access. The AppArmor controls are advanced security controls that require significant effort to ensure they do not negatively impact the application and/or site functionality. It is highly recommended that the configuration states described in this section be tested thoroughly on test servers prior to deploying them to production servers.

AppArmor and SELinux provide similar controls, and it is not recommended to use both SELinux and AppArmor on the same system. Depending on which Linux distribution is in use either AppArmor or SELinux are likely to be already installed or readily available as packages. AppArmor differs from SELinux in that it binds the controls to programs rather than users and uses path names rather than labeled type enforcement.

12.1 Ensure the AppArmor Framework Is Enabled (Automated)

Profile Applicability:

- Level 2

Description:

AppArmor is a Linux kernel security module that provides a named based mandatory access control with security policies. AppArmor can enforce rules on programs for file access and network connections and restrict actions based on defined policies.

Rationale:

Web applications and web services continue to be one of the leading attack vectors for black-hat criminals to gain access to information and servers. The threat is high because web servers are often externally accessible and typically have the greatest share of server-side vulnerabilities. The AppArmor mandatory access controls provide a much stronger security model which can be used to implement a deny-by-default model which only allows what is explicitly permitted.

Audit:

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

Use the aa-status command with the `--enabled` option to check that AppArmor is enabled. If AppArmor is enabled the command will return a zero (`0`) exit code for success. The `&& echo Enabled` is added to the command below to provide positive feedback. If no text is echoed, then AppArmor is not enabled.

```
# aa-status --enabled && echo Enabled
Enabled
```

Remediation:

Perform the following to implement the recommended state:

- If the `aa-status` command is not found, then the AppArmor package is not installed and needs to be installed using the appropriate the Linux distribution package management. For example:

```
# apt-get install apparmor
# apt-get install libapache2-mod-apparmor
```

- To enable the AppArmor framework run the `init.d` script as shown below.

```
# /etc/init.d/apparmor start
```

Default Value:

AppArmor is enabled by default.

References:

1. https://help.ubuntu.com/community/AppArmor

CIS Controls:

Version 6

2.2 Deploy Application Whitelisting

Deploy application whitelisting technology that allows systems to run software only if it is included on the whitelist and prevents execution of all other software on the system. The whitelist may be very extensive (as is available from commercial whitelist vendors), so that users are not inconvenienced when using common software. Or, for some special-purpose systems (which require only a small number of programs to achieve their needed business functionality), the whitelist may be quite narrow.

2.7 Utilize Application Whitelisting

Utilize application whitelisting technology on all assets to ensure that only authorized software executes and all unauthorized software is blocked from executing on assets.

12.2 Ensure the Apache AppArmor Profile Is Configured Properly (Manual)

Profile Applicability:

- Level 2

Description:

AppArmor includes customizable profiles that may be used to confine the Apache web server to enforce least privileges so that the server has only the minimal access to specified directories, files and network ports. Access is controlled by a profile defined for the apache2 process. The default AppArmor profile is typically a very permissive profile that allows read-write access to all system files. Therefore, it's important that the default profile be customized to enforce least privileges. The AppArmor utilities such as `aa-autodep`, `aa-complain`, and `aa-logprof` can be used to generate an initial profile based on actual usage. However thorough testing, review and customization will be necessary to ensure that the Apache profile restrictions allow necessary functionality while implementing least privilege.

Rationale:

With the proper implementation of AppArmor profile, vulnerabilities in the web application may be prevented from being exploited due to the additional restrictions. For example, a vulnerability that allows an attacker to read an inappropriate system files may be prevented from execution by AppArmor because the inappropriate files are not allowed by the profile. Likewise writing to an unexpected directory or execution of unexpected content can be prevented by similar mandatory security controls enforced by AppArmor.

Audit:

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

1. Find the Apache AppArmor profile typically found in `/etc/apparmor.d/usr.sbin.apache2` along with any files included by the profile such as `/etc/apparmor.d/apache2.d/*` and files in the `/etc/apparmor.d/abstractions/` directory.
2. Review the capabilities and permissions granted to ensure that the profile implements least privileges for the web application. Wild-card paths such as `/**`, which grant access to all files and directories starting with the root level directory, should not be present in the profile. Instead read only access to specific necessary system files such `/etc/group` and to the web content files such as `/var/www/html/**`

should be given. Refer to the `apparmor.d` man page for additional details. Shown below are some possible example capabilities and path permissions.

```
capability dac_override,
capability dac_read_search,
capability net_bind_service,
capability setgid,
capability setuid,
capability kill,
capability sys_tty_config,
. . .

/usr/sbin/apache2 mr,
/etc/gai.conf r,
/etc/group r,
/etc/apache2/** r,
/var/www/html/** r,
/run/apache2/** rw,
/run/lock/apache2/** rw,
/var/log/apache2/** rw,
/etc/mime.types r,
```

Remediation:

Perform the following to implement the recommended state:

1. Stop the Apache server

```
# service apache2 stop
```

2. Create a mostly empty apache2 profile based on program dependencies.

```
# aa-autodep apache2
Writing updated profile for /usr/sbin/apache2.
```

3. Set the apache2 profile in complain mode so that access violations will be allowed and logged.

```
# aa-complain apache2
Setting /usr/sbin/apache2 to complain mode.
```

4. Start the apache2 service

```
# service apache2 start
```

5. Thoroughly test the web application attempting to exercise all intended functionality so that AppArmor will generate the necessary logs of all resources accessed. The logs are sent via the system syslog utility and are typically found in

either the `/var/log/syslog` or `/var/log/messages` files. Also stop and restart the web server as part of the testing process.

6. Use `aa-logprof` to update the profile based on logs generated during the testing. The tool will prompt for suggested modifications to the profile, based on the logs. The logs may also be reviewed manually in order to update the profile.

```
# aa-logprof
```

7. Review and edit the profile, removing any inappropriate content, and adding appropriate access rules. Directories with multiple files accessed with the same permission can be simplified with the usage of wild-cards when appropriate. Reload the updated profile using the `apparmor_parser` command.

```
# apparmor_parser -r /etc/apparmor.d/usr.sbin.apache2
```

8. Test the new updated profile again and check for any new AppArmor denied logs generated. Update and reload the profile as necessary. Repeat the application tests, until no new AppArmor deny logs are created, except for access which should be prohibited.

```
# tail -f /var/log/syslog
```

9. Set the apache2 profile to enforce mode, reload AppArmor, and then test the web site functionality again.

```
# aa-enforce /usr/sbin/apache2
# /etc/init.d/apparmor reload
```

Default Value:

The default Apache profile is very permissive.

References:

1. https://wiki.ubuntu.com/AppArmor

CIS Controls:

Version 6

2 Inventory of Authorized and Unauthorized Software
Inventory of Authorized and Unauthorized Software

14.7 Enforce Access Control to Data through Automated Tools

Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.

12.3 Ensure Apache AppArmor Profile is in Enforce Mode (Automated)

Profile Applicability:

- Level 2

Description:

AppArmor profiles may be in one of three modes: disabled, complain or enforce. In the `complain` mode, any violations of the access controls are logged but the restrictions are not enforced. Also, once a profile mode has been changed, it is recommended to restart the Apache server, otherwise the currently running process may not be confined by the policy.

Rationale:

The `complain` mode is useful for testing and debugging a profile, but is not appropriate for production. Only the confined process running in enforce mode will prevent attacks that violate the configured access controls.

Audit:

Perform the following steps to determine if the recommended state is implemented:
Use the `aa-unconfined` command to check that the apache2 policy is enforced, and that the currently running apache2 processes are confined. The output should include both `confined by` and `(enforce)`

```
# aa-unconfined --paranoid | grep apache2
1899 /usr/sbin/apache2 confined by '/usr/sbin/apache2 (enforce)'
1902 /usr/sbin/apache2 confined by '/usr/sbin/apache2 (enforce)'
1903 /usr/sbin/apache2 confined by '/usr/sbin/apache2 (enforce)'
. . .
```

Note: Non-compliant results may include `not confined` or `(complain)` such as the following:

```
3304 /usr/sbin/apache2 not confined
2502 /usr/sbin/apache2 confined by '/usr/sbin/apache2 (complain)'
4004 /usr/sbin/apache2 confined by
'/usr/sbin/apache2//HANDLING_UNTRUSTED_INPUT (complain)'
```

Remediation:

Perform the following to implement the recommended state:

1. Set the profile state to enforce mode.

```
# aa-enforce apache2
Setting /usr/sbin/apache2 to enforce mode.
```

2. Stop the Apache server and confirm that is it not running. In some cases, the AppArmor controls may prevent the web server from stopping properly, and it may be necessary to stop the process manually or even reboot the server.

```
# service apache2 stop
 * Stopping web server apache2
# service apache2 status
 * apache2 is not running
```

3. Restart the Apache service.

```
# service apache2 start
 * Starting web server apache2
```

Default Value:

The default mode is `enforce`.

CIS Controls:

Version 6

2.2 Deploy Application Whitelisting

Deploy application whitelisting technology that allows systems to run software only if it is included on the whitelist and prevents execution of all other software on the system. The whitelist may be very extensive (as is available from commercial whitelist vendors), so that users are not inconvenienced when using common software. Or, for some special-purpose systems (which require only a small number of programs to achieve their needed business functionality), the whitelist may be quite narrow.

Version 7

2.7 Utilize Application Whitelisting

Utilize application whitelisting technology on all assets to ensure that only authorized software executes and all unauthorized software is blocked from executing on assets.

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Planning and Installation**		
1.1	Ensure the Pre-Installation Planning Checklist Has Been Implemented (Manual)	☐	☐
1.2	Ensure the Server Is Not a Multi-Use System (Manual)	☐	☐
1.3	Ensure Apache Is Installed From the Appropriate Binaries (Manual)	☐	☐
2	**Minimize Apache Modules**		
2.1	Ensure Only Necessary Authentication and Authorization Modules Are Enabled (Manual)	☐	☐
2.2	Ensure the Log Config Module Is Enabled (Automated)	☐	☐
2.3	Ensure the WebDAV Modules Are Disabled (Automated)	☐	☐
2.4	Ensure the Status Module Is Disabled (Automated)	☐	☐
2.5	Ensure the Autoindex Module Is Disabled (Automated)	☐	☐
2.6	Ensure the Proxy Modules Are Disabled (Automated)	☐	☐
2.7	Ensure the User Directories Module Is Disabled (Automated)	☐	☐
2.8	Ensure the Info Module Is Disabled (Automated)	☐	☐
2.9	Ensure the Basic and Digest Authentication Modules are Disabled (Automated)	☐	☐
3	**Principles, Permissions, and Ownership**		
3.1	Ensure the Apache Web Server Runs As a Non-Root User (Automated)	☐	☐
3.2	Ensure the Apache User Account Has an Invalid Shell (Automated)	☐	☐
3.3	Ensure the Apache User Account Is Locked (Automated)	☐	☐
3.4	Ensure Apache Directories and Files Are Owned By Root (Automated)	☐	☐
3.5	Ensure the Group Is Set Correctly on Apache Directories and Files (Automated)	☐	☐
3.6	Ensure Other Write Access on Apache Directories and Files Is Restricted (Automated)	☐	☐
3.7	Ensure the Core Dump Directory Is Secured (Automated)	☐	☐
3.8	Ensure the Lock File Is Secured (Automated)	☐	☐
3.9	Ensure the Pid File Is Secured (Automated)	☐	☐
3.10	Ensure the ScoreBoard File Is Secured (Automated)	☐	☐
3.11	Ensure Group Write Access for the Apache Directories and Files Is Properly Restricted (Automated)	☐	☐

3.12	Ensure Group Write Access for the Document Root Directories and Files Is Properly Restricted (Automated)	☐	☐
3.13	Ensure Access to Special Purpose Application Writable Directories is Properly Restricted (Manual)	☐	☐
4	**Apache Access Control**		
4.1	Ensure Access to OS Root Directory Is Denied By Default (Automated)	☐	☐
4.2	Ensure Appropriate Access to Web Content Is Allowed (Manual)	☐	☐
4.3	Ensure OverRide Is Disabled for the OS Root Directory (Automated)	☐	☐
4.4	Ensure OverRide Is Disabled for All Directories (Automated)	☐	☐
5	**Minimize Features, Content and Options**		
5.1	Ensure Options for the OS Root Directory Are Restricted (Automated)	☐	☐
5.2	Ensure Options for the Web Root Directory Are Restricted (Automated)	☐	☐
5.3	Ensure Options for Other Directories Are Minimized (Automated)	☐	☐
5.4	Ensure Default HTML Content Is Removed (Automated)	☐	☐
5.5	Ensure the Default CGI Content printenv Script Is Removed (Automated)	☐	☐
5.6	Ensure the Default CGI Content test-cgi Script Is Removed (Automated)	☐	☐
5.7	Ensure HTTP Request Methods Are Restricted (Automated)	☐	☐
5.8	Ensure the HTTP TRACE Method Is Disabled (Automated)	☐	☐
5.9	Ensure Old HTTP Protocol Versions Are Disallowed (Automated)	☐	☐
5.10	Ensure Access to .ht* Files Is Restricted (Automated)	☐	☐
5.11	Ensure Access to Inappropriate File Extensions Is Restricted (Automated)	☐	☐
5.12	Ensure IP Address Based Requests Are Disallowed (Automated)	☐	☐
5.13	Ensure the IP Addresses for Listening for Requests Are Specified (Automated)	☐	☐
5.14	Ensure Browser Framing Is Restricted (Automated)	☐	☐
6	**Operations - Logging, Monitoring and Maintenance**		
6.1	Ensure the Error Log Filename and Severity Level Are Configured Correctly (Automated)	☐	☐
6.2	Ensure a Syslog Facility Is Configured for Error Logging (Automated)	☐	☐
6.3	Ensure the Server Access Log Is Configured Correctly (Automated)	☐	☐

6.4	Ensure Log Storage and Rotation Is Configured Correctly (Automated)	☐	☐
6.5	Ensure Applicable Patches Are Applied (Automated)	☐	☐
6.6	Ensure ModSecurity Is Installed and Enabled (Automated)	☐	☐
6.7	Ensure the OWASP ModSecurity Core Rule Set Is Installed and Enabled (Automated)	☐	☐
7	**SSL/TLS Configuration**		
7.1	Ensure mod_ssl and/or mod_nss Is Installed (Automated)	☐	☐
7.2	Ensure a Valid Trusted Certificate Is Installed (Automated)	☐	☐
7.3	Ensure the Server's Private Key Is Protected (Automated)	☐	☐
7.4	Ensure the TLSv1.0 and TLSv1.1 Protocols are Disabled (Automated)	☐	☐
7.5	Ensure Weak SSL/TLS Ciphers Are Disabled (Automated)	☐	☐
7.6	Ensure Insecure SSL Renegotiation Is Not Enabled (Automated)	☐	☐
7.7	Ensure SSL Compression is not Enabled (Automated)	☐	☐
7.8	Ensure Medium Strength SSL/TLS Ciphers Are Disabled (Automated)	☐	☐
7.9	Ensure All Web Content is Accessed via HTTPS (Automated)	☐	☐
7.10	Ensure OCSP Stapling Is Enabled (Automated)	☐	☐
7.11	Ensure HTTP Strict Transport Security Is Enabled (Automated)	☐	☐
7.12	Ensure Only Cipher Suites That Provide Forward Secrecy Are Enabled (Automated)	☐	☐
8	**Information Leakage**		
8.1	Ensure ServerTokens is Set to 'Prod' or 'ProductOnly' (Automated)	☐	☐
8.2	Ensure ServerSignature Is Not Enabled (Automated)	☐	☐
8.3	Ensure All Default Apache Content Is Removed (Automated)	☐	☐
8.4	Ensure ETag Response Header Fields Do Not Include Inodes (Automated)	☐	☐
9	**Denial of Service Mitigations**		
9.1	Ensure the TimeOut Is Set to 10 or Less (Automated)	☐	☐
9.2	Ensure KeepAlive Is Enabled (Automated)	☐	☐
9.3	Ensure MaxKeepAliveRequests is Set to a Value of 100 or Greater (Automated)	☐	☐
9.4	Ensure KeepAliveTimeout is Set to a Value of 15 or Less (Automated)	☐	☐
9.5	Ensure the Timeout Limits for Request Headers is Set to 40 or Less (Automated)	☐	☐
9.6	Ensure Timeout Limits for the Request Body is Set to 20 or Less (Automated)	☐	☐
10	**Request Limits**		

10.1	Ensure the LimitRequestLine directive is Set to 512 or less (Automated)	☐	☐
10.2	Ensure the LimitRequestFields Directive is Set to 100 or Less (Automated)	☐	☐
10.3	Ensure the LimitRequestFieldsize Directive is Set to 1024 or Less (Automated)	☐	☐
10.4	Ensure the LimitRequestBody Directive is Set to 102400 or Less (Automated)	☐	☐
11	**Enable SELinux to Restrict Apache Processes**		
11.1	Ensure SELinux Is Enabled in Enforcing Mode (Automated)	☐	☐
11.2	Ensure Apache Processes Run in the httpd_t Confined Context (Automated)	☐	☐
11.3	Ensure the httpd_t Type is Not in Permissive Mode (Automated)	☐	☐
11.4	Ensure Only the Necessary SELinux Booleans are Enabled (Manual)	☐	☐
12	**Enable AppArmor to Restrict Apache Processes**		
12.1	Ensure the AppArmor Framework Is Enabled (Automated)	☐	☐
12.2	Ensure the Apache AppArmor Profile Is Configured Properly (Manual)	☐	☐
12.3	Ensure Apache AppArmor Profile is in Enforce Mode (Automated)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
Oct 31, 2020	2.0.0	Major Release