



the CENTER for
INTERNET SECURITY

Apache Benchmark for Unix

For Apache Versions 1.3 and 2.0
Levels I and II

`cis-feedback@lists.cisecurity.org`

Agreed Terms of Use

Background.

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

No representations, warranties and covenants.

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

User agreements.

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

1. No network, system, device, hardware, software or component can be made fully secure;
2. We are using the Products and the Recommendations solely at our own risk;
3. We are not compensating CIS to assume any liabilities associated with our use of the Products or the Recommendations, even risks that result from CIS’s negligence or failure to perform;
4. We have the sole responsibility to evaluate the risks and benefits of the Products and Recommendations to us and to adapt the Products and the Recommendations to our particular circumstances and requirements;
5. Neither CIS, nor any CIS Party (defined below) has any responsibility to make any corrections, updates, upgrades or bug fixes or to notify us if it chooses at its sole option to do so; and
6. Neither CIS nor any CIS Party has or will have any liability to us whatsoever (whether based in contract, tort, strict liability or otherwise) for any direct, indirect, incidental, consequential, or special damages (including without limitation loss of profits, loss of sales, loss of or damage to reputation, loss of customers, loss of software, data, information or emails, loss of privacy, loss of use of any computer or other equipment, business interruption, wasted management or other staff resources or claims of any kind against us from third parties) arising out of or in any way connected with our use of or our inability to use any of the Products or Recommendations (even if CIS has been advised of the possibility of such damages), including without limitation any liability associated with infringement of intellectual property, defects, bugs, errors, omissions, viruses, worms, backdoors, Trojan horses or other harmful items.

Grant of limited rights.

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

1. Except to the extent that we may have received additional authorization pursuant to a written agreement with CIS, each user may download, install and use each of the Products on a single computer;
2. Each user may print one or more copies of any Product or any component of a Product that is in a .txt, .pdf, .doc, .mcw, or .rtf format, provided that all such copies are printed in full and are kept intact, including without limitation the text of this Agreed Terms of Use in its entirety.

Retention of intellectual property rights; limitations on distribution.

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

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

Special rules.

The distribution of the NSA Security Recommendations is subject to the terms of the NSA Legal Notice and the terms contained in the NSA Security Recommendations themselves (<http://nsa2.www.conxion.com/cisco/notice.htm>).

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

Choice of law; jurisdiction; venue.

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

Introduction

This document provides a security benchmark consensus from The Center for Internet Security ("CIS") for securing Apache web servers on Unix operating systems. While much of the information in this benchmark can be applied to Apache servers on Microsoft Windows-based operating systems, emphasis is on Unix installations such as Linux, Sun Solaris, and HP-UX, due to significant differences in directory structure, directory permissions, and source compilation. A future CIS benchmark may be dedicated exclusively to Apache running on Windows-based architectures.

This benchmark document covers both Apache 1.3.XX and 2.0.XX versions. The example screen shot sections are assuming the Apache 2.2.2 version. The platform used for the examples in this document is Fedora Core 5, therefore all of the OS level commands are Linux specific. If you are using a different Unix OS, you will need to make sure that you use the correct flags, etc... for your OS.

This Benchmark document defines both Level 1 and Level 2 benchmark settings. These settings are designed primarily to enhance the security of the web server itself. Level 1 benchmarks are considered to be minimum and essential requirements. Level 2 benchmarks are more advanced settings and may not apply in all situations. It is left to the discretion of the reader to determine the relevance of each setting as it applies to their web environment. Please review both the Level I and Level II sections entirely prior to implementing the benchmark. Many of the security issues discussed have multiple mitigation strategies, which can be addressed by either a Level I or Level II setting.

Emphasis for this benchmark is on high-security (vs. ease of use or installation) and assumes static vs. dynamic web pages. This document focuses on the security of the Apache web server (which resides in the HTTP Presentation Tier – communication between an http client and the web server) and does not cover "secure coding" practices (such as PERL/PHP CGI script creation) and/or Web Application security issues (such as Java).

For Web Application security issues, visit the Open Web Application Security Project (OWASP) website - <http://www.owasp.org> and the Web Application Security Consortium <http://www.webappsec.org/>

Useful Related Resources

- **Apache Website:**
<http://www.apache.org>
- **Apache Security Tips:**
http://httpd.apache.org/docs/2.0/misc/security_tips.html
- **SANS/FBI Top 20 Vulnerabilities: Apache**
<http://www.sans.org/top20/>
- **Apache Server Security**
<http://www.cgisecurity.com/webserver/apache/>

Apache Vulnerability Resources

- **CERT Apache Advisories**
<http://search.cert.org/query.html?col=certadv&col=vulnotes&qt=apache&charset=iso-8859-1>
- **CVE Mitre Search**
<http://www.cve.mitre.org/cgi-bin/cvekey.cgi?keyword=apache>
- **SecurityFocus Vulnerabilities Search**
<http://www.securityfocus.com/bid>

It is the intent of this benchmark to be applicable for all major Unix operating systems. Users running the benchmark on Unix systems should verify command syntax, using the Unix "man" command, before executing commands on their systems.

While experienced Apache/Web administrators will find the Apache Benchmark to be a valuable technical resource in their arsenal, the benchmark is especially intended for those organizations that lack the resources to train, or those without technically advanced web security administrators. The individuals with responsibility for web security in those organizations often report that they have not corrected many of these flaws because they simply do not know which vulnerabilities are most dangerous, they are too busy to correct them all, or they do not know how to correct them safely. Traditionally, auditors and security managers have used vulnerability scanners to search for five hundred or a thousand or even two thousand very specific vulnerabilities, blunting the focus administrators need to ensure that all web servers are protected against the most common attacks. Unfortunately, most current web scanners do not do much more than a simple CGI script check. When a web administrator then receives a report listing the web vulnerabilities identified in a scan, they are often left to the inaccurate conclusion that by simply removing the default CGI scripts, they will become secure. This is far from the truth and this document will prove this fact.

Common Web Vulnerabilities Addressed By The Apache Benchmark

- Buffer Overflow Attacks
- Denial of Service
- Attacks on vulnerable scripts
- URL Manipulation
- Sniffing/Spoofing Credentials
- Client Parameter Manipulation
- Brute Force Attacks
- Web Server Fingerprinting
- Web defacements

Notes For Readers:

CIS Apache Benchmark for Unix Complies with the following Security Documents

- NIST Special Publication 800-44
["Guidelines on Securing Public Web Servers"](#)

Apache Benchmark Pre-configuration Checklist

One of the key benefits of the CIS Benchmarks are the Scoring Tools. Ideally, the accompanying tool can score every section of a benchmark. When dealing with complex applications such as web servers, it becomes more difficult to evaluate and score key components to the overall security of web servers.

It is important to realize that “Web Security” extends beyond the Web Server itself. There are many different web security vulnerabilities, which do not directly involve the web server itself. In order to truly secure a web infrastructure, many different information technology divisions must work together. These include, but are not limited to Firewalls, Intrusion Detection Systems, DNS, Networks Branch, etc...Take the time to build relationships with these groups and discuss web security issues. Hopefully, you will be able to identify deficiencies in your environment and fix them prior to exploitation attempts.

The benchmark reader should complete this checklist prior to applying the CIS Apache Benchmark.

Check	Description
<input type="checkbox"/>	Reviewed and implement my company's security policies as they relate to web security.
<input type="checkbox"/>	Implemented a secure network infrastructure by controlling access to/from your web server by using: Firewalls, Routers and Switches.
<input type="checkbox"/>	Implemented a Network Intrusion Detection System to monitor attacks against the web server.
<input type="checkbox"/>	Patched servers.
<input type="checkbox"/>	Implemented load-balancing/failover capability in case of Denial of Service or server shutdown
<input type="checkbox"/>	Educated developers about writing secure code.
<input type="checkbox"/>	Implemented a log rotation mechanism.
<input type="checkbox"/>	Implemented a disk space monitoring process.
<input type="checkbox"/>	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).
<input type="checkbox"/>	Our Domain Name Service (DNS) servers have been properly secured to prevent domain hi-jacking via cache poisoning, etc...

LEVELS I AND II	1
APACHE BENCHMARK PRE-CONFIGURATION CHECKLIST	6
LEVEL I -- APACHE BENCHMARK SETTINGS	9
L1 1. Harden the Underlying Operating System	9
L1 2. Install Apache Web Server	10
L1 3. Create the Web Groups	10
L1 4. Create the Apache Web User Account	11
L1 5. Lock Down the Apache Web User Account	11
L1 6. Subscribe to the Appropriate Security Advisories	12
L1 7. Apply Current Patches	12
L1 8. User Oriented General Directives	13
L1 9. Disable Unnecessary Apache Modules	14
L1 10. Denial of Service (DoS) Protective General Directives	15
L1 11. Web Server Software Obfuscation General Directives	16
L1 12. Mod_Security	18
L1 13. Access Control Directives	20
L1 14. Authentication Mechanisms	22
L1 15. Directory Functionality/Features Directives	23
L1 16. Limiting HTTP Request Methods	25
L1 17. Logging General Directives	26
L1 18. Remove Default/Unneeded Apache Files	27
L1 19. Updating Ownership and Permissions for Enhanced Security	29
L1 20. Implementing Secure Socket Layer (SSL) with Mod_SSL	30

L1 21. Deny HTTP TRACE Requests with Mod_Rewrite	33
LEVEL II -- PRUDENT SECURITY BEYOND THE MINIMUM LEVEL	34
L2 1. Create Web User Account Disk Quota	34
L2 2. Prevent the Web Server from Accessing OS Commands	36
L2 3. CHROOT Apache	36
L2 4. ErrorLog - Syslog	38
L2 5. Tracking Security Related HTTP Status Codes	39
L2 6. Mod_Evasive – Apache Denial of Service Prevention Module	40
L2 7. Buffer Overflow Protections	42
L2 8. URL Inspection with Mod_Rewrite	43
L2 9. Mod_Security – Level II Settings	44
L2 10. Web Server Fingerprinting	45
L2 11. Monitoring the ErrorLog File with SWATCH	47
L2 12. Reverse Proxy for Protection	48
L2 13. Update the Apachectl Script for Email Notification	51
APPENDIX A – APACHE MODULES LISTING	52
APPENDIX B -- BUILD APACHE FROM SOURCE	63
APPENDIX C – REFERENCES	67
APPENDIX D – RED HAT LINUX REFERENCES	68

Level 1 (L1) Apache Benchmark Settings

The Prudent Level of Minimum Due Care

Level-I Benchmark settings/actions meet the following criteria.

1. System administrators with any level of security knowledge and experience can understand and perform the specified actions.
2. The action is unlikely to cause an interruption of service to the operating system or the applications that run on the system.
3. The actions can be automatically monitored, and the configuration verified, by Scoring Tools that are available from the Center or by CIS-certified Scoring Tools.

Many organizations running the CIS scoring tools report that compliance with a CIS "Level-1" benchmark produces substantial improvement in security for their systems connected to the Internet.

[Back to Top ^](#)

LEVEL I -- Apache Benchmark Settings

L1 1. Harden the Underlying Operating System

Description

Operating System Security is beyond the scope of this document, however, this step should not be ignored. The undeniable symbiotic relationship between a Web server and its underlying OS cannot be overstated. Both the Web server and the OS could potentially be used to exploit each other. For instance, a vulnerable version of the BIND daemon could potentially give an attacker command line access to the system. This unauthorized access could put the web site's contents into jeopardy. Conversely, a web server running a vulnerable version of the CGI script PHF could allow an intruder to illegally access the OS password file. This information might eventually lead to unauthorized system access. Addressing the security concerns of a Web server and ignoring the system OS is akin to locking the front door of a house while leaving the backdoor wide open. Therefore, it is imperative to harden the OS to truly prevent successful web attacks. A perfect example of failing to address this issue and how it could leave a web server vulnerable to attack is explained in [Hackers Win Security Challenge](http://www.wired.com/news/technology/0,1282,43234,00.html). <http://www.wired.com/news/technology/0,1282,43234,00.html>.

Action: Apply CIS benchmarks and perform baseline security of OS

Users of this document should apply any and all available operating system benchmarks prior to installing or securing Apache. This hardening process usually includes steps to disable un-needed services and to apply the latest security patches. As of this benchmark release, the following benchmarks are available for Unix-like operating systems.

- Red Hat Linux Benchmark v1.0.5 and scoring tools
- SUSE Linux Benchmark v1.0
- Slackware Linux Benchmark v1.1
- HP-UX Benchmark Version v1.3.1 and scoring tools
- Sun Solaris Benchmarks v1.3 and v2.1.1
- FreeBSD Benchmark v1.0.5
- IBM AIX Benchmark v1.01
- MacOS X Benchmark v2.0

The Center For Internet Security benchmarks are available at: <http://www.cisecurity.org/bench.html>
Additionally, users are encouraged to refer to the SANS Step by Step Guides from The SANS Institute where OS benchmarks are available: <http://www.sansstore.org/>

The server should be offline during the OS hardening process. If installing on an existing server with an unknown usage history, the OS should be reinstalled and secured using the CISEcurity baseline tools, where applicable.

L1 2. Install Apache Web Server

Question:

Are you planning to use the precompiled Apache httpd binary that comes default with many Unix Operating Systems?

Description

The CIS Apache Benchmark now recommends using the Apache binary provided by your OS vendor for most situations. The benefits of using the OS vendor supplied binaries include:

- Ease of initial installation
- Allows the updates and patch process to be automated, and integrated with the patch update cycle used for the operating system updates.
- The security modules mod_ssl and mod_security no longer require source code compilation.

There are times when compilation from source code will be necessary or advantageous, however for most situations the vendor binaries will provide better security by ensuring that updates are applied according to the existing update process.

Action: Install the Apache Software using vendor provided binaries if available

```
# yum install httpd
```

In the event vendor binaries are not available or suitable, recommended instructions for downloading, building from the source and installing are included in Appendix B.

[Back to Top ^](#)

L1 3. Create the Web Groups

Description

In order to segment duties and associate real users with web content, we want to create new web group accounts. The account names will vary depending on your environment. The goal is to create specific groups that serve certain functions: for example, the webadmin group would own and maintain the web servers configuration documents located in the /usr/local/apache/conf, /usr/local/apache/bin and /usr/local/apache/logs directories. The webdev group would own and maintain all of the actual web document root files within the /usr/local/apache/htdocs directory. The apache group is only used as the group association that the apache user has. We do not want the apache user to be a member of the users group. The apache group is often automatically created when the Apache software is install, so creating the apache group may not be necessary. If the apache group does need to be created, it should be created as a gid < 500, so that it is a system group rather than a user group, as recommended in the CIS Red Hat Benchmark for system group ids. The -r option for the RedHat groupadd command will select the first available system gid < 500. We will discuss how to modify the ownership and permissions on directories and files in a later section.

Action: Create Dedicated Groups

Execute the following commands to create the appropriate web groups:

```
# groupadd webadmin
# groupadd webdev
# groupadd -r apache
```

In the example in this section, we use the names webadmin, webdev, apache. These names are provided as examples only.

L1 4. Create the Apache Web User Account

Description

One of the best ways to reduce your exposure to attack when running a web server is to create a unique, unprivileged userid and group for the server application. The "nobody" userid & 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 userid used for the apache user should be a unique value between 1 and 499 as these lower values are reserved for the special system accounts not used by regular users, such as discussed in User Accounts section of the CIS Red Hat benchmark.

Create an account with a name such as: **apache**, which runs the web server software. In the entry below, we designate the web document root as the apache user's home directory. Using the "-m" flag will create this directory if it does not already exist. Since this account will never be used to log into for shell access, we do not need to create the normal user account login files. Designating the web server document root as the home directory for the apache user also helps with security since we will be creating a restrictive disk quota in a Level II section. This will prevent the apache OS account from ever creating any new files in the document root, thus preventing many web defacement attacks.

Action: Create Apache Account if does not already exist.

Execute the following command :

```
# GID=$(awk -F':' ' /^apache/ {print $3}' < /etc/group)
# useradd -d /var/www/ -g apache -u $GID -c "Web Server" -m \
-s /dev/null apache
```

Again, use an account naming convention unique to your site as described in the previous section. The apache account may already have been created when you installed the apache software.

[Back to Top ^](#)

L1 5. Lock Down the Apache Web User Account

Description

In the example below, **apache**, is the name of the web user account you created. To make sure the user account you created cannot be logged into, you want to lock this new account by using the passwd command below. This will lock the account within the /etc/shadow file by replacing the encrypted password hash with the locked letters. The encrypted password field should contain the entry "!!" or "***LK***" indicating that the account is locked and cannot be used to log in. Additionally, by using the usermod command below, you are changing the default system shell for the new user to a non-valid shell. After updating the user account to specify this non-valid shell, verify the apache account entry within the /etc/passwd and /etc/shadow file. Blocking all system accounts is covered in the CIS Red hat benchmark under the section on User Accounts.

Action: Account Lockdown

Execute the following commands to lock down the new apache account:

```
# passwd -l apache
# usermod -s /dev/null apache
# grep apache /etc/passwd /etc/passwd
/etc/shadow:apache:!!:13362:0:99999:7:::
/etc/passwd:apache:x:48:48:Apache:/var/www:/dev/null
```

[Back to Top ^](#)**L1 6. Subscribe to the Appropriate Security Advisories****Description**

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

Action: Subscribe to the appropriate Security Advisory List

List Description	URL
Apache Web Server Announce List	http://httpd.apache.org/lists.html
CERT security advisories	https://forms.us-cert.gov/maillists/
Sun Solaris Announcements	https://subscriptions.sun.com
Fedora Core Announcements	https://www.redhat.com/mailman/listinfo/fedora-announce-list

L1 7. Apply Current Patches**Description**

Obviously knowing about newly discovered vulnerabilities is only part of the solution; there needs to be a process in place were patches are tested and installed on the systems. These patches fix diverse problems, including security issues, and are created from both in-house testing and user-community feedback.

Possible update and patch steps listed below for selected platforms will ensure that your Apache software is up to date with the most current fixes available. If kernel updates are included, it will be necessary reboot the server, and when the apache software or other services are updated you should verify that services have been properly restarted after being updated.

Action: Follow the patch/update process for your organization to update your OS and Apache software at least every 1 per month.

Fedora Core Manual Update:

```
# yum update
```

Fedora Core Automated Nightly Update:

```
# chkconfig yum on
# chkconfig --list yum
yum          0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

Patch for Source Code:

If you built from Apache source code:

- Check the following web site for the latest patches for the version of Apache you downloaded:
<http://www.apache.org/dist/httpd/patches/>
- Look for a directory called "apply_to_2.X.XX/" which matches your source version.
- If this directory exists, read the description to verify if the patch is applicable to your OS and then download the file.
- Also verify the integrity of the patch by importing the Apache PGP keys and checking the PGP signature. The apache PGP key's are available on the same web site <http://www.apache.org/dist/httpd/KEYS> however it's best to retrieve them from a different source, such as the MIT PGP key servers, or any of the PGP key mirrors.

Place this file in the new Apache source build directory, which is created once you have unzipped and untarred the Apache archive in the section below. Once this patch file is in the this directory, issue the following commands (**this is an example and shows how to apply the patch called "no_zombies.patch". You may not need to apply any patches**):

```
# pwd
/apache/build/directory
# gpg --verify httpd-2.X.XX.tar.gz.asc
# tar xzf httpd-2.X.XX.tar.gz
# cd httpd-2.X.XX
# patch -s < /apache/build/directory/no_zombies.patch
```

[Back to Top ^](#)

Level 1 Benchmark Settings for the Apache Configuration File (httpd.conf)

Unfortunately, most web server's default configurations are not adequate for deployment on today's Internet. Usually these default settings are configured with a too "open" mindset. In actuality, the exact opposite of the aforementioned statement should be the standard. This is known as the "Principal of Least Privilege." Access controls should start off with total restriction and then access rights should be applied appropriately. If a production web server is bound for the Internet, various web server system settings need to be changed and/or implemented.

In the following sections, we will be discussing many different Apache configuration settings. For most of these settings, we will be altering for a desired security effect. There are some settings, however, which are the default setting and it is recommended that the reader merely confirm this particular setting. The examples highlighted in all of the pseudo-terminal screens are the **RECOMMENDED** setting.

[Back to Top ^](#)

L1 8. User Oriented General Directives

Description

We will be verifying/updating the following Apache directives:

- **User**
The **User** directive establishes who is the owner of the children processes spawned by the main Apache service. **Do not run your Apache web server as root.** Since the well-known port of httpd is 80 and since only root can bind to ports below 1024, the main Apache listening process must be owned by root. However, it would be very dangerous if any CGI child process also ran as root because if they were compromised the attacker would have full root access to the system. The User directive restricts the privileges that can be obtained if the child process is compromised. In following with the set up preparation that has already been done, the recommended setting for this directive is **apache**.
- **Group**

The **Group** directive provides similar functionality to the User directive. That is it is another means of restricting the privilege level that the child processes can run in. For instance, if the Group directive is set to wheel and a CGI script was compromised, the attacker would gain root-level privilege. We created a new group called "apache" which should ONLY have the apache user as a member. This will prevent any undesired Group permissions to be associated with the apache account. The recommended setting for this directive is **apache**.

- **ServerAdmin**

The **ServerAdmin** directive is the email address for the site's web administrator. The security vulnerabilities associated with publicly displaying an actual user's email address may seem trivial, however, do not underestimate the security implications. Specifying a user's local email address - such as - "Jdoe@webserver.com" reveals that user's local OS account name. Using an e-mail alias such as "SysAdmin@companyx.com" instead of "RealName@companyx.com," reduces the likelihood of a successful e-mail spoofing attack. This technique can also prevent attackers from determining legitimate system usernames from the e-mail addresses. For instance, if your ServerAdmin setting was this:

```
ServerAdmin bsmith@www.companyx.com
```

It would be a good bet that "bsmith" is an actual user's account name on this server. With this information, an attacker could then try to gain OS level access to the system by Brute Forcing the bsmith account. An additional benefit that comes with using the Webmaster email address instead of an actual user's email account is redundancy. What happens if you put your personal email address in the ServerAdmin directive and then when you away from work for an extended period of time, who else gets these emails? If you use an email alias group for WebAdmin@yourhostname.com, then these emails will actually be forwarded to a group of people instead of an individual.

Action: Edit/Verify the following bolded directives in the httpd.conf file
Italicized entries should be set appropriately for your environment.

```
User apache  
Group apache  
ServerAdmin WebAdmin@hostname.com
```

[Back to Top ^](#)

L1 9. Disable Unnecessary Apache Modules

Description

It is important to disable the Apache modules that are not needed, in order to reduce the risk to the web server, as well as increase the performance. This is similar to the OS security issue of running unnecessary network services; such as Telnet and FTP. By enabling these unused modules, you are potentially providing additional avenues of attack against your web server. You should only enable the modules that you absolutely need for the functionality of your web site. Appendix A of the benchmark describes most modules along with their security risk and the recommended setting. If you are not sure a modules you need, you may also read about them at <http://httpd.apache.org/docs/mod/>, then disable the module and test the functionality without the module. Shown below are a few sample LoadModules directives which have been commented out on Fedora Code 5.

Action: Review Appendix A with the recommended modules settings and comment out all LoadModule directives for modules that are not used in the httpd.conf file. Some examples follow:

```
LoadModule access_module modules/mod_access.so
```

```
LoadModule auth_module modules/mod_auth.so
LoadModule auth_anon_module modules/mod_auth_anon.so
## LoadModule auth_dbm_module modules/mod_auth_dbm.so
## LoadModule auth_digest_module modules/mod_auth_digest.so
## LoadModule ldap_module modules/mod_ldap.so
## LoadModule auth_ldap_module modules/mod_auth_ldap.so
## LoadModule cern_meta_module modules/mod_cern_meta.so
## LoadModule dav_module modules/mod_dav.so
```

Action: Review the module *.conf files included from /etc/httpd/conf.d/ and comment out or remove the rpm associated with each unnecessary module.

The rpm -qf in the for-loop below shows to which rpm a module *.conf file belongs. If the module is not needed, and does not belong to the base Apache install or some other rpm that you do need, then remove the rpm with 'rpm -e'. If the .conf file is part of something you need, but you don't need the Apache module, then you should either comment out the contents of the file, and/or truncate the file. Removing the file is not recommended, as future updates will replace the removed file if it does not exist.

```
# for f in *.conf; do
    echo -n "$f: "; rpm -qf $f;
done
manual.conf: httpd-manual-2.2.2-1.2
mod_security.conf: mod_security-1.9.4-1.fc5
perl.conf: mod_perl-2.0.2-5.1
php.conf: php-5.1.4-1
proxy_ajp.conf: httpd-2.2.2-1.2
python.conf: mod_python-3.2.8-3
squid.conf: squid-2.5.STABLE14-2.FC5
ssl.conf: mod_ssl-2.2.2-1.2
webalizer.conf: webalizer-2.01_10-29.2.1
welcome.conf: httpd-2.2.2-1.2

# rpm -e squid
# rpm -e mod_python
# rpm -e mod_perl
# rpm -e httpd-manual
# for fi in welcome.conf proxy_ajp.conf; do
    echo "## Module not used." > $fi
done
```

L1 10. Denial of Service (DoS) Protective General Directives

Description

It is almost impossible to fully protect against denial of service attacks since they are attacks against the fundamental underpinnings of the Internet. Denial of service attacks often try to exhaust the system's resources. For instance, since the system can only wait for a limited number of connection requests to complete, an attacker will flood a system with a bunch of connection requests that will never be completed. The server will keep the connection request open until either the connection has been made or a specified length of time has gone by. Once the attacker has filled the pending connection queue, a small trickle of new requests will keep the server unavailable to legitimate requests. Apache has several directives available that provide some protective capabilities against denial of services attacks.

We will be verifying/updating the following Apache directives:

- **Timeout**

One way of attacking systems on the Internet is to try to prevent the target system from operating correctly by overloading it. This is called a 'denial of service' attack. One method of doing this is to open multiple connections to a server and never close them. The more connections the server has open at once, the more resources are tied up holding details of those connection, which can lead to increased load and eventually to the server running out of resources. The Timeout directive tells the server how long to wait to receive a GET request, the amount of time between receipt of TCP packets on a POST or PUT request, or the amount of time between ACKs on transmissions of TCP packets in responses. In order to prevent a denial of service attack from shutting down our web server, we need to change the default setting of 300 (which is 5 minutes) to **60** (which is 1 minute). You may even adjust this setting to be lower than 60.

- **KeepAlive**

In order to make our web server more efficient, and therefore more resistant to DoS attacks, we want to allow TCP KeepAlives. This setting will allow for multiple HTTP requests to be serviced across one connection. Without this setting, a new Apache server would have to be spawned for every subsequent HTTP request for gifs, etc. Recommended value is **On**.

- **KeepAliveTimeout**

This directive will limit the time the Apache web server will wait in seconds for a KeepAlive request to be completed. Setting KeepAliveTimeout to a high value may cause performance problems in heavily loaded servers. The higher the timeout, the more server processes will be kept occupied waiting on connections with idle clients. If this number is set too high, then a DoS attack could be more successful. Recommend value is **15** which is also the default value.

- **Other Performance and DoS Related Modules and Directives**

Taking together the directives StartServers, MaxSpareServers and MaxClients, along with the MPM (MultiProcessing Module) work to establish a reasonable yet dynamic number of child processes and/or threads. Starting with Apache 1.3 the process creation process was dramatically improved so that tweaking the StartServers, MaxSpareServers and MaxClients settings is unnecessary for most situations. Therefore the default settings are fine for most usages. For very high traffic servers, and optimal performance consult the Apache performance recommendations at <http://httpd.apache.org/docs/2.0/misc/perf-tuning.html> or for Apache 1.3 refer to <http://httpd.apache.org/docs/1.3/misc/perf-tuning.html>

Action: Edit/Verify the following bolded directives in the httpd.conf file.

```
Timeout 60
KeepAlive On
KeepAliveTimeout 15
```

[Back to Top ^](#)

L1 11. Web Server Software Obfuscation General Directives

Description

One way to protect the Apache server is to make it limit the information provided to a potential attacker about the web server version. There are several ways that the server can leak identifying information. We will be verifying/updating the following Apache directives:

- **ServerTokens**

This configuration setting aids in hiding the web server software version and module versions. We do not want to give away any more information about our web server than is absolutely necessary. We will only give out the minimum server token information, therefore the recommended setting is **Prod** or **ProductOnly** which does not provide any information on versions or modules loaded, but only provides the "Apache" in the server HTTP response header.

- **ServerTokens Prod[uctOnly]**
Server sends (e.g.): Server: Apache
 - **ServerTokens Major**
Server sends (e.g.): Server: Apache /2
 - **ServerTokens Minor**
Server sends (e.g.): Server: Apache /2.0
 - **ServerTokens Min[imal]**
Server sends (e.g.): Server: Apache/2.0.41
 - **ServerTokens OS**
Server sends (e.g.): Server: Apache/2.0.41 (Unix)
 - **ServerTokens Full (or not specified)**
Server sends (e.g.): Server: Apache/2.0.41 (Unix) PHP/4.0 MyMod/1.2
- **ServerSignature**
In order to protect which web server software we are using, we should not disclose this information in any of our system generated Error pages. The ServerSignature directive instructs Apache to append certain footer information to the bottom of error pages. Here is an example response with a signature:

`The requested URL /nosuch was not found on this server.`

`Apache Server at 10.1.1.12 Port 80`

If we do not disable the ServerSignature setting, we may be reducing the security benefit gained by changing the ServerTokens. While it is true that the ServerSignature would show only the "Apache" server token setting we specified in the ServerTokens directive, this signature feature is still in the **Apache style** and may be an additional identifier. By removing the ServerSignature, we can take another step towards protecting our web server software information.

Technically, the ServerSignature issue becomes moot if you follow the next section and configure Apache NOT to utilize the default error pages for all HTTP Status Codes.

- **ErrorDocument (Custom Error Pages)**
Each type of web server has its own distinct style of error pages. The server sends these pages when an error, such as "404 - Not found," has occurred. By issuing a request for a file that is not present on a web server, an attacker may determine the web server software by simply identifying the 404 – Not Found error pages displayed. To avoid this software disclosure, the default error pages presented by the web server must be changed. There are two possible choices:
 - Edit the default error pages to a style that is consistent with the website's template. This may include changing color schemes and altering the text message displayed.
 - For deception purposes, edit the error pages to the exact style of a different web server. For example, if a web server is currently running Apache, change the error pages to resemble a different web server version.

In the example below, we are configuring Apache to send the custom html files, within the **DocumentRoot** directory for a number of common errors on the web server. You should create a custom

html file for each http status code error 4xx – 5xx. Further information - <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

Action: Edit/Verify the following bolded directives in the httpd.conf file
Italicized entries should be set appropriately for your environment.

```
ServerTokens Prod
ServerSignature Off
ErrorDocument 400 /custom400.html
ErrorDocument 401 /custom401.html
ErrorDocument 403 /custom403.html
ErrorDocument 404 /custom404.html
ErrorDocument 405 /custom405.html
ErrorDocument 500 /custom500.html
-- CUT --
```

[Back to Top ^](#)

L1 12. Mod Security

Description

Mod_Security is an Intrusion Prevention module for Apache. It is extremely flexible and provides a robust security layer for HTTP inspection. Here is a quick list of Mod_Security's features:

- **HTTP Intrusion Prevention Gateway:** An Apache server with Mod_Security, functioning as a Reverse Proxy Server, can protect internal web servers from attacks.
- **Understanding of the HTTP Protocol:** Since the engine understands HTTP, it performs very specific and fine granulated filtering. This is the functionality, which Firewalls are lacking.
- **Request Filtering:** Incoming requests are analyzed as they come in, and before the web server or other modules handle them.
- **Inspect Any/All Headers:** With Mod_Security, we can set filters on any client request header, not just the URL Request line.
- **Flexible Rules:** The Mod_Security rule directives can leverage the use of Regular Expressions.
- **Anti-Evasion Techniques Normalization:** Paths and parameters are normalized before analysis takes place in order to fight evasion techniques. Anti-evasion techniques include:
 - URL Encoding Validation
 - URL Decoding
 - Reduce ./ to /
 - Reduce // to /
- **POST Payload Analysis:** The engine will intercept the contents transmitted using the POST method, which allows for basic inspection of file uploads.
- **Audit Logging:** Full details of every request (including POST) can be logged for later analysis. The entire environmental session tokens are dumped in the audit_log file. This data is crucial for investigating incidents.
- **Built-In Chroot Functionality:** All you need to do is specify the Chroot directory in the Mod_Security Directive brackets like this – “SecChrootDir /path/to/chroot”. More information below.

- **Pause Feature:** This will cause Mod_Security to wait a specified period of time (in milliseconds) before acting on a request trigger. The main benefit here is that time delays can significantly slowdown, and in some cases break, vulnerability scanners. Syntax is –
`SecFilterDefaultAction "deny,log,pause:10000,status:403"`
- **Buffer Overflow Protection:** Mod_Security can set limits on acceptable byte ranges for user input (excluding the POST Payload of a request). This can be useful for avoiding many stack overflow attacks (since they often contain random “binary” data). Default byte range is 0-255, meaning all bytes are allowed. Typically, byte ranges between 32–126 are all you need for http communication. This link provides an ASCII Chart showing the byte ranges - <http://i-technica.com/whitestuff/asciichart.html>.
- **Server Identity Masking:** Instead of editing and recompiling source code, Mod_Security has a directive, which will change your HTTP “Server:” banner token just before it is sent to the client. Example – `SecServerSignature "Microsoft-IIS/5.0"`
- **HTTPS Filtering:** Since the engine is embedded in the web server, it gets access to request data after decryption takes place. Normal NIDS cannot inspect SSL/Encrypted traffic, but Mod_Security can!

Action: Execute the following commands to implement Mod_Security

Prior to implementing Mod_Security, it is HIGHLY recommended that you read the full User’s Manual at this location: <http://www.modsecurity.org/documentation/>. This document will provided full descriptions of Mod_Security’s capabilities and will help to clarify the quickstep information below. First you must install mod_security.

Installing mod_security via yum

```
# yum install mod_security
```

Review and edit the `/etc/httpd/conf.d/mod_security.conf` to suit your needs. The following are enabled by default for Fedora Core 5 and recommended and are likely to be safe for most web servers, also review suggestions from the Mod_security website <http://www.modsecurity.org/documentation/quick-examples.html>

```
LoadModule security_module modules/mod_security.so
<IfModule mod_security.c>
```

```

# Turn the filtering engine On or Off
SecFilterEngine On

# The audit engine works independently and
# can be turned On of Off on the per-server or
# on the per-directory basis
SecAuditEngine RelevantOnly

# Make sure that URL encoding is valid
SecFilterCheckURLEncoding On

# Unicode encoding check
SecFilterCheckUnicodeEncoding On

# Only allow bytes from this range
SecFilterForceByteRange 1 255
```

```

# Cookie format checks.
SecFilterCheckCookieFormat On

# The name of the audit log file
SecAuditLog logs/audit_log

# Should mod_security inspect POST payloads
SecFilterScanPOST On

# Default action set
SecFilterDefaultAction "deny,log,status:406"

# Require HTTP_USER_AGENT and HTTP_HOST headers
SecFilterSelective "HTTP_USER_AGENT|HTTP_HOST" "^$"

# Only accept request encodings we know how to handle
# we exclude GET requests from this because some (automated)
# clients supply "text/html" as Content-Type
SecFilterSelective REQUEST_METHOD "!^GET$" chain
SecFilterSelective HTTP_Content-Type "!(^$|^application/x-www-form-
urlencoded$|^multipart/form-data)"

# Require Content-Length to be provided with
# every POST request
SecFilterSelective REQUEST_METHOD "^POST$" chain
SecFilterSelective HTTP_Content-Length "^$"

# Don't accept transfer encodings we know we don't handle
# (and you don't need it anyway)
SecFilterSelective HTTP_Transfer-Encoding "!^$"

</IfModule>

```

Installing mod_security as DSO for Source Code builds

Untar the modsecurity tar ball, and cd to the apache2 or apache1 directory according to the apache version used. Run the Apache apxs command to build and install the mod_security dynamic module.

```

# pwd
/modsecurity/build/directory
# /usr/local/apache2/bin/apxs -cia mod_security.c

```

You then need to check that the LoadModule directive for mod_security was added to the appropriate httpd.conf file and then add the same mod_security configuration filters as given above; also the Mod_Security website also provides example httpd.conf entries - <http://www.modsecurity.org/documentation/quick-examples.html>

[Back to Top ^](#)

L1 13. Access Control Directives

Description

The Allow and Deny directives are straightforward. The **Allow** directive grants access while the **Deny**

denies access. The Order directive is trickier to understand. The order does matter! Don't confuse this with the order that items appear in the configuration file. Consider the following example:

```
Order Allow,Deny
Allow from apache.org
Deny from foo.apache.org
```

The Order directive states that the Allow directives should be evaluated first. The Allow directive states that everyone from apache.org can have access. Then the Deny directive states that everyone from foo.apache.org should be denied access. Taken together these directives instruct the server to "allow access to everyone from apache.org except for those from foo.apache.org".

If the Order directive in the above example were reversed to "Deny,Allow", then the server would grant access to everyone from apache.org, because the Allow directive is evaluated after the Deny directive. In other words the server was told to "deny access to anyone from foo.apache.org then grant access to everyone from apache.org". Since foo.apache.org is part of apache.org, access will be granted to its users. The order and allow/deny parameters should be applied to any portions of your web site, which you would like to protect. You should use IP addresses when possible, to prevent any DNS spoofing attacks. You can use the allow/deny options in conjunction with any password-protected methods as well.

Action: Edit/Verify the following bolded directives in the httpd.conf file
Apply Access Control for OS Root and DocumentRoot Directories

OS Root Directory Access Control

In order to prevent any form of directory traversal trying to get outside of the document root, we will specify the directive listed above. 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 serve it to clients. For instance, consider the following example:

```
# cd /; ln -s / ~user/public_html
```

When accessing - http://localhost/~user/ within a web browser would allow clients to walk through the entire filesystem. The httpd.conf configuration directive listed below will deny any access attempts for the OS root directory "/".

```
<Directory />
  Options None
  AllowOverride None
  deny from all
</Directory>
```

DocumentRoot Directory Access Control

The example listed below essentially allows anyone to access the web site:

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

You can use more restrictive options to the allow/deny options such as specific hostnames, IP addresses, domain names and IP ranges.

```
<Directory "/var/www/html/">
    Order allow,deny
    deny from all
    allow from 202.54.
</Directory>
```

This would restrict access to only hosts who reside in the appropriate IP range (202.54.X.X).

[Back to Top ^](#)

L1 14. Authentication Mechanisms

Description

There are two ways of restricting access to documents: either by the hostname of the browser being used, or by asking for a username and password. The former can be used to, for example, restrict documents to use within a company. However if the people who are allowed to access the documents are widely dispersed, or the server administrator needs to be able to control access on an individual basis, it is possible to require a username and password before being allowed access to a document. This is called **user authentication**.

Setting up user authentication takes two steps: first, you create a file containing the usernames and passwords. Second, you tell the server what resources are to be protected and which users are allowed (after entering a valid password) to access them.

Two forms of Authentication:

- **Basic** (Must have Mod_Auth implemented)
Client's web browser sends MIME base64 encoded user credentials (username + password) to the web server when the browser receives a "401 –Authorization Required" status code. Basic Authentication is easy to implement, but does not provide any real security against sniffing attacks.
<http://httpd.apache.org/docs/howto/auth.html#basiccaveat>
- **Digest** (Must have Mod_Digest implemented)
Makes sending of passwords across the Internet more secure. It effectively encrypts the password before it is sent such that the server can decrypt it. It works exactly the same as Basic authentication as far as the end-user and server administrator is concerned. The use of Digest authentication will depend on whether browser authors write it into their products. While Digest Authentication does help with protecting the user's credentials, it does not protect the data itself. You should implement SSL if you need to protect sensitive data in transit.
<http://httpd.apache.org/docs/howto/auth.html#digestcaveat>

Action: Restrict Access to a Directory or File

Make sure that the password file containing user credentials is NOT stored within the DocumentRoot directory! If this happens, clients may be able to access this file and view the data. Refer to section L1 24 for further information on proper permissions for the password files (READ only for the apache user).

If you need to restrict access to a directory or file, use the following commands:

Basic Authentication:

```
# htpasswd -c /path/to/passwordfile test
New password: password
Re-type new password: password
```

Adding password for user test

Within the httpd.conf file, add an entry to protect the desired content:

```
<Directory /var/www/html/protected>
AuthType Basic
AuthName "Private Access"
AuthUserFile /path/to/passwordfile
Require user test
</Directory>
```

Digest Authentication:

```
# htdigest -c /path/to/digestfile "Private Access" test
New password: password
Re-type new password: password
Adding password for user test
```

Within the httpd.conf file, add an entry to protect the desired content:

```
<Directory /var/www/html/protected>
AuthType Digest
AuthName "Private Access"
AuthDigestFile /path/to/digestfile
Require user test
</Directory>
```

[Back to Top ^](#)

L1 15. Directory Functionality/Features Directives

Description

The Options Directive controls what extended web server functions are applied to directories and/or files.

- The **All** setting states that all features are available except for Multiviews.
- The **ExecCGI** setting permits the execution of CGI scripts within the directory. This feature should only be applied to the designated cg-bin directory.
- The **FollowSymLinks** setting allows the server to follow symbolic links found in the directory. The FollowSymLinks directive will instruct the Apache web server on how to handle the processing of files, which are symbolic links. If you must use symbolic links for the functionality of your web site, consider the security risks that follow. It is possible for an attacker to gain access to areas outside the specified document root if the web server is configured to follow symbolic links. We will configure this parameter setting to NOT follow symbolic links. This option is preferred over the SymLinksIfOwnerMatch due to the performance hit when Apache verifies a symlink and its ownership.
- The **SymLinksIfOwnerMatch** setting instructs the server to only follow symbolic links if the file has the same owner as the symbolic link. This directive should be used only if the use of symbolic links is absolutely necessary for proper functioning of you web server. This will apply an additional security check to verify that the file the symbolic link points to is owned by the same UID that the web server is running under. If proper ownerships and permissions are set for the

DocumentRoot, ServerRoot and the OS directories (addressed in a later section), then the chances of exploiting a symbolic link is significantly reduced.

- The **Includes** setting permits the execution of server side includes. This directive in the terminal screen below will prevent the web server from processing Server Side Includes (SSI). These are OS commands located within the html code of a web page. SSIs are executed by the web server before the page is sent to the client. The format of a typical SSI is shown below.

```
<!--#<tag><variable set> '--->
```

A common technique of web attackers is to mirror a web site and search through the html code looking for insecurities. They may look for comment tags, email addresses, etc... If the attacker finds that the web site is using SSI within html pages, they may be able to use this feature for malicious purposes. The example below shows an attacker who is using SSI in the client's request headers in the hopes that the web server will execute these commands:

```
# telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET / HTTP/1.0
Referer: <!--#virtual include="somefile.log"--->
User-Agent: <!--#exec cmd="/bin/id"--->

HTTP/1.1 200 OK
Date: Mon, 17 Dec 2001 20:39:02 GMT
Server:
Connection: close
Content-Type: text/html
```

If you must use SSI, then it is recommended that you use the IncludesNoExec option, which will allow the server to parse SSI enabled web pages but it will not execute any system commands.

- The **IncludesNOEXEC** refines the Includes setting by disabling the exec command.
- The **Indexes** setting tells the server to automatically create a page that lists all the files within the directory if no default page exists (in other words - no index.html). Directory listings should not be allowed, since they reveal too much information to an attacker (such as naming conventions and directory structures). This directive below will prevent the web server from producing any dynamic directory listings when a default index page is not present. This httpd.conf file directive is actually redundant, since we have already disabled the mod_autoindex module when we compiled our httpd binary file.
- The **AllowOverride** setting tells the server how to handle access control from .htaccess files. When the server finds a .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. While the functionality of htaccess files is sometimes relevant, from a security perspective, this decentralizes the access controls out from the httpd.conf file. This could make it much more difficult to manage the security of your web site if rogue htaccess files are created.
- The **Multiviews** setting allows for multiple files to refer to the same request. It can be used to have the same page in different languages with each language having a different final file suffix. This setting could also be leveraged as part of an IDS Evasion attack. An example scenario is

when a NIDS may have an attack signature such as the one below (taken from Snort's web-cgi.rules file). Notice the bolded section -

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-CGI /wwwboard/passwd.txt access";
flow:to_server,established; uricontent:"/wwwboard/passwd.txt";
nocase; reference:arachnids,463; reference:cve,CVE-1999-0953;
reference:nessus,10321; reference:bugtraq,64
9; classtype:attempted-recon; sid:807; rev:7;)
```

This Snort rule would trigger if the URL request line contains the text string `"/wwwboard/passwd.txt"`. An attacker could try to evade the IDS system by sending a request which is similar to the actual filename (such as `"GET /wwwboard/passwd HTTP/1.0"`). Notice this request did not include the `.txt` suffix and therefore would not trigger the Snort alert. Multiviews would handle the negotiation and serve the desired file.

For increased security, only those features that are absolutely necessary should be enabled. All other features should be disabled. As a side note, since we disabled numerous Apache modules when we compiled the new httpd binary, many of these Options Directives would not work anyways. This adheres to security-in-layers and prevents the accidental enabling of an unauthorized service or feature.

Action: Edit/Verify the following bolded directives for the DocumentRoot directory in the httpd.conf file

The Options directives listed above will implement the security related features.

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

  Options -FollowSymLinks -Includes -Indexes -MultiViews
  AllowOverride None
</Directory>
```

[Back to Top ^](#)

L1 16. Limiting HTTP Request Methods

Description

We want to restrict the functionality of our web server to only accept and process certain HTTP Methods. For normal web server operation, you will typically need to allow the both the GET and POST request methods. This will allow for downloading of web pages and uploading any type of basic form submission information. The HEAD requests are included with GET requests when using the LimitExcept directive, so excluding HEAD requests will effectively stop anyone from downloading your web pages. This entry will cause all non-allowed HTTP Methods to trigger a 403-Forbidden status code.

The LimitExcept Directive works fine for all HTTP Methods EXCEPT for TRACE. The TRACE Method is unique and it has some specific security implications, therefore we will discuss TRACE in depth in the Mod_Rewrite section.

Action: Execute the following commands

Edit the httpd.conf file and add in the bolded lines within the DocumentRoot Directive:

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

```
allow from all
Options -Indexes -Includes -FollowSymLinks -MultiViews
AllowOverride None
</Directory>
```

[Back to Top ^](#)

L1 17. Logging General Directives

Description

The server logs are invaluable for a variety of reasons. They can be used to determine what resources are being used most. 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 that may be an indication that an attack is pending or has occurred. If there are multiple web sites, or with large websites with multiple people responsible for portions of the web site, each responsible individual or organization needs access to their own web logs, and needs the skills/training/tools for monitor the logs. We will be covering the following logging directives:

- **LogLevel**
This directive controls the verbosity of information that is logged in the error log file. This style of level warning is similar to that of the syslog facility (emerg, alert, crit, error, warn, notice, info and debug). The parameter in the recommendation below specifies that all information which is at a level of "notice" or higher will be logged.
- **ErrorLog**
This directive sets the name of the file to which the server will log any errors it encounters. Make sure that there is adequate disk space on the partition that will hold the all log files, and that log rotation is configured. 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.
- **LogFormat**
The **LogFormat** directive allows for the definition of log entries that have exactly the information desired. The basic structure of the directive is **LogFormat *format_specification format_name***. The format specification is comprised of a series of variable replacements. A named LogFormat directive does nothing, but is used in other directives such as the CustomLog entry. We will come back to the LogFormat settings in the Level II section.
- **CustomLog**
This directive specifies both which file to log access attempts to and what type of format the file should use. The directive above says that the "access_log" file will be combined and will contain both "referrer" and "user_agent" information. This information will be use in a later section when we receive alerts for malicious http requests. The **CustomLog** directive is used to log requests to the server. Its structure is **CustomLog logfile_name format_specification**. The entry below uses the format specification name "combined". The CustomLog directive is also valid in a virtual host context which means that different user requests can be logged by virtual host.

The CustomLog directive can also pipe its output to a command, but this can represent a severe security risk. This is because the command will be executed as the user that started the Apache server. Remember - in order to listen on port 80, Apache must be started with root. That means that when the log output is piped to a command the command will be running with root privileges. If that command can be compromised, then the attacker has gained root access to the server.

Make sure that there is adequate disk space on the partition that will hold the all log files. 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.

Action: Edit/Verify the following bolded directives in the httpd.conf file

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

Action: Edit/Verify the log rotation configuration, such as /etc/logrotate.d/httpd is in place to rotate the logs.

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

[Back to Top ^](#)

L1 18. Remove Default/Unneeded Apache Files

Description

Most web server applications come with sample applications or features that can be remotely exploited and which 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. The primary function for these sample routines is to demonstrate the capabilities of the web server. Sometimes they only exist to prove that the system is capable of executing a CGI script.

We will be removing the following files:

- Default HTML Files**

By default, Apache will install a number of files within the document root directory. These files are meant to be of assistance to the WebAdmin after successfully installing Apache. Included in these files is the famous "Seeing this instead of the web site you expected?" page. This is the page that will be displayed if you have not created any new html index pages. Also included in these files is the entire Apache documentation library. While all of these files are helpful, they do not conform to our security goal of hiding which type of web server software we are running. It would be foolish to go through all of our previous steps to protect our web server software version, only to loudly announce with these web pages that we are running Apache. By the way, all of the Apache documentation is available at the Apache web site - <http://httpd.apache.org/docs/>.
- Sample CGIs**

Attackers will often try to exploit CGI programs on the web server. They will either use these programs for reconnaissance purposes or to try and exploit the web server/OS directly. CGI programs have a long history of security bugs and problems associated with improperly accepting user-input. Since these programs are often targets of attackers, we need to make sure that there are no stray CGI programs that could potentially be used for malicious purposes. By default, Apache 1.3.27 comes with two stock CGI scripts. These are called - printenv and test-cgi. Both of these programs should be either renamed or removed entirely from the web server. This is due to the sensitive information, which an attacker could gain if they are able to successfully

access these files. In addition to removing the stock CGIs, care should be taken to review the functionality and code of any new CGIs which are created. The topic of safe CGI scripting is beyond the scope of this document, however, it should not be overlooked. The Web Developers should follow safe coding practices, such as, those outlined in the WWW Security FAQ - <http://www.w3.org/Security/Faq/wwwsf4.html>

- **Apache User Files**

Make sure to remove any normal user files associated with the apache user. For example, remove any shell history files (.bash_history, etc...). If you specified the Apache DocumentRoot as the home directory when you created the apache user account, then you will have already removed all extraneous files when you deleted the default html files. You do not want to leave any of these files lying around for clients to stumble upon. FYI – The vulnerability scanner Nikto (<http://www.cirt.net/code/nikto.shtml>) will search for these types of files!!!

- **Apache Source Code Files**

In order to keep our compiled installations of Apache secure, we will not keep the Apache source code on the production server. This will prevent any unauthorized users from re-compiling/compiling a new version of Apache.

Action: Execute the following commands:

Remove Any Default Apache HTML Files, if any:

```
# pwd
/var/www/
# ls html
index.html
# rm -rf html/*
```

Remove Default Apache CGI Scripts, if any:

```
# pwd
/usr/local/apache/cgi-bin
# mkdir ../old_cgis
# ls -l
total 58
-rwxr-xr-x  1 root  other      3009 Apr 29 14:01 printenv
-rwxr-xr-x  1 root  webadmin   757 Jan 22 15:48 test-cgi
# mv * ../old_cgis/
```

Remove Apache Source Code if built from source:

```
# pwd
/apache/build/directory
# rm -rf httpd-2.X.XX
# rm httpd-2.X.XX.tar.gz*
```

[Back to Top ^](#)

L1 19. Updating Ownership and Permissions for Enhanced Security

Description

Setting the appropriate ownership and permissions (utilizing the newly created webadmin, webdev and apache groups from a previous section) can help to prevent/mitigate exploitation severity. These changes should be made just before deployment into production to correct any insecure settings during your testing phase. It is also advisable to check/update these settings on a continued basis through a Cron job. We will be updating the ownership and permissions of the following directories and files:

- **Server Configuration Files**

We want to protect all of the Apache web server configuration files from unauthorized changes. These settings below will modify the files so that only the root user, and members of the WebAdmin group, will be able to read, write or execute any of the files within the conf subdirectory. **NOTICE** – Care should be taken if you are utilizing any ACLs. Keep in mind; if you create an Apache passwd file by using the htpasswd binary, the appropriate permissions must in place to allow the httpd child process (owned by apache) to read this file. If you place the passwd file in the server configuration directory, then you will have to reapply the Read permissions back to your /etc/httpd/conf/passwd file.

- **DocumentRoot Files**

We want to protect all of the content within our document root from unauthorized changes. These settings below will modify the files so that only the root user, and members of the WebDev group, will be able to read, write or execute any of the files within the conf subdirectory. This means that the user which our web server is running as - apache, will only be able to read files. The apache user will not have write access to any files within our web site.

- **CGI-Bin**

We want to protect all of the content within our cgi-bin directory from unauthorized changes. These settings below will modify the files so that only the root user, and members of the WebDev group, will be able to read, write or execute any of the files within the cgi-bin subdirectory. This means that the user which our web server is running as - apache, will only be able to execute files. The apache user will not have read or write access to any files within our cgi-bin directory.

- **Logs**

We want to protect all of the Apache web server's log files from unauthorized changes. These settings below will modify the files so that only the root user, and members of the WebAdmin group, will be able to read or write any of the files within the logs subdirectory. **We need to also allow for both Read and Execute permissions on this directory if you are using Mod_SSL.**

- **Bin**

We want to protect all of the Apache web server files and executable within the /bin directory from unauthorized changes or use. These settings below will modify the files so that only the root user, and members of the WebAdmin group, will be able to read, write or execute any of the files within the conf subdirectory.

Action: Execute the following commands:

Update the Server Configuration Directory's Ownership/Permissions:

Note: We are assuming that you have created a password database in this directory that needs to have read permissions.

```
# chown -R root:webadmin /etc/httpd/conf /etc/httpd/conf.d
# chmod -R 660 /etc/httpd/conf /etc/httpd/conf.d
# chmod 664 /etc/httpd/conf/password
```

Update the DocumentRoot Directory's Ownership/Permissions:

```
# chown -R root:webdev /var/www/html
# chmod -R 664 /var/www/html
```

Update the CGI-BIN Directory's Ownership/Permissions:

```
# chown -R root:webadmin /var/www/cgi-bin
# chmod -R 555 /var/www/cgi-bin
```

Update the Log Directory's Ownership/Permissions:

```
# chown -R root:webadmin /var/log/httpd/
# chmod -R 664 /var/log/httpd/
```

Update the Server Bin Directory's Ownership/Permissions:

```
# chown -R root:webadmin /usr/sbin/httpd /usr/sbin/apachectl
# chmod -R 550 /usr/sbin/httpd /usr/sbin/apachectl
```

[Back to Top ^](#)

L1 20. Implementing Secure Socket Layer (SSL) with Mod SSL

Description

Digital certificates encrypt data using Secure Sockets Layer (SSL) technology, the industry-standard method for protecting web communications developed by Netscape Communications Corporation. The SSL security protocol provides data encryption, server authentication, message integrity, and optional client authentication for a TCP/IP connection. Because SSL is built into all major browsers and web servers, simply installing a digital certificate turns on their SSL capabilities. Server certificates are designed to protect you and visitors to your site. Installing a digital certificate on your server lets you:

- Authenticate your site. 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.
- Keep private communications private. Digital certificates encrypt the data visitors that exchange with your site to keep it safe from interception or tampering using SSL (Secure Sockets Layer) technology.

IMPORTANT – Implementing SSL does NOT directly make your web server more secure! SSL is used to encrypt traffic and therefore does provide confidentiality of the users credentials when access your web server. 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 your server. Also, attackers will target SSL-Enabled web servers, since the encrypted channel will hide their activities from Network Based Intrusion Detection Systems. (See the Level II mod_security section for IDS functionality over SSL).

Action: Execute the following commands

For most systems it should be a simple matter of getting the mod_ssl and openssl rpms. For Fedora Core the simple command below will install mod_ssl if it is not already installed.

```
# rpm -q mod_ssl || yum install mod_ssl
```

If you have built from source code, starting with Apache 2 the mod_ssl is bundled in and you simply need to openssl and openssl-devel rpm's installed and add the '--enable-ssl' option to the configure script, before compiling, then configure mod_ssl as explained below. If you have built Apache 1.3 from source code, follow the instructions found at the mod_ssl website <http://www.modssl.org/example/>

The mod_ssl will automatically generate a dummy certificate which will not allow visitors to authenticate your server, but will provide encrypted communications. To get a trusted SSL certificate, follow the instructions below substituting example.com etc. with your organizations information. If the web server is only for lab use, a trusted certificate is not required, and you can skip this step. There are 3 requirements for the certificate to be trusted:

1. The common name (CN) on the certificate must match the URL host name which the user typed into the browser or received in an html link.
2. The certificate must be signed by a certificate authority trusted by the users browser (or by the web service / application).
3. The current date must not be beyond the certificate expiration date.

```
# cd /etc/pki/tls/certs
# make www.example.com.key
umask 77 ; \
/usr/bin/openssl genrsa -des3 1024 > example.com.key
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase:
Verifying - Enter pass phrase:
# chown root www.example.com.key
# chmod 600 www.example.com.key
# mv www.example.com.key /etc/pki/tls/private/

# make www.example.com.csr
umask 77 ; \
/usr/bin/openssl req -utf8 -new -key example.com.key -out example.com.csr
Enter pass phrase for example.com.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:New York
Locality Name (eg, city) [Newbury]:Lima
Organization Name (eg, company) [My Company Ltd]:Durkee Consulting
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:www.example.com
Email Address []:webadmin@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
```

A challenge password []:
An optional company name []:

```
# more example.com.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIByzCCATQCAQAwYoxCzAJBgNVBAYTA1VTMREwDwYDVQQIEWhOZXcgWW9yazEN
MAsGA1UEBxMETGltYTEaMBGGA1UEChMRRHVya2VlIENvbnN1bHRpbmcxGDAWBgNV
. . . CUT . . .
```

Next send the text content of example.com.csr to your certificate authority to be signed. It should be sent via a means that ensure that it arrives at the proper destination and is protected it from being modified in transit. Typically the certificate signing requests are submitted, not surprisingly, to a web site with an SSL connection. The resulting signed certificate we will name example.com.crt and placed in /etc/pki/tls/certs/. 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. Do not forget the pass phrase used to encrypt the private key. It will be required everytime the server is started in https mode. 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 if the risks are well managed. To decrypt the private the 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.

```
# openssl rsa -in example.com.key -out example.com.key.clear
# chmod 600 example.com.key.clear
# chown root example.com.key.clear
# more example.com.key
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: DES-EDE3-CBC, 2329D7D488CCA032

fdvTGcfVDJR9wteGVkqUgAe5lHYUmKcaW20IupDRcVxjWH7ieKs1ETgIVmrpJ9T3
5nJEp4d9Sulcv6NSNGptmEPpEiWuoLEz15wTGKzGxdF+12Nw/2Wl6AXtGANlTrN4
. . . CUT . . .
```

```
# more example.com.key.clear
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQDw7TZBR83WEoG40Podbe4ruvaCAMuGVtXpvC+8NSK8JOrREMA7
lDoJNEkxKhgGqrc9TqNHicQiuM0ZZv4hiNDw9I2w46r3iRPnAJUwhlwHeU4SVvxW
. . . CUT . . .
```

To view the information details in a certificate, use:

```
openssl x509 -in www.example.com.crt -text | more
```

Now we need to update the SSL entries in the the ssl.conf located in /etc/httpd/conf.d or in the httpd.conf file, or possibly the httpd-ssl.conf file located in the conf/extra directory.

```
ErrorLog logs/ssl_error_log
TransferLog logs/ssl_access_log
LogLevel info
SSLEngine on
SSLProtocol all -SSLv2
```

```

SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:-LOW:-SSLv2:-EXP
SSLCertificateFile /etc/pki/tls/certs/www.example.com.crt
SSLCertificateKeyFile /etc/pki/tls/certs/www.example.com.key
# Default CA file, can be replaced with your CA's certificate.
SSLCACertificateFile /etc/pki/tls/certs/ca-bundle.crt
SSLVerifyClient none

```

The SSLProtocol line disables the weak SSLv2 protocol, which is important. In addition, the SSLCipherSuite selects which ciphers are allowed. The openssl command can be very useful in debugging and testing the SSL configurations. See <http://www.openssl.org/docs/apps/ciphers.html> as well as OWASP testing tips http://www.owasp.org/index.php/SSL/TLS_Testing:_support_of_weak_ciphers

The CA file can be replaced by your own CA, or the self signed certificate if you did not setup a trusted certificate. If you did use an accepted Certificate Authority, you can use the certificate of the CA instead of the CA bundle, to speed up the initial SSL connection. Lastly, you can start or restart the httpd service verify correct functioning with your favorite browser:

```
# service httpd start
```

L1 21. Deny HTTP TRACE Requests with Mod Rewrite

Description

A relatively newer technique for both mapping HTTP request paths and for identifying possible new attack targets is the use of the HTTP TRACE Request Method. The HTTP "TRACE" Method is essentially traceroute for Web traffic. It is an application layer loop-back of the request. It will send a HTTP packet to a destination host and then return a packet describing the path, which was traveled. The interesting reconnaissance technique is that the TRACE Method will identify intermediate Proxy and Load Balancer Servers that are between the client and the destination host. The TRACE method cannot be controlled via the Limit/LimitExcept Apache directives. In order to prevent the Apache web server from responding to TRACE requests, add in an additional Mod_Rewrite rule:

```

RewriteEngine On
RewriteLog /usr/local/apache/logs/rewrite.log
RewriteLogLevel 2
RewriteRule    [^a-zA-Z0-9|\.|/|_|-] - [F]
RewriteCond   %{REQUEST_METHOD} ^TRACE
RewriteRule   .* - [F]

```

The bolded rules above will deny all connection attempts if the HTTP Request Method starts with "TRACE". Since we are using CGI scripts for our 403 status codes messages, we will receive an email alert notifying us that someone has tried to use this request method.

[Back to Top ^](#)

LEVEL II -- Prudent Security Beyond the Minimum Level

Level-II security configurations vary depending on network architecture and server function. These are of greatest value to system administrators who have sufficient security knowledge to apply them with consideration to the operating systems and applications running in their particular environments.

Important – Level II Benchmark settings also differ from Level I settings in that you do NOT have to apply all of the Level II settings to address each security issue. There are settings, which are an either/or option. For example, in order help identify potential security issues and provide more information during an investigation, you could update the CustomFormat setting for the access_log file or you could use the custom CGI scripts. You do not need to do both. The reader should review all Level II settings before determining which sections to implement. Level II settings expects the reader to have sufficient knowledge of the security issues involved to determine the best option for their environment.

[Back to Top ^](#)

L2 1. Create Web User Account Disk Quota

Description

The use of the quota settings should be evaluated by each organization. This setup may not work appropriately if the Apache web server uses more interactive application add-ons which need to create files such as PHP, MySQL, etc. Before implementing any disk quotas, it is necessary to identify how the system is partitioned. While partitioning will vary from system to system, the important partition in this scenario is the "/var" partition. Your partition naming convention may be different and most of the references in this document will use the */usr/local* path description. This will be the partition that functions as both the Apache html document root and will hold the apache user's home directory.

The goal of these commands is to apply a disk quota to the new system account "apache" which will be the userid that the Apache web server runs as. By placing a restrictive quota that will not allow the apache user to create ANY new files on the partition which the web server document root resides, we can defend against many of the typical methods attackers use to deface or compromise a web site. This technique would deter attacks where the web server is tricked by an insecure CGI script into executing a command such as:

```
"/bin/echo `You've been Owned!!!` >
/usr/local/apache/htdocs/index.htm"
```

After you issue the "quotaon -v" command, the next command is to verify that that partition now has the quota parameter enabled on it. This is accomplished by looking for the "quota" parameter (bolded below) on the partition in the /etc/mnttab file.

Important- When editing the quota for the user with the edquota utility, take care! The fields in the edquota edit session are as follows:

```
# quotaon -v <file_system>
```

```
fs mount_point blocks (soft =number, hard =number ) inodes (soft
=number, hard =number)
```

Where a block is considered to be a 1024 byte (1K) block.

It is easy to unintentionally implement an undesired effect. Our goal is to disallow any new files to be created or owned by the apache user. Note that setting a quota to 0 disables that quota; this is why we set the hard limits to 1. We then executed the touch command to create a file for the apache user within the proper partition. The apache user's quota has now been reached.

Once we have set a hard quota for the apache account, we must test the setting. We need to su to the apache user to test this quota. Notice the status message above when we log into this account. The system lets us know that we have reached our limit for the /var partition immediately upon entering our shell. We then try and create a new file in the DocumentRoot directory and we are denied with the status message below saying that creating this file would put us over the hard quota limit.

Action: Execute the following commands:

In the following sessions, the partition that holds the Apache DocumentRoot is /var. Edit the /etc/fstab and add the ,usrquota mount option.

```
# grep /var /etc/fstab
/dev/hdb1      /var          ext3          defaults,usrquota  1 2

# mount -o remount /var

# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol100
                15G   6.5G   7.3G   47% /
/dev/hda1       289M   23M   252M    9% /boot
/dev/hdb1       9.3G   5.8G   3.2G   65% /var
none            125M    0    125M    0% /dev/shm
# touch /var/aquota.user
# edquota apache
```

Within the vi session – change both of the block and inode hard settings to equal 1:

```
Disk quotas for user apache (uid 48):
Filesystem      blocks      soft      hard      inodes      soft      hard
/dev/hdb1       0            0          1          0            0          1
```

Execute more commands:

```
# quotacheck -avugm
# quotaon -avugm

# touch /var/www/test
# chown apache /var/www/test
# repquota -v /var
*** Report for user quotas on device /dev/hdb1
Block grace time: 7days; Inode grace time: 7days
Block limits          File limits
User      used      soft  hard  grace  used  soft  hard  grace
-----
root      -- 5992276    0    0      10    0    0
apache    ++      8    0    1      2    0    1

Statistics:
Total blocks: 6
Data blocks: 1
Entries: 2
Used average: 2.000000
```

Execute the following commands to verify the quota for the new account:

```
# su -m apache
# id
uid=48(apache) gid=48(apache) groups=48(apache)
# cd /var/www
# > test2
hdb1: write failed, user file limit reached.
bash: test2: Disk quota exceeded
```

[Back to Top ^](#)

L2 2. Prevent the Web Server from Accessing OS Commands

Description

Attackers will often try to access OS level commands by tricking the web server with specially crafted HTTP requests. The goal is to use HTTP/Web Server to execute OS commands to deface a webpage, access sensitive data (viewing files such as /etc/passwd), download tools (via Trivial FTP or Wget) and implement backdoors (by spawning command shells on unused ports). Additional information is available at the Cgisecurity.com website –

<http://www.cgisecurity.com/papers/fingerprint-port80.txt>.

Another example of how this problem reared its ugly head was when the OpenSSL/Apache Slapper Worm was making its rounds - <http://www.cert.org/advisories/CA-2002-27.html>. Part of this worm's propagation process was to upload the source code for itself and then try and compile it on the target host by accessing the system's compiler (cc/gcc). One mitigation technique that could have prevented propagation would have been to disallow access to the compiler by altering the ownership/permissions - <http://www.hackinglinuxexposed.com/articles/20020924.html>.

Two other Level II settings address this same issue:

- Chroot (Next Section)
- Mod_Security Directives

You should review these two sections and decide which one to implement for your environment.

Action: Update the default OS permissions of the standard Unix utilities

CAUTION – Care should be taken if you implement this setting!

The following action will remove the EVERYONE permissions from many of the system binaries targeted by attackers. System Administrators should review these directories and test to verify that local users have proper access to these binaries by creating appropriate groups.

Execute the following commands to update the default file permissions of the /bin, /sbin, /usr/bin, /usr/sbin and /usr/dt/bin directory contents.

```
# for dir in /bin /sbin /usr/bin /usr/sbin /usr/dt/bin /var /opt; do \
chmod -R 750 $dir \
done
# ls -l /bin/cat
-r-xr-x--- 1 root  bin          10092 Jul 10  2000 /bin/cat
```

[Back to Top ^](#)

L2 3. CHROOT Apache

Description

IMPORTANT – This section describes the traditional method of running Apache in a Chrooted filesystem. If you plan on implementing Mod_Security, then you may want to use it's built-in Chroot function instead as it is much easier to configure.

This part focuses on preventing Apache from being used as a point of break-in to the system hosting it. Apache by default runs as a non-root user, which will limit any damage to what can be done as a normal user with a local shell. Of course, allowing what amounts to an anonymous guest account falls rather short of the security requirements for most Apache servers, so an additional step can be taken - that is, running Apache in a chroot jail.

The main benefit of a chroot jail is that the jail will limit the portion of the file system the daemon can see to the root directory of the jail. Additionally, since the jail only needs to support Apache, the programs available in the jail can be extremely limited. Most importantly, there is no need for setuid-root programs, which can be used to gain root access and break out of the jail.

Pros and Cons of Chroot

- If apache is ever compromised, the attacker will not have access to the entire file system.
- Poorly written CGI scripts that may allow someone to access your server will not work.
- There are extra libraries you'll need to have in the chroot jail for Apache to work.
- If you use any Perl/CGI features with Apache, you will need to copy the needed binaries, Perl libraries and files to the appropriate spot within the chroot space. This includes /bin/mail, /bin/lis, etc...
- The same applies for SSL, PHP, LDAP, PostgreSQL and other third-party programs.
- Chroot usually requires extensive testing to ensure proper functionality/security are working properly.

Action: Execute the following commands to create a Chroot file system on a Solaris host.

In the terminal screen below, there is an example shell script (called chroot_setup.sh) included in the CIS Apache Benchmark archive, which will setup the file system. You may copy this script and update it per your environment.

Execute the following commands to execute the chroot_setup.sh script and then start/stop the Apache server in a chroot environment:

```
# ./chroot_setup.sh
# chroot /var/chroot /usr/local/apache/bin/apachectl start
/usr/local/apache/bin/apachectl start: httpd started
# chroot /var/chroot /usr/local/apache/bin/apachectl stop
/usr/local/apache/bin/apachectl stop: httpd stopped
```

[Back to Top ^](#)

Level 2 Benchmark Settings for the Apache Configuration File (httpd.conf)

Unfortunately, most web server's default configurations are not adequate for deployment on today's Internet. Usually these default settings are configured with a too "open" mindset. In actuality, the exact opposite of the aforementioned statement should be the standard. This is known as the "Principal of Least Privilege." Access controls should start off with total restriction and then access rights should be applied appropriately. If a production web server is bound for the Internet, various web server system settings need to be changed and/or implemented.

In the following sections, we will be discussing many different Apache configuration settings. For most of these settings, we will be altering for a desired security effect. There are some settings, however, which are the default setting and it is recommended that the reader merely confirm this particular setting. The examples highlighted in all of the pseudo-terminal screens are the **RECOMMENDED** setting.

[Back to Top ^](#)

L2 4. ErrorLog - Syslog

Description

Apache has the capability to send its error log output to the local Syslog daemon process by setting the ErrorLog directive to "ErrorLog syslog". This directive specifies that all of the error messages get sent through the syslog facility. We want to ensure the integrity of your error logs in case the web server ever gets compromised. After a compromise, the log files on the host cannot be trusted, since the attacker could have easily altered them. In the steps below, we have configured the syslog facility to log all of the errors locally to the normal /usr/local/apache/logs/error_log file, as well as, log remotely to the syslog facility on a secure, remote host. This ensures that the error logs have not been altered if we are investigating a break-in.

Action: Execute the following commands

Edit the httpd.conf file and update the ErrorLog directive:

```
#
# ErrorLog: The location of the error log file. If you do
# not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host
# will be logged here. If you *do* define an error
# logfile for a <VirtualHost> container, that host's
# errors will be logged there and not here.
#
ErrorLog syslog:local7
```

This configuration will send all error log entries to syslog and be logged with the "local7" facility. You can specify other syslog facility levels if you wish to segregate your Apache syslog data from your other system messages. Keep these issues in mind when using syslog for Apache logging:

- The syslog daemon will not create files. If you are logging to a file (as specified in the syslog.conf configuration file) that file must already exist and have permissions that allow the syslog daemon to write to it.
- You must restart the syslog daemon for it to recognize changes to its syslog.conf configuration file.
- The syslog daemon must be active prior to starting Apache.
- Apache will default the facility to "local7" if you omit the facility name from the ErrorLog directive (that is "ErrorLog syslog").
- The syslog facility name must be one that is recognized by both Apache and the syslog.h header file. The facility names "local0" through "local7" are explicitly set aside for your use.
- Although "local0" through "local7" are recommended user facility names, here is the complete list of names recognized by both Apache and TPF's syslog.h: auth, cron, daemon, kern, local0, local1, local2, local3, local4, local5, local6, local7, lpr, mail, news, syslog, user, and uucp.
- You won't see the normal Apache startup/shutdown messages when you use syslog with your Apache error log.

Verify that the Apache error messages are being sent to syslog.

```
# apachectl start
# tail -2 /var/log/messages
Aug  8 15:39:18 netwk8 httpd[3345]: [notice] Digest: done
Aug  8 15:39:19 netwk8 httpd[3345]: [notice] Apache configured -- resuming
normal operations
```

Edit the /etc/syslog.conf file to log locally and to add in the ability to send the Apache error data off to a remote syslog host.

```
# vi /etc/syslog.conf
# grep local6 /etc/syslog.conf
local6.* @IP_of_Remote_Host
# killall -v -HUP syslogd
```

Verify that the syslog data is being captured correctly on the remote syslog host.

```
# hostname
remote_sysloghost
# echo "SYSLOGD_OPTIONS="-m 0 -r" >> /etc/sysconfig/syslog
# service syslog restart
#
# tail -f /var/log/messages
Aug 8 15:44:19 netwk8 httpd[3417]: [error] [client 10.1.2.16] File does not
exist: /var/www/html/foo
```

[Back to Top ^](#)

L2 5. Tracking Security Related HTTP Status Codes

Description

SysAdmins need to keep tabs on all of these security related issues with their web servers. To assist with this monitoring, the web server can be configured to use custom CGI error response pages for both 401 and 403 server response codes. The CIS Apache Benchmark has included a template.cgi script within the ZIP archive. Please review this file for further information on the code/functionality discussed below.

- **Use custom CGI error pages for – 401/403 Status Codes**
 - Scripts uses “sleep 10” command before response
 - This slows down Brute Force Tools
- **The CGI scripts automate many important tasks**
 - Uses Environmental Variables from the printenv script – sends this info to WebAdmin instead of to the client
 - Issues HTML page to attacker with a Warning Banner
 - Notifies WebAdmin via Email
 - Throttles email after the 5 illegal connection from the client IP to prevent CGI Denial of Service or email bombing
- **Emails contain the following info:**
 - The CGI Environment Variables (Full Client/Server HTTP Header Info)
 - A URL hyperlink to immediately run a Traceroute and WHOIS on the attacker's IP address from different public websites.

The hyperlink feature, within the e-mail message, is useful for tracking down the appropriate "network abuse" contact personnel responsible for the attacker's IP segment. While not every 401 and 403 message warrants these investigative actions, repeated errors identified from a certain IP address should be handled appropriately. This CGI alert e-mail system facilitates the prompt notification of proper personnel. This information is critical when investigating web based attacks, as it provides a “snap-shot” of the malicious request.

Action: Execute the following commands.

Update the ErrorDocument directives in the httpd.conf file.

```
ErrorDocument 400 /cgi-bin/400.cgi
ErrorDocument 401 /cgi-bin/401.cgi
ErrorDocument 403 /cgi-bin/403.cgi
ErrorDocument 405 /cgi-bin/405.cgi
ErrorDocument 406 /cgi-bin/406.cgi
ErrorDocument 409 /cgi-bin/409.cgi
ErrorDocument 413 /cgi-bin/413.cgi
ErrorDocument 414 /cgi-bin/414.cgi
ErrorDocument 500 /cgi-bin/500.cgi
ErrorDocument 501 /cgi-bin/501.cgi
```

Each one of these CGI scripts should display the appropriate HTML error page/warning banner and then email the proper security personnel.

[Back to Top ^](#)

[L2 6. Mod Evasive – Apache Denial of Service Prevention Module](#)

Description

[Mod_evasive](http://www.nuclearelephant.com/projects/dosevasive/) (<http://www.nuclearelephant.com/projects/dosevasive/>) is an Apache security module which will allow us to manage repeated attempts by specific IP addresses to access specific URLs on our Apache server. Mod_evasive is an evasive maneuvers module for Apache to provide evasive action in the event of an HTTP DoS attack or brute force attack. It is also designed to be a detection tool, and can be easily modified to talk to ipchains, firewalls, routers, and etcetera. Detection is performed by creating an internal dynamic hash table of IP Addresses and URIs, and denying any single IP address from any of the following:

- Requesting a single page more than a few times per second
- Making more than 50 concurrent requests on the same child per second
- Making any requests while temporarily blacklisted (on a blocking list)

This method has worked well in both single-server script attacks as well as distributed attacks, but just like other evasive tools, is only as useful to the point of bandwidth and processor consumption (e.g. the amount of bandwidth and processor required to receive/process/respond to invalid requests), which is why it's a good idea to integrate this with your firewalls and routers. This module instantiates for each listener individually, and therefore has a built-in cleanup mechanism and scaling capabilities. Because of this, legitimate requests are never compromised but only scripted attacks. Even a user repeatedly clicking on 'reload' should not be affected unless they do it maliciously.

HOW IT WORKS

A web request comes in. The following steps take place:

- The IP address of the requestor is looked up on the temporary blacklist
- The IP address of the requestor and the URI are both hashed into a "key". A lookup is performed in the listener's internal hash table to determine if the same host has requested this page more than once within the past 1 second.
- The IP address of the requestor is hashed into a "key". A lookup is performed in the listener's internal hash table to determine if the same host has requested more than 50 objects within the past second (from the same child).
- If any of the above is true, a 403 response is sent. This conserves bandwidth and system resources in the event of a DoS attack.

Once a single 403 incident occurs, mod_dosevasive now blocks the entire IP address for a period of 10 seconds (configurable). If the host requests a page within this period, it is forced to wait even longer. Since this is triggered from requesting the same URL multiple times per second, this again does not affect legitimate users.

The blacklist can/should be expanded to talk to your network's firewalls and/or routers to push the attack out to the front lines, but this is not required.

Action: Execute the following commands

First you must compile mod_dosevasive into the Apache httpd binary. You can do this by placing the mod_dosevasive.c file in the /install/apache_1.3.29/src/modules/standard directory. You then need to add the following lines when configuring Apache on the command line:

```
# pwd
/install/apache_1.3.29
# ./configure --activate-module=src/modules/standard/mod_dosevasive.c \
--enable-module=dosevasive \
-- CUT --
```

The command above will activate and enable the mod_dosevasive module in the new httpd binary.

Next, you will need to add in the appropriate httpd.conf entries for this module.

```
<IfModule mod_dosevasive.c>
    DOSHashTableSize    3097
    DOSPageCount        2
    DOSSiteCount        50
    DOSPageInterval     1
    DOSSiteInterval     1
    DOSBlockingPeriod   10
    DOSEmailNotify      webmaster@companyx.com
</IfModule>
```

The entries above have the functionality:

DOSHashTableSize

The hash table size defines the number of top-level nodes for each child's hash table. Increasing this number will provide faster performance by decreasing the number of iterations required to get to the record, but consume more memory for table space. You should increase this if you have a busy web server. The value you specify will automatically be tiered up to the next prime number in the primes list (see mod_dosevasive.c for a list of primes used).

DOSPageCount

This is the threshold for the number of requests for the same page (or URI) per page interval. Once the threshold for that interval has been exceeded, the IP address of the client will be added to the blocking list.

DOSSiteCount

This is the threshold for the total number of requests for any object by the same client on the same listener per site interval. Once the threshold for that interval has been exceeded, the IP address of the client will be added to the blocking list.

DOSPageInterval

The interval for the page count threshold; defaults to 1 second.

DOSSiteInterval

The interval for the site count threshold; defaults to 1 second.

DOSBlockingPeriod

The blocking period is the amount of time (in seconds) that a client will be blocked for if they are added to the blocking list. During this time, all subsequent requests from the client will result in a 403 (Forbidden) and the timer being reset (e.g. another 10 seconds). Since the timer is reset for every subsequent request, it is not necessary to have a long blocking period; in the event of a DoS attack, this timer will keep getting reset.

DOSEmailNotify

This directive will specify who to send an email alert to when mod_dosevasive has blackholed a client.

[Back to Top ^](#)

L2 7. Buffer Overflow Protections

Description

Buffer overflow vulnerabilities stem from problems in string handling. Whenever a computer program tries copying a string or buffer into a buffer that is smaller than the string, an overflow is caused. If the destination buffer is overflowed sufficiently it will overwrite various crucial system data. In most situations an attacker can leverage this to takeover a specific program's process, thereby acquiring the privileges that process or program has. (11)

In addition to OS network stack tuning, the Apache directives listed below limit the size of the various HTTP header "strings" being copied. Implementing these directives greatly reduces the chance of a successful buffer overflow.

- **LimitRequestBody** setting will limit the total size of the HTTP request body that is sent to the Apache web server. These parameters usually come into effect during HTTP PUT and POST requests where the client is sending data back the web server from a form, or sending data into a CGI script. The setting below will restrict the request body size to be no more than **10K**. You will need to increase this size if you have any forms that require larger input from clients.
- **LimitRequestFields** limits the number of additional headers that can be sent by a client in an HTTP request, and defaults to 100. In real life, the number of headers a client might reasonably be expected to send is around 20, although this value can creep up if content negotiation is being used. A large number of headers may be an indication of a client making abnormal or hostile requests of the server. A lower limit of **40** headers can be set with the setting below.
- **LimitRequestFieldsize** limits the maximum length of an individual HTTP header sent by the client, including the initial header name. The default (and maximum) value is 8190 characters. We can set this to limit headers to a maximum length of **100** characters with the setting below.
- **LimitRequestLine** limits the maximum length of the HTTP request itself, including the HTTP method, URL, and protocol. The default limit is 8190 characters; we can reduce this to **500** characters with the line below. The effect of this directive is to effectively limit the size of the URL that a client can request, so it must be set large enough for clients to access all the valid URLs on the server, including the query string sent by GET requests. Setting this value too low can prevent clients from sending the results of HTML forms to the server when the form method is set to GET.

CAUTION – You should test these setting extensively prior to deploying into production. You should verify that all CGI scripts and uploading scripts work appropriately with these settings, otherwise, you can effectively cause a Denial of Service attack against yourself!

Action: Execute the following commands

Edit the httpd.conf file and add in the following directives:

```
LimitRequestBody 10240
LimitRequestFields 40
LimitRequestFieldsize 100
LimitRequestline 500
```

[Back to Top ^](#)

L2 8. URL Inspection with Mod Rewrite

Description

Manipulating the data sent between the browser and the web application to an attacker's advantage has long been a simple but effective way to make applications behave in an undesired manner. In a badly designed and developed web application, malicious users can modify things like prices in web carts, session tokens or values stored in cookies and even HTTP headers. No data sent from the client can be trusted – validation mechanisms must be in place to validate any parameter, which will accept user input. Additionally, no data sent to the browser can be relied upon to stay the same when returned to the web server unless the data is cryptographically protected. (3)

Attackers will often send malicious HTTP requests to the web server which will contain non-alphanumeric characters. Besides the advanced functionality of some CGI scripts, all HTTP requests should only contain either letters, numbers, period, dash or underline characters. Any other character in a request could be used to trick the web server or application in to executing undesired code. Examples of common HTTP attack character are as follows (These example are taken from <http://www.cgisecurity.com/papers/fingerprint-port80.txt>) -

- **Directory Traversal** – “.”, “..”, “...”
http://host/cgi-bin/lame.cgi?file=../../../../etc/motd
- **Hex Value** – “%20” Space, “%00” Null Requests
http://host/cgi-bin/lame.cgi?page%00=ls%20-a|
- **Pipe Request** – “|”
http://host/cgi-bin/lame.cgi?page=ps%20-ef|grep%20root
- **Semi-Colon Requests** – “;”
http://host/cgi-bin/lame.cgi?page=id;uname%20-a
- **Redirect Requests** – “<”, “>”, “>>”
http://host/cgi-gi?page=echo%20“You’re%20owned”>index.htm
- **System Commands** – “ls”, “echo”, “cat”, “tftp”, “ps”
http://host/cgi-bin/bad.cgi?doh=ps%20-aux

We want to use Mod_Rewrite to inspect the entire client HTTP URL Requests. We will define a rule set which will only allow **ACCEPTABLE** characters:

- Upper/Lowercase Letters – [a-zA-Z]
- Numbers – [0-9]
- Forward Slash – “/”
- Period – “.”
- Dash – “-”
- UnderScore – “_”

If a client makes a URL request that contains characters other than the acceptable ones listed above, it will be denied.

Action: Execute the following commands

Edit the httpd.conf file and add in the following directives:

```
RewriteEngine On
RewriteLog /var/log/httpd/rewrite.log
RewriteLogLevel 2
RewriteRule    [^a-zA-Z0-9|\.\|/|_|-] - [F]
```

The directives above are telling Mod_Rewrite to ONLY accept these URL characters. If a URL request has any character other than those defined, then the request will be automatically FORBIDDEN. This will then redirect the request to our 403 CGI alerting script for email notification. After adding the appropriate RewriteRule lines from above to the httpd.conf file, it is a good idea to monitor the RewriteLog file to verify the effects of these new rules. We do not want our new rules to register any False Positives and potentially match any legitimate traffic. This is why we designated the RewriteLogLevel of 9 during our testing phase. If your web server needs to use additional meta-characters (perhaps you are using CGI scripts which need to use the = or & characters) then you can simply add them into the RewriteRule above.

CAUTION – You should test these setting extensively prior to deploying into production. You should verify that all CGI scripts and uploading scripts work appropriately with these settings, otherwise, you can effectively cause a Denial of Service attack against yourself!

Important - After you have completed your testing phase of these RewriteRules, you should probably change the RewriteLogLevel down from 9 to something less verbose like the 2-3 range. Leaving this setting at 9 will cause your rewrite.log file to grow extremely fast and could fill up the local partition.

[Back to Top ^](#)

L2 9. Mod_Security – Level II Settings

Description

We previously discussed Mod_Security in the Level I section. We will now cover some advanced Mod_Security settings which, once implemented, can greatly increase the security of our Apache server.

Action: Add the following rules to httpd.conf to implement advanced Mod_Security settings

As mentioned earlier, Mod_Security can function as an HTTP IDS/IPS system especially when an attacker is using SSL as the transport protocol. In this setup, a NIDS such as Snort is not much use since the encryption has blinded Snort's signature analysis capabilities. With Mod_Security's flexible, expandable rule sets, we can still leverage Snort's outstanding web attack signature files. The only obstacle with this idea is to translate the Snort signature formats into the Mod_Security format. The screen below shows the normal Snort attack signature file entry format. We are interested in the "uricontent/content" portions of the signatures.

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC Cisco IOS HTTP
configuration attempt"; uricontent:"/level/*/exec/"; regex; flags:A+;
classtype: web-application-attack; reference:bugtraq,2936; sid:1250; rev:3;)
```

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"WEB-MISC Netscape
Enterprise directory listing attempt"; content:"INDEX "; offset:0; depth:6;
flags:A+; reference: cve,CAN-2001-0250; reference:bugtraq,2285;
classtype:web-application-attack; sid:1048; rev:4;)
```

You can extract these portions from the various web-attack signature files to translate these into Mod_Security signatures by using the snort2modsec.pl script. Both the convert script and an example snortmodsec-rules.txt signature file are included in the CIS Apache Benchmark archive. After extracting the appropriate information from the various "WEB" attack Snort signature files, we can have a Mod_Security signature file with more than **300+** signatures! Here is an example of the new Snort/Mod_Security signatures:

```
# WEB-ATTACKS ps command attempt
SecFilterSelective THE_REQUEST "/bin/ps"

# WEB-ATTACKS /usr/bin/id command attempt
SecFilter "/usr/bin/id"

# WEB-ATTACKS /usr/bin/g++ command attempt
SecFilter "/usr/bin/g\+\\"
-- CUT --
```

These entries use Regular Expression matching. **CAUTION** – If you use the snort2modsec.pl script, it will try to escape all meta-characters in the resulting rules file. This is needed so that mod_security will compile the regular expression correctly. If you do not use the mod_security script to translate signatures or you are creating your own, please keep this in mind. For instance, the final entry on the slide shows how I had to escape the "+" signs with back-slashes. The URL Requests will be inspected, and if the alert signature between the "" is found anywhere in the request it is acted upon. The SecFilter directive tells Mod_Security that you want to look for the text strings anywhere in the request, including the Post Payload if you have "SecFilterScanPost On" set. If you want to create attack signatures for specific portions of the client request headers, you will need to use the SecFilterSelective directive. Variable names are the same as in mod_rewrite.

Chrooting with Mod_Security

Apart from simplicity, mod_security chrooting brings another significant advantage. Unlike external chrooting (mentioned previously) mod_security chrooting requires *no additional files to exist in the jail*. The chroot call is made after web server initialization but before forking. Because of this, all shared libraries are already loaded, all web server modules are initialized, and log files are opened. You only need your data in the jail. There are some cases, however, when you will need additional files in jail, and that is if you intend to execute CGI scripts or system binaries. They may have their own file requirements. If you fall within this category then you need to proceed with the external chroot procedure as normal but you still won't have to maintain extra copies of the Apache libraries, etc

[Back to Top ^](#)

L2 10. Web Server Fingerprinting

Description

Knowledge of software vendor and version information provides an advantage to attackers in penetrating a system. In a sophisticated cyber attack, information about potential vulnerabilities in the target system is first gathered then a penetration attempt is launched. Although the security of a system should not be based solely on obscurity, it is desirable to protect against attempts to gain system information.

Unfortunately, few vendors have taken the advice of obscuring the version of software being used. Since HTTP servers do not generally provide a simple means for modifying or omitting the server's self description (e.g. as a configuration file option) a server administrator cannot easily manipulate this information. However, through modification of source code (or even the binary using reverse engineering) it is possible to remove or obscure the server header from an HTTP response, as we covered in a Level I section by editing the Apache httpd.h file.

Even after hiding the server banner information a determined attacker can still learn the server's vendor and version number with a high degree of confidence. These methods uses fingerprinting techniques to

gather information about a server's identity from HTTP headers returned from a variety of "normal" and "abnormal" Requests. (12)

IMPORTANT – Web Server fingerprinting countermeasures do NOT replace the need for other security measures! These settings will make the attacker's reconnaissance more difficult, however, they do not directly provide any real security for your Apache server.

For further information on Web Server Fingerprinting techniques and tools:

- http://www.whitehatsec.com/presentations/Black_Hat_Singapore_2002/Black_Hat_2002-Singapore.ppt
- <http://net-square.com/httpprint/>

Action: Alter the default headers to confuse fingerprinting techniques/tools

There are three basic choices for defeating fingerprinting analysis:

- **Emulation** – Involves mimicking the characteristics of another type of web server. Since Apache is open source, it is possible to achieve a certain level of emulation through source code editing and header manipulation. Even with extensive modifications, if an attacker uses advanced fingerprinting techniques, they will be able to correctly identify the true application version. Below is an example for implementing some bogus IIS headers by using the Header directive in the httpd.conf file:

```
Header set X-Powered-By "ASP.NET"  
Header set X-AspNet-Version "1.1.4322"
```

- **Dynamic** – The accuracy of current fingerprinting tools rely heavily on the use of static, or non-changing response tokens. If the target web server's headers were "dynamic", then this would confuse most fingerprinting tools since the headers would not match those stored in the fingerprint tool's database. It is a non-trivial task to implement this type of functionality into most web servers. One way to accomplish this task is to implement the Mod_Security module. With Mod_Security, an Apache server will "act" differently (meaning it will deny certain requests) when scanned with most current web fingerprinting applications.
- **Complex Architecture** – The last option is to mimic complex web architectures by adding bogus HTTP headers into the responses sent by the Apache server. When attackers are footprinting a target's environment, they use numerous tools/techniques to try and enumerate the network topology. By accurately identifying intermediate routers, firewalls and proxy servers, the attackers can formulate the appropriate attack strategies. Below are some fake headers, which would indicate that some form of caching/proxy server was being utilized.

```
Header set Via "1.1 proxy.company.com Squid/2.4.STABLE7"  
Header set X-Cache "MISS from www.company.com"
```

Further proxy/caching information is available at the Netcraft website – <http://uptime.netcraft.com/up/accuracy.html#os>

[Back to Top ^](#)

L2 11. Monitoring the ErrorLog File with SWATCH

Description

Our CGI scripts will catch suspicious HTTP status codes, which are normally generated during an attack, however they will not identify all web server problems. Apache provides important information regarding the status of the httpd processes within the ErrorLog file. This file may yield valuable information when an attack is underway. It is for this reason that the ErrorLog file should be monitored in real-time and have email alerts should be sent out for suspicious entries. Question: What error_log entries should we look for?

The best way to identify error messages to monitor is to search for the "ap_log_error/perror/error" messages from within the Apache source code. By searching files from within the /install/apache_1.3.29/src/main and /install/apache_1.3.29/src/modules/standard directories, we can identify text strings to monitor. Remember, Apache can only log error messages, which are defined within these files. If it is not defined within these sections of code, it will not be reported.

Action: Monitor Error Messages

The session below shows an example of searching the /install/apache_1.3.29/src/main directory for error strings. As you can see, the ap_log_error functions will log all sorts of information that we would like to monitor. If our Apache server ever reports these messages within the error_log file, we will want to investigate.

```
# pwd
/install/apache_1.3.29/src/main
# egrep -A rerror *.c
--
http_core.c:      ap_log_rerror(APLOG_MARK, APLOG_NOERRNO|APLOG_ERR, r,
http_core.c-      "Invalid URI in request %s", r->the_request);
--
http_core.c:      ap_log_rerror(APLOG_MARK, APLOG_ERR, r,
http_core.c-      "file permissions deny server access: %s", r->filename);
--
http_protocol.c:  ap_log_rerror(APLOG_MARK, APLOG_NOERRNO|APLOG_ERR, r,
http_protocol.c-      "request failed: URI too long");
--
http_protocol.c:  ap_log_rerror(APLOG_MARK, APLOG_NOERRNO|APLOG_ERR, r,
http_protocol.c-  "request failed: erroneous characters after protocol
string: %s",
```

To efficiently identify these error messages, diligent monitoring of the web server's error logs is paramount. Manual monitoring of web server log files can be tedious, and in most circumstances, unfeasible due to their large size. Web Administrators lack the time to manually review log files on a daily basis. How can the process of monitoring the log files be addressed? [SWATCH](#) is a PERL program that continually monitors a specified file while it is being appended. SWATCH reads a configuration file which specifies regular expression text strings to identify. If a match is found within a log file, automatic actions can be taken.

SWATCH uses PERL to accomplish its pattern matching functionality, and therefore, care should be taken when defining attack patterns to monitor. Familiarity with Regular Expressions (RegExpr) to effectively define an attack signature is needed. SWATCH must be configured to identify the "content" portion of this request. If SWATCH finds an HTTP request that matches one of the specified RegExpr, both an e-mail and a pager message are sent to the WebAdmin. To report this attempt, use RegEx to convert the content section into a SWATCH acceptable format.

The session window below shows an example of using the error text strings extracted from the Apache source code files and using them with SWATC. If these strings are identified, the resulting error_log file entry will be emailed to webmaster@companyx.com with a Subject line of --[SWATCH Alert]--.

```
watchfor /Invalid URI in request|file permissions deny server access|request
failed\: URI too long|request failed\: erroneous characters after protocol
string|Invalid method in request|error reading the headers|client sent
invalid HTTP\0\9 request\: HEAD|client sent HTTP\1\1 request without
hostname|client sent an unrecognized expectation value|client used wrong
authentication scheme|chunked Transfer-Encoding forbidden/
  mail=webmaster\@companyx.com,subject=--[SWATCH Alert]--
  throttle 01:00;use=regex
```

Artificial Ignorance with SWATC

- Term coined by Marcus Ranum
- Artificial Ignorance How-To: <http://archives.neohapsis.com/archives/nfr-wizards/1997/09/0098.html>
- Opposite implementation of standard signature based rulesets
- Instead of looking for "What is not allowed", you specify "Normal behavior" to ignore
- By definition, if it is not normal, we want to know about it!

The session window below shows an example of using "Artificial Ignorance" within our SWATC config file. We use the "ignore" keyword and specify normal Apache startup and shutdown messages. We then add in the last Regular Expression signature directive "/*/*", that will catch any other message within the error_log file. You will need to conduct extensive testing to identify and filter out the normal messages generated from your Apache installation.

```
watchfor /Invalid URI in request|file permissions deny server access|request
failed\: URI too long|request failed\: erroneous characters after protocol
string|Invalid method in request|error reading the headers|client sent
invalid HTTP\0\9 request\:
HEAD|client sent HTTP\1\1 request without hostname|client sent an
unrecognized expectation value|client used wrong authentication
scheme|chunked Transfer-Encoding forbidden/
  mail=webmaster\@companyx.com,subject=--[SWATCH Alert]--
```

```
ignore /MyServer\2\0 configured \-\- resuming normal operations|Accept
mutex\: sysvsem|SIGHUP received. Attempting to restart|Digest\: generating
secret for digest authentication|Digest\: done/
```

```
watchfor /*/*/
mail=webmaster\@companyx.com,subject=--[SWATCH Alert]--
```

[Back to Top ^](#)

L2 12. Reverse Proxy for Protection

Description

The market for Web Application Firewalls/Gateways is booming. This new breed of security tools combat web attacks by erecting a "shield" in front of the Web server/application itself. These security tools--whether they're characterized as an application firewall, application-level security or a reverse proxy--add a layer of protection around the Web server, guarding it against attacks that aren't addressed by generic hardening techniques. (14)

Apache can function quite well as a Reverse Proxy server, and if used in conjunction with the Mod_Security module, it provides a low-cost solution for protecting multiple web servers. If Apache is

used as a DMZ Reverse Proxy server, it can protect web servers (perhaps you have an IIS server that you must provide access) by acting as a http filtering host. The Mod_Security module can identify and stop (Intrusion Prevention is the current buzz word) malicious http requests and provides extensive audit logging of all web requests. Mod_Security's audit_log feature is only available when Apache is using a "handler" such as a cgi-script or the proxy module. An example of the Mod_Security audit_log contents is shown below.

One caveat with regards to using Apache as a proxy server – you do **NOT** have to enable the full proxying module capabilities to achieve this security benefit. As a matter of fact, you should not turn on the proxy capabilities unless you know how to configure it properly. There have been many reports of people incorrectly implementing a proxy server and suffering the following security problems:

- If you do not place proper access control on the proxy module, it is possible that anyone on the Internet could use your server as an open proxy. This means that you could be flooded by clients who want to use your server to hide their identity. Examples of common open proxy abuses:
 - Vulnerability Scanning -> looking for vulnerable CGI scripts
 - Brute Forcing Attacks -> trying to break into password protected web pages
 - SPAM -> open proxies allow email spammers to hide their origins
 - Web Hit Counters -> using your proxy to connect to their own web site to inflate their guest counters
- Even worse than allowing external clients to use your proxy for their Internet activities, is the possibility of proxying external users into your internal network! If not secured properly, external users could possibly use your proxy server to enumerate internal hosts and ports. This is bad news.

To avoid many of the security issues related to implementing a proxy, Apache can actually perform basic proxying by using the ProxyPass Directive. This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. *Path* is the name of a local virtual path; *url* is a partial URL for the remote server.

Suppose the local server has address `http://www.companyx.com/`; then

```
ProxyPass / http://dmzhost1.comanyx.com/
ProxyPassReverse / http://dmzhost1.comanyx.com/
```

will cause a local request for the `<http://www.companyx.com/>` to be internally converted into a proxy request to `<http://dmzhost1.comanyx.com>`.

Warning: The [ProxyRequests](#) directive should be set **off** when using ProxyPass. This will prevent the open proxy issues described above.

Action: Enable Proxy Functionality

Execute the following commands to enable proxying capabilities to one other webserver:

Edit the `httpd.conf` file and enable the following directives:

```
<IfModule mod_proxy.c>
  ProxyRequests Off
```

```
##### Proxy Rules #####
ProxyPass / http://192.168.1.101/
ProxyPassReverse / http://192.168.1.101/
```

#####

</IfModule>

The session window below shows an example from the Mod_Security audit_log file.

As you can see, Mod_Security will log extensive details about each proxied request including both the client and server headers.

```
=====  
Request: 192.168.1.100 - - [Mon Aug 4 17:41:31 2003] "GET /index.html  
HTTP/1.0" 200 28641  
Handler: proxy-server  
-----  
GET /index.html HTTP/1.0  
Accept: /*/*  
Accept-Language: en-us  
Connection: Keep-Alive  
Host: www.companyx.com  
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)  
  
HTTP/1.0 200 OK  
Last-Modified: Sun, 13 Jul 2003 14:31:51 GMT  
Accept-Ranges: bytes  
Cache-Control: private  
Content-Type: image/gif  
Content-Length: 28641  
Connection: close  
=====
```

The session window below shows another Mod_Security audit_log entry. In this entry, the client was trying to access the following URL - /cgi-bin/finger. If Apache (with Mod_Security) was not being used as a proxy, then this request would have been received by the end web server. In this case, Mod_Security denied the connection and did not proxy the request because of an attack signature match.

```
=====  
Request: 192.168.1.100 - - [Mon Aug 4 17:41:42 2003] "GET /cgi-bin/finger  
HTTP/1.0" 403 743  
Handler: proxy-server  
-----  
GET /cgi-bin/finger HTTP/1.0  
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,  
application/vnd.ms-powerpoint, application/vnd.ms-excel, application/msword,  
application/x-shockwave-flash, /*/*  
Accept-Language: en-us  
Connection: Keep-Alive  
Host: www.companyx.com  
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)  
mod_security-message: Access denied with code 403. Pattern match "/finger" at  
THE_REQUEST.  
mod_security-action: 403  
  
HTTP/1.0 403 Forbidden  
Connection: close  
Content-Type: text/html  
=====
```

[Back to Top ^](#)

L2 13. Update the Apachectl Script for Email Notification

Description

This setting may not be feasible for large organizations with many Apache web servers. In order to keep track of when our Apache web server has been started, we will update the normal apachectl start script. We can add some lines to the default script, which will send out an email alert to the appropriate SysAdmins. The email lets the SysAdmins know that the web server has been restarted, who is currently logged onto the system and it will give the last few lines of the error_log file. This notification could aid in determining why the web server was restarted. If you receive these email alerts and they are at non-standard hours, you should investigate to determine if it is in fact malicious activity or actual server problems. In the example below, we are sending email to the local root account. In a production environment, you would want to specify the correct webmaster email distribution list. This information should also be sent to the appropriate WebAdmin's pagers.

Action: Execute the following commands:

Update the default apachectl start script's code to send email alerts to the appropriate personnel:

```
# pwd
/usr/local/apache/bin
# mv apachectl apachectl.orig
# vi apachectl
# egrep -C3 mail /tools/apachectl
    echo "$0 $ARG: httpd started"
    who > /tmp/error_log
    tail /usr/local/apache/logs/error_log >> /tmp/error_log
    /bin/mail -s 'Apachectl start - Has Been Used' root < /tmp/error_log
    rm /tmp/error_log
else
--
    echo "$0 $ARG: httpd started"
    who > /tmp/error_log
    tail /usr/local/apache/logs/error_log >> /tmp/error_log
    /bin/mail -s 'Apachectl startssl - Has Been Used' root < /tmp/error_log
    rm /tmp/error_log
else
-- CUT --
```

[Back to Top ^](#)

Appendix A – Apache Modules Listing

Module Name	Description	Security Risk	Recommend
Mod_mmap_static	Maps identified web pages directly into memory for fast access speeds	minimal	Disable
Mod_vhost_alias	Creates dynamically configured virtual hosts, by allowing the IP address and/or the Host: header of the HTTP request to be used as part of the pathname to determine what files to serve.	minimal	Disable
Mod_bandwidth	Enables server-wide or per connection bandwidth limits, based on the directory, size of files and remote IP/domain	Minimal. Will not significantly assist with denial of service attacks.	Disable
Mod_throttle	Intended to reduce the load on your server & bandwidth generated by popular virtual hosts, directories, locations, or users according to supported policies that decide when to delay or refuse requests. Also mod_throttle can track and throttle incoming connections by IP address or by authenticated remote user.	Minimal. Will not significantly assist with denial of service attacks.	Disable
Mod_env	This module allows for control of the environment that will be provided to CGI scripts and SSI pages. Environment variables may be passed from the shell which invoked the httpd process. Alternatively, environment variables may be	Enabling CGI and SSI within the httpd server may imply a significant security impact, however, the addition of mod_env is unlikely to increase the security risk significantly.	Enable if you are using the CGI scripts for ErrorDocuments, otherwise, Disable.

	set or unset within the configuration process.		
Mod_log_config	Provides for logging of the requests made to the server, using the Common Log Format or a user-specified format.	Server logging provides useful statistical and security functionality on the web server. See the section on auditing below for a discussion on log management	Enable, configure to use common log format.
Mod_log_agent	Provides logging of the client user agents	Server logging provides useful statistical and security functionality on the web server. See the section on auditing below for a discussion on log management.	Disable, use log_config instead.
Mod_log_referer	Provides logging of the referer page	Server logging provides useful statistical and security functionality on the web server. See the section on auditing below for a discussion on log management.	Disable, use log_config instead.
Mod_mime_magic	Determines the MIME type of a file by looking at a few bytes of its contents. This provides functionality over and above mod_mime.	Minimal. This does not significantly affect server security, but allows the mime-type of files to be correctly sent to the web browser client.	Disable by default, but enable subject to web server requirements if mod_mime is insufficient
Mod_mime	Determines the MIME type of a file by looking at the file extension	Minimal. This does not significantly affect server security, but allows the mime-type of files to be correctly sent to the web browser client.	Enable. This module is normally an essential prerequisite for normal operation.

Mod_negotiation	Provides a content negotiation capability for web data. Content negotiation, is the selection of the document (or image) that best matches the clients capabilities, from one of several available documents. An example would be where three different languages are supported by three (otherwise identical) web pages. Web browsers that specify a preference for Spanish (for example), may be sent the Spanish language version, whilst English language speakers will receive the English version.	Minimal.	Disable unless you have an identified requirement for content negotiation
Mod_status	This module provides information on server activity and performance through the meta-web-page /server-status	The server-status page can provide potential attackers with useful information about your web server configuration, from which targeted attack profiles can be derived.	Disable. Use ACLs if you must implement. Note that although this module is normally active, most apache configurations disable the /server-status link elsewhere in the configuration file.
Mod_info	This module provides information on server activity and performance through the meta-web-page /server-info	The server-info page can provide potential attackers with useful information about your web server configuration, from which targeted attack profiles can be derived.	Disable. Use ACLs if you must implement. Note that although this module is normally active, most apache configurations disable the /server-info link elsewhere in the configuration file.
Mod_include	This module facilitates Server Side Includes (SSI). SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology	SSI facilitates the provision of dynamic content, which can potentially be the result of a server-side executable (shell / perl scripts, or other executables). Allowing the execution of applications from the web server increases the risk profile of the web server. Passing user input to external applications may further increase that risk.	Disable unless the site administration benefits clearly outweigh the potential risk of enabling SSI. It is recommended that code evaluation/checking procedures be implemented for any applications that are called by an SSI-enabled page.
Mod_autoindex	Provides automatic index generation for directories within the webroot that do not have a default html page (eg: index.html).	Automatic index generation allows external users to see the entire contents of the directory. There are situations where	Disable

		this is appropriate, such as file archives. If there is an intention to rely on 'security through obscurity' to protect web resources, then this feature should be disabled.	
Mod_dir	This module redirects users to either an appropriate "index.html" file, or a automatically generated index (via autoindex) when a user requests a URL with a trailing slash character	This form of redirection is an accepted part of the normal operation of a web server. The security implications are minimal.	Disable
Mod_cgi	This module redirects facilitates the execution of external applications, generally in order to provide dynamic content to a web page.	CGI facilitates the provision of dynamic content, which can potentially be the result of a server-side executable (shell / perl scripts, or other executables). Allowing the execution of applications from the web server increases the risk profile of the web server. Passing user input to external applications may further increase that risk. Significant web server vulnerabilities have resulted from bugs in CGI code in the past.	Disable unless you are using CGI scripts for ErrorDocuments. It is recommended that code evaluation/checking procedures be implemented for any applications that are called by a CGI-enabled page.
Mod_asis	This module facilitates the provision of a particular file via HTTP, without prepending HTTP headers that are a normal part of the file delivery. Files can therefore include their own custom HTTP headers	Minimal.	Disable, unless there is a requirement for custom headers.
Mod_imap	This module facilitates server-side image-map processing	Minimal.	Disable unless required.
Mod_actions	This module provides for executing CGI scripts based on media type or request method – eg: a CGI script can be run whenever a file of a certain type is requested	CGI facilitates the provision of dynamic content, which can potentially be the result of a server-side executable (shell / perl scripts, or other executables). Allowing the execution of applications from the web server increases the risk profile of the web server. Passing user input to external applications may further increase that risk.	Disable unless you have a specific requirement, and the benefits clearly outweigh the potential risk of enabling CGI. It is recommended that code evaluation/checking procedures be implemented for any applications that are called by a CGI-enabled page

		Significant web server vulnerabilities have resulted from bugs in CGI code in the past.	
Mod_spelling	This module attempts to correct misspellings of URLs that users might have entered, by ignoring capitalization and by allowing up to one misspelling.	Minimal.	Disable.
Mod_userdir	This module allows apache to include within the web directory hierarchy, a specific directory within the home directories of local system users	Users can create a directory (such as public_html) within their home directories. With the addition of mod_userdir, apache will look within this directory when a request in the format of: http://localhost/~username is received. Files within user directories are generally outside the control of the normal site webmaster, and if CGI/SSI is used, can also be outside the control of the site security administrator.	Disable, unless there is a clear benefit to be gained, and only as a result of a risk assessment.
Mod_alias	This module allows an administrator to maintain multiple document stores, under different directory hierarchies, and map them into the web document tree. For example, although the default document root may be /www, the /data/applications/executables could be mapped to the /apps directory in the web tree. As such, a request for http://localhost/index.html would go to /www/index.html on the file system, whereas a request for http://localhost/apps/index.html, would go to /data/applications/executables/index.html on the file system.	Minimal.	Enable.
Mod_rewrite	Mod_rewrite is a complex module that provides a rule-based URL-rewriting facility. Mod_rewrite is particularly useful when a site upgrade leads to changes in URL locations, but the site wishes to	Mod_rewrite has no significant security implications.	Enable. It allows for filtering of identified malicious requests.

	allow users to retain their normal bookmarks, and still be able to get to the new information.		
Mod_access	Provides access control based on client hostname, IP address, or other characteristics of the client request	Mod_access provides access control based only on information provided by the connection layer, or the client browser. It is recommended that mod_access be used for access control only where the organization has control over the data provided. For example, access control by IP address is likely to be inappropriate for Internet connections, where the security administrator has no control over the IP address. Access control by IP address may be more appropriate for internal networks where address allocation and network monitoring facilitate a reduced risk profile	Enable for use with ACLs (IP, network names and hostnames)
Mod_auth	This module allows the use of HTTP Basic Authentication to restrict access by looking up users in plain text password and group files. Similar functionality and greater scalability is provided by mod_auth_dbm and mod_auth_db. HTTP Digest Authentication is provided by mod_auth_digest.	Mod_auth provides a very basic authentication and access control facility that is usually difficult to administer for large volumes of users. Basic unix 'crypt' format passwords are used, which could potentially be exported from /etc/passwd and /etc/shadow on unix systems to alleviate administration somewhat.	Enable for user ACLs. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication or LDAP authentication. If mod_auth is used, consider using a specific, designated authentication file outside the normal web document tree, rather than the alternative .htaccess files within the document directory.
Mod_auth_anon	This module allows the use of HTTP Basic Authentication to restrict access by looking up users in plain text password and group files. Similar functionality and greater scalability is provided by mod_auth_dbm and mod_auth_db. HTTP Digest Authentication is provided by mod_auth_digest.	Mod_auth provides a very basic authentication and access control facility that is usually difficult to administer for large volumes of users. Basic unix 'crypt' format passwords are used, which could potentially be exported from /etc/passwd and /etc/shadow on unix	Disable. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication, LDAP authentication, or ssh authentication using mod_auth_any. If mod_auth is used, consider using a specific, designated authentication file outside the normal web document tree,

		systems to alleviate administration somewhat.	rather than the alternative .htaccess files within the document directory.
Mod_auth_db	This module allows the use of Berkeley database files for authentication purposes.	Mod_auth_db provides a very basic authentication and access control facility that is usually difficult to administer for large volumes of users. Basic unix 'crypt' format passwords are used within the DB file, which could potentially be exported from /etc/passwd and /etc/shadow on unix systems to alleviate administration somewhat.	Disable. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication, LDAP authentication, or ssh authentication using mod_auth_any. If mod_auth_db is used, consider using a specific, designated authentication file outside the normal web document tree, rather than the alternative .htaccess files within the document directory.
Mod_auth_any	This module allows the use of an arbitrary command line tool to authenticate a user.	Mod_auth_any is a powerful authentication facility that enables apache to utilize external user databases (such as LDAP directories, or potentially even Windows 2000 active directory) to authenticate users against provided authentication details.	Disable by default. If authentication details need to be synchronized with an external database, consider using this functionality. Note that the supplied username and password are passed as command line arguments to the indicated authentication application. As such, users on the local system may potentially pick up the authentication information using the 'ps' command. Applications that verify the authentication information should also be evaluated in the context of buffer-overflow vulnerabilities, as the supplied userid/password may potentially contain overflow code. If mod_auth_any is used, consider using a specific, designated authentication file outside the normal web document tree, rather than the alternative .htaccess files within the document directory.
Mod_auth_dbm	This module allows the use of Berkeley DBM files for authentication purposes.	Mod_auth_dbm provides a very basic authentication and access control facility that is usually difficult to administer for large volumes of users. Basic unix 'crypt' format	Disable. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication or ssh authentication using mod_auth_any. If mod_auth_dbm is used,

		passwords are used within the DBM file, which could potentially be exported from /etc/passwd and /etc/shadow on unix systems to alleviate administration somewhat.	consider using a specific, designated authentication file outside the normal web document tree, rather than the alternative .htaccess files within the document directory.
Mod_auth_ldap	This module allows the use of an external LDAP database for authentication purposes.		Disable by default. Consider this authentication mechanism if the organization is interested in using an LDAP directory for authentication purposes.
Mod_auth_mysql	This module allows the use an external MYSQL database for authentication purposes	Mod_auth_mysql provides an authentication and access control facility. Basic unix 'crypt' format passwords are used within the database, which could potentially be exported from /etc/passwd and /etc/shadow on unix systems to alleviate administration somewhat	Disable. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication, LDAP, or ssh authentication using mod_auth_any.
Mod_auth_pgsq	This module allows the use an external postgresql database for authentication purposes	Mod_auth_pgsq provides an authentication and access control facility. Basic unix 'crypt' format passwords are used within the database, which could potentially be exported from /etc/passwd and /etc/shadow on unix systems to alleviate administration somewhat	Disable. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication, LDAP, or ssh authentication using mod_auth_any
Mod_auth_digest	This module is similar to mod_auth, but allows the use of MD5 digest-encrypted passwords, rather than basic unix CRYPT passwords	Mod_auth_digest provides an authentication and access control facility using MD5 encrypted passwords, as enabled on many recent Linux distributions	Disable. If authentication is required, consider alternative authentication mechanisms, including certificate-based authentication, LDAP, or ssh authentication using mod_auth_any
Mod_proxy	This module turns the apache web server into a web proxy server.	Care should be taken with the configuration of proxy servers, as if the intent is to facilitate internal organization access to external web sites, there is a risk that	If this is server is a normal web server then this module is not required for the normal operation and it should be disable. If this server is being used as a proxy or a reverse proxy then this module must

		the reverse could be enabled, allowing Internet users to potentially browse internal web servers.	be enabled.
Mod_cern_mata	This module facilitates the inclusion of custom CERN header data when a web page is served to a client	Minimal	Disable. This module is not required for the normal operation of a web server.
Mod_expires	Facilitates the inclusion of custom expiry headers within web pages	Minimal	Disable. This module is not required for the normal operation of a web server
Mod_headers	Facilitates the inclusion/modification/removal of headers within web pages	Minimal	Enable. We will use this module to insert bogus headers to help obfuscate both our web server software version and our web architecture.
Mod_usertrack	Allows the web site administrator to track the actions of individual users on a web site using cookies	It should be noted that it is a client/user choice whether to accept cookies from the site or not. As such, the data derived from this module should not be considered accurate or comprehensive	Enable is you want to insert bogus cookies to emulate a different web server (I.E. – ASPSESSIONIDGGGQQXC for Microsoft-IIS).
Mod_example	This is an example module only, and should not be enabled on production servers	Minimal	Disable. This module is not required for the normal operation of a web server
Mod_unique_id	This module generates a unique identifier for a URL that is (almost) guaranteed to be unique across a cluster of http servers	Minimal	Disable. For normal web server activity, even in a clustered environment, unique id's are not required
Mod_setenvif	This module allows for control of the environment that will be provided to CGI scripts and SSI pages, based on attributes associated with the client HTTP request. Environment variables may be passed from the shell which invoked the httpd process. Alternatively, environment variables may be set or unset within the configuration process. Environment variables can be set, for example, only if the User-Agent string provided by the client, matches "netscape".	Enabling CGI and SSI within the httpd server may imply a significant security impact, however, the addition of mod_setenvif is unlikely to increase the security risk significantly	The normal recommendation would be to disable this feature unless you have CGI/SSI enabled, and you have an identified requirement to pass specific, static, environment variables to your scripts based on items such as browser type/version. However, as the feature is used within most configuration files to force a HTTP 1.0 response (as opposed to HTTP 1.1) for older browser technology, the default for most web servers would be to enable this feature.

Libperl	This module allows a web author to embed a subset of the PERL language within a web page, to be acted upon by the web server prior to delivering HTML to the client	Enabling any active scripting feature within the httpd server can increase the risk to the web server if external user input is acted upon by the script in question	Disable this functionality unless you have a specific requirement for active scripting using the perl language. Note that although executing the perl script using CGI capabilities is an option, the perl interpreter is executed each time the CGI script is loaded. Using embedded perl via the perl module, only loads the interpreter once, therefore increasing average processing speed
Mod_php Libphp3 Libphp4	This module allows a web author to embed PHP (personal home page) language components within a web page, to be acted upon by the web server prior to delivering HTML to the client.	Enabling any active scripting feature within the httpd server can increase the risk to the web server if external user input is acted upon by the script in question.	Disable this functionality unless you have a specific requirement for active scripting using the PHP language
Libdav	This module implements DAV server capabilities within apache. DAV is a collaborative web development environment that allows multiple authors to update web data in a controlled fashion	DAV allows modification of web pages by remote users, and integrates into the default apache authentication and access control facilities. If DAV is enabled on a web server that also serves pages to the general public, consider either: 1) Using a reverse proxy server in front of the http server that blocks facilities such as "PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK" from non-internal sources, or 2) Using web-dav on a 'acceptance' server only, with changed data mirrored to the production (available to the internet) web server.	Disable this functionality unless you have a specific requirement for multiple users to update files. If DAV is required, analyze the risk to the infrastructure in the context of a risk assessment
Mod_roaming	Mod_roaming allows the use of an apache server as a Netscape Roaming Access server. This facilitates the storage of Netscape Communicator 4.5 preferences,	A HTTP server that implements Mod_roaming should generally be a special-purpose web server, only used for the	Disable this functionality unless you exclusively utilize netscape clients with roaming-profile capabilities. It is recommended that this be used only for intranet clients

	bookmarks, address books, cookies etc. on the server. Netscape Communicator web clients can be used to access and update the settings.	storage/management of roaming profiles. Both read and write protocols are implemented to facilitate roaming profile capabilities	unless an appropriate risk assessment has been conducted
Libssl	The apache SSL module facilitates the use of X.509 certificates to provide Secure-Sockets-Layer encryption (and potentially, authentication) capabilities to Apache	Web pages served via HTTPS will increase the processing requirements of your system, but provide a level of confidentiality between client web browser and the web server	Disable this functionality unless you require message confidentiality or authentication within an encrypted channel. Note that software or hardware x.509 authentication tokens can be supported with this module, assuming appropriate client-side infrastructure is in place.
Mod_put	This module supports uploads of web pages via the HTTP PUT method	Write access to your server web pages should be carefully considered in the context of an appropriate risk assessment. If mod_put is enabled on a web server that also serves pages to the general public, consider either: 1) Using a reverse proxy server in front of the http server that blocks facilities such as "PUT" from non-internal sources, or 2) Using mod_put on a 'acceptance' server only, with changed data mirrored to the production (available to the internet) web server.	Disable this functionality unless you have a specific requirement for non-local users to update files

Appendix B -- Build Apache From Source

The following the recommend build from source procedure. The examples are built on Solaris 8.

1. Apache Distribution Download

Question:

Are you planning to use the precompiled Apache httpd binary?

If the answer to the question above is NO – then the following steps are provided as an example.

Description

Many Unix Operating Systems come with a precompiled version of Apache. There are also numerous compiled binaries for various platforms located at <http://www.apache.org/dist/httpd/binaries/>. While these versions do help with the ease of installation and ease of patching, there are cases where compiling the Apache software from source is either necessary or beneficial:

Checking the PGP and/or MD5 checksum of the Apache distribution is vital to ensure that the code you have downloaded has been downloaded successfully without any errors.

Action: Download the Apache Source

Go to the following location for the Apache Distributions – <http://httpd.apache.org/download.cgi>

Download the appropriate current version – httpd-2.2.3.tar.gz

Download the PGP and/or MD5 Checksums for this distribution - httpd-2.2.3.tar.gz.asc or httpd-2.2.3.tar.gz.md5

[Back to Top ^](#)

2. Verify the PGP Digital Signature and/or the MD5 Checksum

Verify PGP Digital Signature Description

It's highly recommended that you validate the PGP digital signature of the source code download. This ensures that the download was not corrupted or tampered with as part of the download, and also ensures that the owner of the corresponding private PGP key signed the software tar file. If you download the appropriate public PGP public from one of the PGP key servers or a different source than one of the Apache mirrored websites, you have additional assurance of the authenticity of the software. The commands below will detect the required key, and download the public key from the default configure ppg key server, and then verify the digital signature. Ideally the imported public ppg key should be signed with your own private key, but that step is it not critical and is included in this example. Not doing so causes the warning about the key not being certified, but we it does allow us to authenticate the download.

Action: Verify the PGP Digital Signatures

Execute the following commands to verify the PGP digital signature of the Apache source archive:

```
$ gpg --verify httpd-2.2.3.tar.gz.asc
gpg: Signature made Thu 27 Jul 2006 01:35:49 PM EDT using RSA key ID 10FDE075
gpg: Can't check signature: public key not found
#### Note: Copy the key id to be received from the default ppg key server.
$ gpg --recv-key 10FDE075

$ gpg --verify httpd-2.2.3.tar.gz.asc
gpg: Signature made Thu 27 Jul 2006 01:35:49 PM EDT using RSA key ID 10FDE075
gpg: Good signature from "wrowe@covalent.net"
```

```

gpg:          aka "wrowe@lnd.com"
gpg:          aka "wrowe@apache.org"
gpg:          aka "William A. Rowe, Jr. <wrowe@rowe-clan.net>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 33 16 9B 46 FC 12 D4 01 CA 6D DB D7 DE EA 4F D7

```

Verify MD5 Checksum Description

MD5 is a one-way hash function, meaning that it takes a message and converts it into a fixed string of digits, also called a message digest. In essence, MD5 is a way to verify data integrity, and is not as reliable as a PGP signature as it doesn't authenticate, but it is much more reliable than a simple checksum.

If you run MD5 against the Apache source you downloaded and the resulting hash number does not match the MD5 check file, then you should download the entire distribution again. The following can cause these mismatches: looking at wrong MD5 file, modified source code and errors in the downloading process. The GNU Privacy Guard is the opensource implementation of PGP available on most Unix/Linux platforms is available from <http://gnupg.org/> The md5 or md5sum programs are also included in many Unix/Linux distributions. It is also available as part of GNU Textutils <http://www.gnu.org/software/textutils/textutils.html>.

Action: Verify the MD5 Checksums

Execute the following commands to verify the MD5 checksum of the Apache source archive. The final command piped to diff should have no output.

```

$ cat httpd-2.2.3.tar.gz.md5
f72ffb176e2dc7b322be16508c09f63c httpd-2.2.3.tar.gz
$ md5sum httpd-2.2.3.tar.gz
f72ffb176e2dc7b322be16508c09f63c httpd-2.2.3.tar.gz
$ md5sum httpd-2.2.3.tar.gz | diff httpd-2.2.3.tar.gz.md5 -

```

3. Configure the Apache Software

Description

Often times, web administrators compile/enable Apache modules which are unnecessary for the proper functioning of their website. This is similar to the OS security issue of running unnecessary network services; such as Telnet and FTP. By compiling and enabling these unused modules, you are potentially providing additional avenues of attack against your web server. You should only enable the modules that you absolutely need for the functionality of your web site. By removing certain modules, not only will you increase security, but you will also increase the resulting speed of your Apache binary. If you are not sure which modules you need, read about them at the following location – <http://httpd.apache.org/docs/2.0/mod/>

Execute the following command to untar and view the configure default options for the Apache compilation. In particular we are interested in enabling and disabling modules. for the resulting Apache httpd binary:

```

$ cd some/build/directory/for/apache
$ tar xzf httpd-2.2.3.tar.gz
$ cd httpd-2.2.3
# ./configure -help
-- CUT --
Optional Features:
  --disable-FEATURE      do not include FEATURE

```

```
--enable-FEATURE[=ARG]  include FEATURE [ARG=yes]

--enable-modules=MODULE-LIST
                        Space-separated list of modules to enable | "all" |
                        "most"
--enable-mods-shared=MODULE-LIST
                        Space-separated list of shared modules to enable |
                        "all" | "most"

-- CUT --
```

We will be changing some of the default parameters from above to specify which modules that we want to compile into the httpd executable that we are creating. It is a good idea to confirm your sites need for each module. Just because you “enable” these modules and compile then into the Apache binary, these modules are not actually used until you specify some parameters within the httpd.conf file.

Refer to [Appendix A](#) for a closer look at the Apache modules list and the rationale for either enabling or disabling the module.

Action: Configure the Apache Software

Execute the following commands but with the appropriate modules enabled and disabled for your requirements:

```
$ ./configure --prefix=/usr/local \
--enable-ssl \
--enable-headers \
--enable-rewrite \
--enable-auth_digest \
--enable-vhost_alias \
--disable-autoindex \
--disable-status \
--disable-imap \
--disable-userdir
```

Remember, when issuing this command, we are only specifying the changes that we want from the default configuration. For those modules that we did not specify on the command line, it will use the default.

IMPORTANT- Prior to compiling the Apache source code, you should review the Level II sections for the following security modules and decide if you want to use them:

- Mod_Dosevasive
- Mod_Security

If you are planning on implementing these modules, you should use the appropriate syntax in the Level II sections for compiling Apache with these added modules.

[Back to Top ^](#)

4. Compile and Install the Apache Software

Description

This step will compile the Apache source and create the resulting httpd binary file. Common problems associated with compiling the source code are having an incorrect compiler and having an incorrect

PATH variable set. If you run into problems, refer to the INSTALL document in the Apache source directory.

By running the “make install” command, you will be copying all of the newly compiled software, files and directories into the locations that were specified either within the config.layout file or with the path you specified on the command line. After successfully installing executing this step, it is possible to start up the apache web server. We will not do this at this point however, since we will be changing many configurations to tighten security.

Action: Compile Apache

Run the following command to compile the Apache httpd binary:

```
$ make
-- CUT --
/usr/local/apache2/build/libtool --silent --mode=link gcc -g -O2 -pthread
-L/usr/local/apache2/lib -o httpd modules.lo buildmark.o -export-dynamic
server/libmain.la modules/aaa/libmod_authn_file.la
modules/aaa/libmod_authn_default.la modules/aaa/libmod_authz_host.la
modules/aaa/libmod_authz_groupfile.la modules/aaa/libmod_authz_user.la
modules/aaa/libmod_authz_default.la modules/aaa/libmod_auth_basic.la
modules/aaa/libmod_auth_digest.la modules/filters/libmod_include.la
modules/filters/libmod_filter.la modules/loggers/libmod_log_config.la
modules/metadata/libmod_env.la modules/metadata/libmod_headers.la
modules/metadata/libmod_setenvif.la modules/ssl/libmod_ssl.la
modules/http/libmod_http.la modules/http/libmod_mime.la
modules/generators/libmod_asis.la modules/generators/libmod_cgi.la
modules/mappers/libmod_vhost_alias.la modules/mappers/libmod_negotiation.la
modules/mappers/libmod_dir.la modules/mappers/libmod_actions.la
modules/mappers/libmod_alias.la modules/mappers/libmod_rewrite.la
modules/mappers/libmod_so.la server/mpm/prefork/libprefork.la
os/unix/libos.la -lm /home/ralf/download/httpd-2.2.3/srclib/pcre/libpcre.la
/usr/local/apache2/lib/libaprutil-1.la -lexpat /usr/local/apache2/lib/libapr-
1.la -lrt -lcrypt -lpthread -ldl
make[1]: Leaving directory `/home/ralf/httpd-2.2.3'
```

Execute the following command as superuser to install the new software:

```
$ sudo make install
Making install in srclib
make[1]: Entering directory `/home/ralf/httpd-2.2.3/srclib'
-- CUT --
Installing configuration files
[PRESERVING EXISTING HTDOCS SUBDIR: /usr/local/apache2/htdocs]
[PRESERVING EXISTING ERROR SUBDIR: /usr/local/apache2/error]
[PRESERVING EXISTING ICONS SUBDIR: /usr/local/apache2/icons]
[PRESERVING EXISTING CGI SUBDIR: /usr/local/apache2/cgi-bin]
Installing header files
Installing build system files
Installing man pages and online manual
make[1]: Leaving directory `/home/ralf/httpd-2.2.3'
```

Note: Users of other Unix-like operating systems including Mandrake Linux, HP-UX, and Sun Solaris should consult their OS documentation for proper use of compilation commands.

Appendix C – References

1. Barnett, Ryan. "Securing Apache: Step by Step." SANS GIAC GCUX Practical, May 31, 2001. http://www.giac.org/practical/ryan_barnett_gcux.zip
2. Barnett, Ryan. "Preventing Web Site Defacements." SANS Information Reading Room, June 13, 2001. <http://rr.sans.org/securitybasics/deface.php>
3. The Open Web Application Security Project. "A Guide To Building Secure Web Applications", September 22, 2002. <http://www.cgisecurity.com/owasp/html/index.html>
4. Rivest, Ron. "The MD5 Message-Digest Algorithm". April 1992. <http://www.ietf.org/rfc/rfc1321.txt>
5. Network Dweebs. "Apache DoS Evasive Maneuvers Module", April 25, 2003. http://www.networkdweebs.com/stuff/mod_dosevasive.tar.gz
6. RedHat Inc. "Securing and Optimizing Linux". 2002 <http://www.tldp.org/LDP/solrhe/Securing-Optimizing-Linux-RH-Edition-v1.3/chap29sec254.html>
7. Stein, Lincoln. "The World Wide Web Security FAQ" V. 2.0.1 13 September, 1999. <http://www.perl.com/CPAN-local/doc/FAQs/cgi/www-security-faq.html>
8. Vandeburg, Paul D. J. & Wyess, Susan D. "Securing Solaris Servers - A Checklist Approach." USENIX, <http://www.usenix.org/sage/sysadmins/solaris/index.html>
9. Netscape Corporation, "Secure Socket Layer", 2000. <http://wp.netscape.com/security/techbriefs/ssl.html>
10. Intersect Alliance, "2.0 Apache Base Installation". Intersect Alliance, April 15, 2002. <http://www.intersectalliance.com/projects/ApacheConfig/index.html>
11. Eeye Security. "SecureIIS Application Firewall". <http://www.sbnetworksecurity.com/SecureIIS.htm>
12. Lee, Dustin. & Rowe, Jeff. & Levitt, Karl. & Ko, Calvin. "Detecting and Defending against Web-Server Fingerprinting". December 9, 2002. <http://acsac.org/2002/papers/96.pdf>
13. Hollander, Yona. "The Future of Web Server Security". Enterscept Security Technologies. <http://www.enterscept.com/products/enterscept/whitepapers/wpfuture.asp>
14. Bobbit, Mike. "Web Security: Bulletproof". InfoSecurity Magazine, May 2002. <http://infosecuritymag.techtarget.com/2002/may/bulletproof.shtml>
15. Ristic, Ivan. "Mod_Security: Reference Manual" July 10, 2003. <http://www.modsecurity.org/documentation/>
16. Zeno. "Fingerprinting Port80 Attacks". Ovember 2001 <http://www.cgisecurity.com/papers/fingerprint-port80.txt>

[Back to Top ^](#)

Appendix D – Red Hat Linux References

These are items that reference specific Red Hat Linux commands or procedures. Similar actions for Solaris are currently in development.

- L1.2 Install Apache Web Server
- L1.3 Create Web Groups
- L1.4 Create the Apache Web User Account
- L1.7 Apply Current Patches
- L1.12 Mod_Security
- L1.20 Implementing Secure Socket Layer (SSL) with Mod_SSL
- Appendix B: Build Apache from Source

Revision History**Original Version 1.0 May 2004 -- Editor Ryan Barnett****Fedora Core 5 & Updates Version 1.3 Aug 2006 -- Editor Ralph Durkee**

- General -- Updated commands and examples to used Fedora Core 5
- General -- Applied custom styles consistently at least to the item headers, and generated TOC.
- General -- Defined Code style and changed colors of examples for better contrast.
- L1 1 Harden the Underlying Operating System - Updated references for Unix CIS benchmarks
- L1 2 Install Apache Web Server - Install Apache Web Server (Simplified)
- L1 3-4 Create the Web Groups and user - Used apache user & group for consistency
- L1 5 Lock Down the Apache Web User Account - Used /dev/null and Apache user
- L1 6 Subscribe to the Appropriate Security Advisories - New item separated to be actionable.
- L1 7 Apply Current Patches - Added usage of yum
- L1 6-8 Build from source is no longer recommend, moved detailed build instructions moved to appendix B.
- L1 9 Disable Unnecessary Apache Modules - New / replacement for build from source
- L1 10 Denial of Service (DoS) Protective General Directives - Dropped Start/Min/Max directives, not needed for Level 1 security, Corrections and minor fixups for recommended value.
- L1 old11 Server oriented Directives - Item was dropped since there was only one recommended setting in this group of directives, the ServerType, which is specific to Apache 1.3 and is more of a performance issue than a security issue.
- L1 11 Web Server Software Obfuscation General Directives - Updates settings to reflect Apache 2.x , added examples responses
- L1 old15 Web Server Fingerprinting -- Move to L2
- L1 12 Intrusion Detection Options - minor changes
- L1 13 Mod Security - Added Linux installation, Added concrete recommended settings.
- L1 14 Access Control Directives - Minor improvements
- L1 18 Logging General Directives - Added log rotation, and discussion for multiple web sites or organizations.
- L1 16 Remove Default/Unneeded Apache Files - Updated examples for Fedora Core
- L1 20 Updating Ownership and Permissions for Enhanced Security - Updated examples for Fedora Core
- L1 old25 Update apachectl for email notification - moved to level 2

- L2 1 Create Web User Account Disk Quota - updated commands and exampls for Fedora Core 5
- L2 3 -- Original Script is not included in the download bundle, probably should consider a different approach or drop the item.

- L2 5 LogFormat Advanced Settings - Dropped as there was no real benefit above the combined format.
- L2 8 Implementing Secure Socket Layer (SSL) with Mod_SSL - This is a pretty easy item to do these days, and has been moved to level 1. Added more details on generating certificates, and emphasised the importance of a signed certificate when appropriate.
- L2 9 Buffer Overflow Protections - Some inconsistencies corrected in the recommended values.
- L2 10 URL Inspection with Mod_Rewrite - only minor changes
- L2 11 Deny HTTP TRACE Requests with Mod_Rewrite - New item separated from previous so as to be scored. This item is easy to do, and shows up on even the most basic vulnerability scans, so it has been moved to level 1.
- L2 13 Web Server Fingerprinting - moved to Level II from Level I

Updates Version 1.4 Oct 2006 -- Editor Ralph Durkee

- Introduction - Updated links and added reference to Web Application Security Consortium along with OWASP.
- L1 5 Lock Down the Apache Web User Account - Changed username to apache.
- L1 9 Disable Unnecessary Apache Modules - Added recommendation to disable module and test functionality.
- L1 12 IDS - Robots.txt and Fake CGI - dropped, use mod_security instead.

Updates Version 1.5 Oct 2006 -- Editor Ralph Durkee

- L1 3 & 4 L1 3. Create the Web Groups and Apache User Account - Explain system gids and uids, and cross reference the consistency with the RedHat BM.

Updates Version 1.6 Nov 2006 -- Editor Ralph Durkee

- L1 7. Apply Current Patches - Updated patching for source builds
- L1 12. Mod_Security - Updated mod_security compilation from source
- L1 19 Remove Default/Unneeded Apache Files - Updated remove src files.
- L1 20 Implementing SSL - Updated for building mod_ssl from source.
- Appendix B -- Build Apache From Source - Redo using up-to-date Fedora Core 5 examples.