

# University of Piraeus

## Department of Digital Systems



### THESIS

#### A Practical Approach for Web Application Security

Panagiotis Kritikos

Postgraduate Programme in "Techno-economic Management  
and Digital Systems Security"

Associate Professor  
Xenakis Christos

May 2016



## Table of Contents

<b>ABSTRACT .....</b>	<b>6</b>
<b>1. INTRODUCTION .....</b>	<b>6</b>
<b>2. LINUX OPERATING SYSTEM.....</b>	<b>6</b>
<b>3. CENTOS 7 INSTALLATION .....</b>	<b>7</b>
<b>4. CENTOS 7 CONFIGURATION .....</b>	<b>8</b>
4.1 CONFIGURE NETWORK WITH STATIC IP ADDRESS .....	8
4.2 SET HOST AND HOSTNAME OF SERVER .....	10
4.3 UPDATE OR UPGRADE CENTOS MINIMAL INSTALL.....	10
4.4 INSTALL COMMAND LINE WEB BROWSER .....	11
4.5 INSTALL AND CONFIGURE SSH SERVER .....	12
4.6 INSTALL GCC (GNU COMPILER COLLECTION) .....	12
4.7 INSTALL JAVA.....	13
4.8 INSTALL APACHE TOMCAT.....	13
4.9 INSTALL NMAP TO MONITOR OPEN PORTS.....	16
4.10 FIREWALLD CONFIGURATION.....	16
4.11 INSTALLING WGET .....	19
4.12 INSTALLING WEBMIN .....	20
4.13 ENABLE THIRD PARTY REPOSITORIES .....	20
4.14 INSTALL 7-ZIP UTILITY.....	20
4.15 INSTALL AND ENABLE SELINUX .....	21
4.16 INSTALL LINUX MALWARE DETECT (LMD).....	21
<b>5. INSTALLING LAMP (LINUX, APACHE, MARIADB, PHP/PHPMYADMIN) IN CENTOS 7.0 .....</b>	<b>22</b>
5.1 INSTALL APACHE HTTP SERVER .....	22
5.2 INSTALL PHP.....	24
5.3 INSTALL MARIADB DATABASE .....	25
5.4 INSTALL PHPMYADMIN .....	27
<b>6. HOW TO INSTALL MOD_SECURITY WITH APACHE ON CENTOS 7 .....</b>	<b>29</b>
6.1 INSTALLATION AND CONFIGURATION OF MOD_SECURITY & MOD_EVASIVE .....	30
<b>7. RISK ASSESSMENT &amp; MITIGATION .....</b>	<b>33</b>
7.1 INSTALLATION OF WEB APPLICATION DVWA –DAMN VULNERABLE WEB APPLICATION- .....	33
7.2 SPLUNK SECURITY INSTALLATION .....	36
7.3 NETWORK & WEB APPLICATION SCAN .....	38
7.3.1 Mod_Security Audit Logs .....	38
7.3.2 Mod_Security Audit Log Example .....	41
7.3.3 Web Application Scan with Acunetix Vulnerability Scanner .....	46
7.3.4 Web Application Scan with Nessus Vulnerability Scanner .....	48
7.4 IDENTIFICATION AND MITIGATION OF WEB ATTACKS.....	51
7.4.1 Handling Mod_Security Rules and False Positives .....	51
7.4.2 SQL Injection Attacks .....	55
7.4.3 Cross-site Scripting (XSS) Attacks .....	57
7.4.4 File Inclusion Attacks.....	60
7.4.5 Cross-Site Request Forgery (CSRF) Attacks .....	63
7.4.6 Total Attacks .....	65
<b>8. CONCLUSIONS .....</b>	<b>66</b>
<b>9. REFERENCES .....</b>	<b>68</b>

## Table of Images

IMAGE 4.1 SETTING THE HOST FILE WITH OUR STATIC IP ADDRESS .....	10
IMAGE 4.2 JAVA INSTALLATION .....	13
IMAGE 4.3 TOMCAT INSTALLATION .....	15
IMAGE 4.4 SETTING NEW USER FOR TOMCAT .....	15
IMAGE 5.1 PHP INSTALLATION .....	25
IMAGE 5.2 MARIADB INSTALLATION.....	26
IMAGE 5.3 PHPMYADMIN CONFIGURATION A.....	28
IMAGE 5.4 PHPMYADMIN LOGIN.....	28
IMAGE 5.5 PHPMYADMIN CONFIGURATION B.....	29
IMAGE 6.1 INSTALLATION OF MOD_SECURITY AND MOD_EVASIVE.....	31
IMAGE 7.1 DVWA INTERFACE .....	35
IMAGE 7.2 SPLUNK SECURITY .....	37
IMAGE 7.3 LOGS EXAMPLE OF SQLi ATTACK .....	41
IMAGE 7.4 ACUNETIX WEB SCAN 1 .....	47
IMAGE 7.5 ACUNETIX WEB SCAN 2 .....	47
IMAGE 7.6 MONITORING LOGS WITH SPLUNK SECURITY .....	48
IMAGE 7.7 TEMPLATES OF NESSUS SCANS.....	49
IMAGE 7.8 BASIC NETWORK SCAN.....	49
IMAGE 7.9 WEB APPLICATION SCAN .....	50
IMAGE 7.10 TOP VALUES OF HTTP STATUS CODES PER TIME.....	50
IMAGE 7.11 INCLUDE CUSTOM RULES WITH APACHE INCLUDE DIRECTIVE .....	53
IMAGE 7.12 CREATED NEW CUSTOM RULES FILES.....	53
IMAGE 7.13 LOGS FROM SPLUNK OF SQL INJECTION ATTACKS.....	55
IMAGE 7.14 TOP URI FROM THE SQL INJECTION ATTACKS.....	56
IMAGE 7.15 SUCCESS RATE OF SQL INJECTION ATTACKS BEFORE AND AFTER OUR WAF IMPLEMENTATION .....	56
IMAGE 7.16 CUSTOM FILE FOR RULES DETECTING SQL INJECTION ATTACKS.....	57
IMAGE 7.17 XSS ATTACKS OVERVIEW BEFORE WAF IMPLEMENTATION .....	58
IMAGE 7.18 XSS ATTACKS OVERVIEW AFTER WAF IMPLEMENTATION .....	58
IMAGE 7.19 LOGS FROM SPLUNK OF XSS ATTACKS.....	59
IMAGE 7.20 CUSTOM FILE FOR RULES DETECTING XSS ATTACKS.....	59
IMAGE 7.21 LOGS FROM SPLUNK OF RFI ATTACKS .....	61
IMAGE 7.22 LFI ATTACKS OVERVIEW BEFORE AND AFTER WAF IMPLEMENTATION .....	61
IMAGE 7.23 CUSTOM FILE FOR RULES DETECTING LFI ATTACKS .....	62
IMAGE 7.24 CUSTOM FILE FOR RULES DETECTING RFI ATTACKS.....	62
IMAGE 7.25 CSRF ATTACKS OVERVIEW BEFORE AND AFTER WAF IMPLEMENTATION.....	63
IMAGE 7.26 LOGS FROM SPLUNK OF CSRF ATTACKS .....	64
IMAGE 7.27 CUSTOM FILE FOR RULES DETECTING CSRF ATTACKS.....	64
IMAGE 7.28 TOTAL ATTACKS OVERVIEW BEFORE AND AFTER WAF IMPLEMENTATION.....	65

## Table of Tables

TABLE 5.1 WEB SERVER DEVELOPERS: MARKET SHARE OF ACTIVE SITES [3].....	23
TABLE 7.1 AUDIT LOG DIRECTIVES [8].....	39
TABLE 7.2 AUDIT LOG PARTS [8] .....	40
TABLE 7.3 WEB APPLICATION SECURITY CONSORTIUM .....	45
TABLE 7.4 OWASP.....	45
TABLE 7.5 PCI STANDARDS.....	46



# Abstract

Protecting Web Applications is quite challenging. Both web applications and web server platforms that run them, are a big source of security vulnerabilities. Policy based confinement and conventional access control policies, firewalls as well as intrusion detection and prevention systems are effective in detecting a majority of attacks. However, they are unable to detect attacks that “hijack” access to web applications. This paper presents a practical approach to achieve security goals, to eliminate common security exploits, to identify various threats and to secure the important and ubiquitous Web Applications.

## 1.Introduction

In this paper, we are demonstrating how we can fortify and at what extent our web servers and web applications by using the Mod\_Security Web Application Firewall (WAF) module and the Splunk Enterprise Security solution for monitoring, analyzing and managing security information and events.

For the purposes of this paper, we decided to use a Linux Operating System (OS) and more specifically the CentOS version 7, Linux distribution. In chapters 2, 3 and 4, we first explain the reasons why we selected to deploy a Linux OS and amongst those the CentOS distribution as well as how to install it and configure it properly up to some degree.

In chapters 5 and 6, we describe how we deployed and configured our web server with all the essential modules and Mod\_Security WAF module in CentOS 7 respectively. Furthermore, in chapter 7 we install our web application and Splunk Security, perform a risk assessment on our web application by using vulnerability scanners tools and exhibit how we can identify and mitigate some well-known web attacks. Finally, in chapter 8 we summarize with our findings and conclusions.

## 2.Linux Operating System

Linux is a computer operating system (OS) quite similar to the Unix OS and it is mostly POSIX-compliant assembled under the model of free and open-source software development and distribution. Linux was originally created to be similar to Unix. Both have similar tools for interfacing with the systems, programming tools, file system layouts, and other key components. However, Unix is not free. Over the years, a number of different operating systems have been created that attempted to be “unix-like” or “unix-compatible,” but Linux has been the most successful, far surpassing its predecessors in popularity.

Linux is the best-known and most-used open source operating system. It lies underneath all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware. For the purposes of this paper, we use the term "Linux" to refer to the Linux kernel, but also the set of programs, tools, and services that are typically bundled together with the Linux kernel to provide all of the necessary components of a fully functional operating system.

In many ways Linux is similar to other operating systems such as Windows, OS X, or iOS. Like other operating systems, Linux has also a graphical interface, and types of software we are accustomed to using on other operating systems, such as word processing applications. Most likely, a Linux version of the same program you use on other systems also exists. If you can use a computer or other electronic device, you can use Linux.

Nevertheless, Linux also is different from other operating systems in many important ways. Primarily, Linux is open source software; the code used to create Linux is free and available to the public to view, edit, and, for users with the appropriate skills, to contribute to it.

Finally, Linux is also different in that, although the core pieces of the Linux OS are generally common, there are many distributions of Linux, which include different software options. This means that Linux is incredibly customizable, because Linux users can swap not just applications out but they can also choose core components, such as which system displays graphics they prefer, and other user-interface components. That is the reason why we chose to use Linux for our purposes.

### **3. CentOS 7 Installation**

Amongst a wide variety of Linux distributions, we chose for our project to be based on the CentOS Linux operating system. Since, it offers a consistent manageable platform that suits a wide variety of deployments, as well as a solid, predictable base to build upon, along with extensive resources to build, test, release, and maintain open source code.

The CentOS Project [\[1\]](#) is a community-driven free Linux software distribution focused on delivering a robust open source ecosystem. It is a stable, predictable, manageable and reproducible platform derived from the sources of Red Hat Enterprise Linux (RHEL).

In this tutorial, we downloaded and installed the minimal ISO image, CentOS-7-x86\_64-Minimal-1511.iso, from the official site (<https://www.centos.org/download/>). Also, the CentOS Project is modeled on the structure of the Apache Foundation and will match with our Apache web server installation.

Web servers are used to serve Web pages requested by client computers. The Apache HTTP Server, informally called Apache, is the world's most used web server software and more specifically the most commonly used Web server on Linux systems. Originally, it was based on the NCSA HTTPd server.

The Apache HTTP Server Project [\[2\]](#) is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards.

## 4. CentOS 7 Configuration

### 4.1 Configure Network with Static IP Address

The first thing needs to be done is to configure Static IP address, Route and DNS to the CentOS Server. This can be done by either the ip command or the ifconfig command. The ifconfig command is still available for most of the Linux distributions and can be installed from default repository.

```
# yum install net-tools
```

So, to configure static IP address, make sure you first check the current IP address.

```
# ip addr show
```

or

```
# ifconfig
```

Now open and edit file /etc/sysconfig/network-scripts/ifcfg-eno167777736 using your choice of editor. Here, we are using VI editor and of course with root privileges to make changes...

```
# vi /etc/sysconfig/network-scripts/ifcfg-eno167777736
```

Inside the file, we will be editing four fields. Note the below four fields and leave everything else untouched.

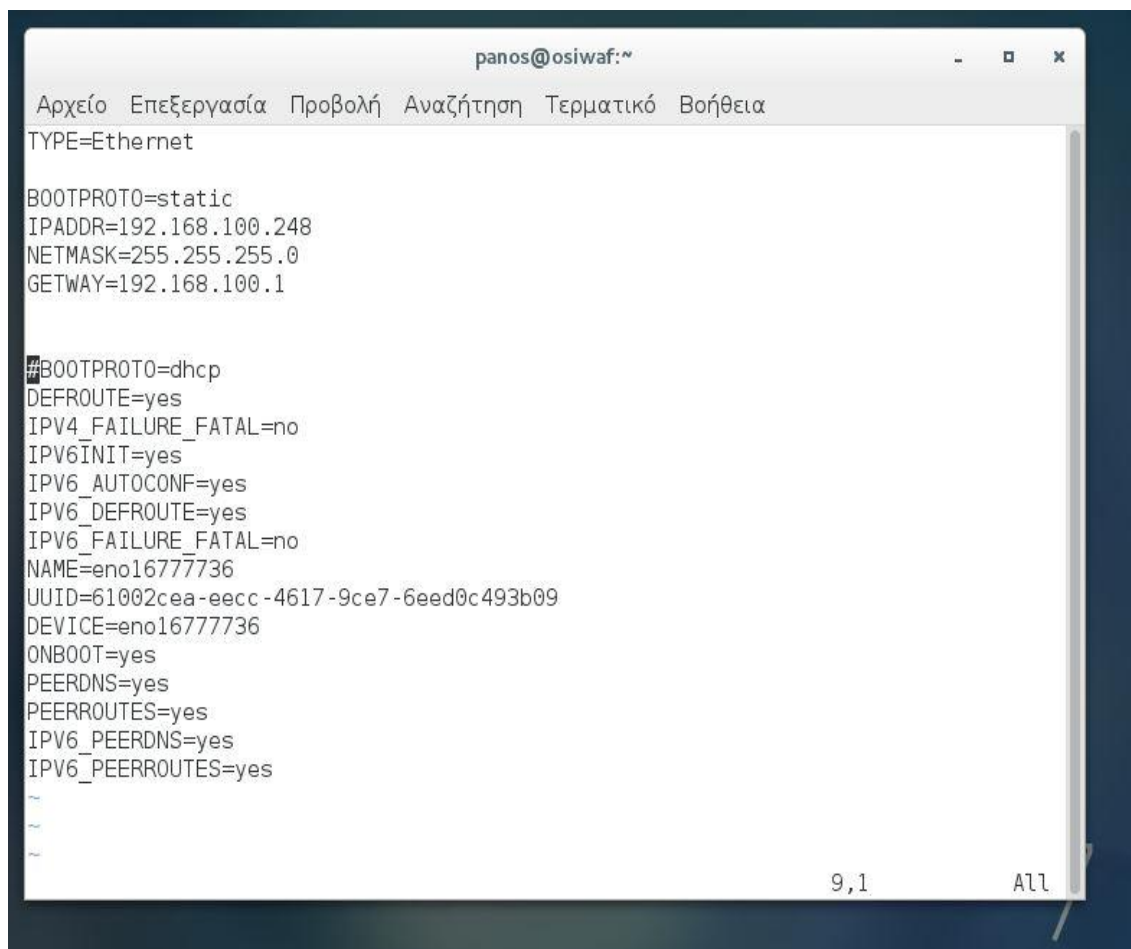
IPADDR = "Enter your static IP here"

GATEWAY = "Enter your Default Gateway"

DNS1 = "Your Domain Name System 1"

After making the changes 'ifcfg-eno167777736', looks something like the image below.



A terminal window titled 'panos@osiwaf:~' with a menu bar in Greek. The terminal displays network configuration for an Ethernet interface. The configuration includes static IP settings (IPADDR, NETMASK, GATEWAY) and DHCP settings (commented out). It also shows IPv6 settings and interface details like NAME, UUID, and DEVICE. At the bottom, there are three tilde characters and a status bar showing '9,1' and 'All'.

```
panos@osiwaf:~  
Αρχείο Επεξεργασία Προβολή Αναζήτηση Τερματικό Βοήθεια  
TYPE=Ethernet  
  
BOOTPROTO=static  
IPADDR=192.168.100.248  
NETMASK=255.255.255.0  
GATEWAY=192.168.100.1  
  
#BOOTPROTO=dhcp  
DEFROUTE=yes  
IPV4_FAILURE_FATAL=no  
IPV6INIT=yes  
IPV6_AUTOCONF=yes  
IPV6_DEFROUTE=yes  
IPV6_FAILURE_FATAL=no  
NAME=enol677736  
UUID=61002cea-eecc-4617-9ce7-6eed0c493b09  
DEVICE=enol677736  
ONBOOT=yes  
PEERDNS=yes  
PEERROUTES=yes  
IPV6_PEERDNS=yes  
IPV6_PEERROUTES=yes  
~  
~  
~  
9,1 All
```

Notice your IP, GATEWAY and DNS will vary, please confirm it with your ISP. Save and Exit.

Restart service network

```
#service network restart
```

After restarting network, make sure to check the IP address and network status...

```
# ip addr show
```

```
# ping -c4 google.com
```

## 4.2 Set Host and Hostname of Server

The next thing to do is to change the Host and Hostname of the CentOS sever. Check the currently assigned HOSTNAME.

```
# echo $HOSTNAME
```

To set new HOSTNAME we need to edit `/etc/hostname` and replace old hostname with the desired one.

```
# vi /etc/hostname
```

After setting hostname, make sure to confirm hostname by logout and login again. To set the host we need to edit `/etc/hosts` file and put our one with the static IP address of our host.

```
# vi /etc/hosts
```

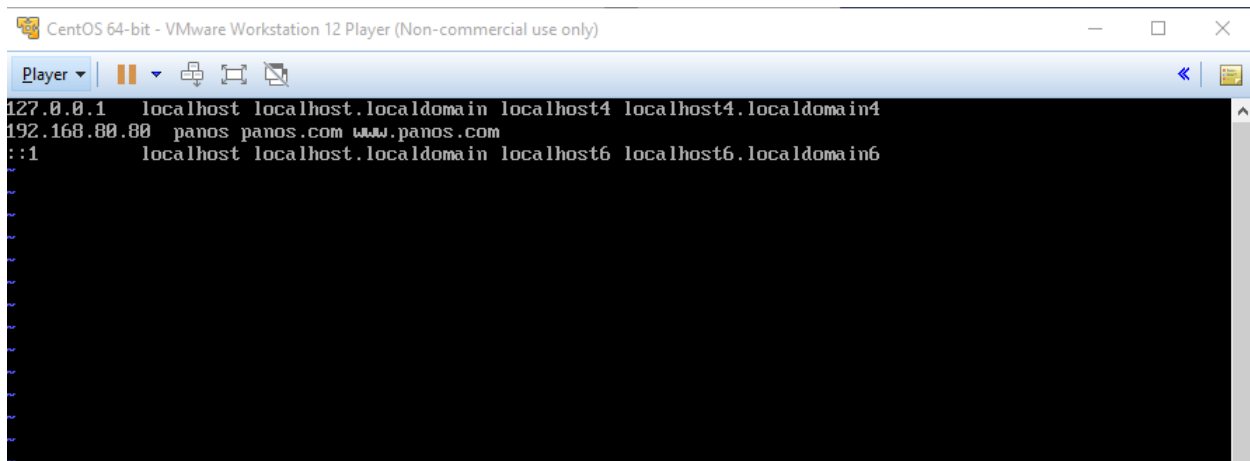


Image 4.1 Setting the host file with our static IP address

## 4.3 Update or Upgrade CentOS Minimal Install

This will not install any new packages other than updating and installing the latest version of installed packages and security updates. Moreover Update and Upgrade are pretty same except the fact that Upgrade = Update + enable obsoletes processing during updates.

```
# yum update && yum upgrade
```

Important: You can also run the below command which will not prompt for the packages update and you do not need to type `y` for accepting the changes. However it

is always a good idea to review the changes which is going to take place on the server specially in production. Hence using the below command may automate the update and upgrade for you but it is not recommended.

```
# yum -y update && yum -y upgrade
```

## 4.4 Install Command Line Web Browser

In most cases, especially in production environment, we usually install CentOS as command line with no GUI; in this situation we must have a command line browsing tool to check websites via terminal. For this, we going to install a most famous tool called 'links'.

```
# yum install links
```

For some people around the globe, a web browser that renders text along with graphics is important since it gives an easy to use and attractive interface, glossy look, nice visibility, easy navigation, and after all click-initiated control. On the other hand there exist some people who want a web browser that render text only.

For System Administrators who generally don't have X-windows as a safety measure on their server, the text based web browser comes to rescue. Some OS comes bundled with the text based browser, viz., the 'links' web comes bundled with Gentoo GNU/Linux where installation proceeds with tar ball.

If a command-line browser is more (speedy, better, interface, etc) then it makes a sense to use such text based browsers. In reality, for some features the text based browser gives better access to encoded information in the page, than the graphical interface.

### Links Browser Properties

- Free and Open source (Foss)
- Text and graphical web browser with a pull down menu.
- Built in support for color and monochrome terminal with the facility of horizontal scrolling.
- Inherits a lot of features from graphical user interface e.g., pop-ups, Menus, etc in textual-fashion.
- Capable of font rendering in different sizes and JavaScript support.

## 4.5 Install and Configure SSH Server

SSH stands for Secure Shell which is the default protocol in Linux for remote management. SSH is one of those essential piece of software which comes default with CentOS Minimal Server.

Check Currently Installed SSH version.

```
# ssh -V
```

Use Secure Protocol over the default SSH Protocol and change port number also for extra Security. Edit the SSH configuration file `/etc/ssh/ssh_config`.

Uncomment the line below line or delete 1 from the Protocol string, so the line seems like:

```
# Protocol 2,1 (Original)
Protocol 2 (Now)
```

This change force SSH to use Protocol 2 which is considered to be more secure than Protocol 1 and also make sure to change the port number 22 to any in the configuration.

Disable SSH `'root login'` and allow to connect to root only after login to normal user account for added additional Security. For this, open and edit configuration file `/etc/ssh/sshd_config` and change `PermitRootLogin yes` to `PermitRootLogin no`.

```
# PermitRootLogin yes (Original)
PermitRootLogin no (Now)
```

Finally, restart SSH service to reflect new changes..

```
# systemctl restart sshd.service
```

## 4.6 Install GCC (GNU Compiler Collection)

GCC stands for GNU Compiler Collection is a compiler system developed by GNU Project that support various programming languages. It is not installed by default in CentOS Minimal Install. To install gcc compiler run the below command.

```
# yum install gcc
```

Check the version of installed gcc.

```
# gcc -version
```

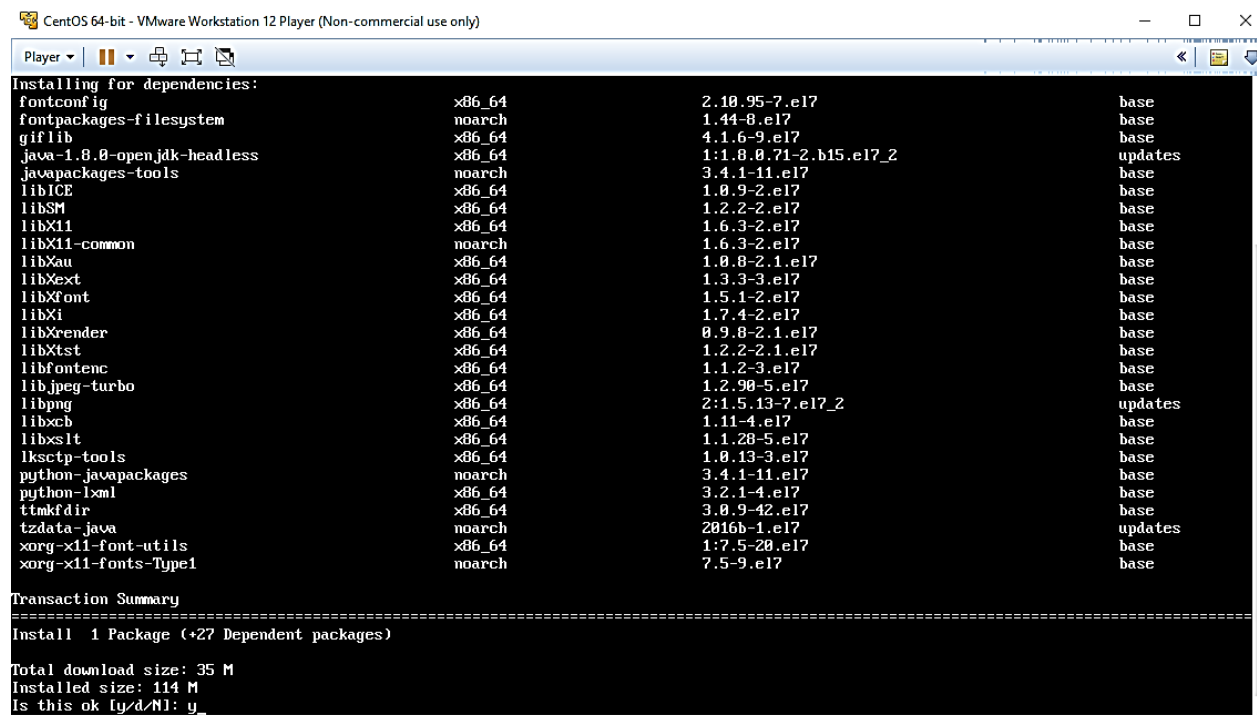
## 4.7 Install Java

Java is a general purpose class based, object-oriented Programming language. It is not installed by default in CentOS Minimal Server. Install Java from repository as below.

```
# yum install java
```

Check version of Java Installed.

```
# java -version
```



```
CentOS 64-bit - VMware Workstation 12 Player (Non-commercial use only)
Installing for dependencies:
fontconfig                x86_64                2.10.95-7.el7                base
fontpackages-filesystem   noarch                1.44-8.el7                   base
giflib                    x86_64                4.1.6-9.el7                   base
java-1.8.0-openjdk-headless x86_64                1:1.8.0.71-2.b15.el7_2       updates
javapackages-tools        noarch                3.4.1-11.el7                  base
libICE                     x86_64                1.0.9-2.el7                   base
libSM                     x86_64                1.2.2-2.el7                   base
libX11                    x86_64                1.6.3-2.el7                   base
libX11-common             noarch                1.6.3-2.el7                   base
libXau                    x86_64                1.0.8-2.1.el7                 base
libXext                   x86_64                1.3.3-3.el7                   base
libXfont                  x86_64                1.5.1-2.el7                   base
libXi                     x86_64                1.7.4-2.el7                   base
libXrender                x86_64                0.9.8-2.1.el7                 base
libXtst                   x86_64                1.2.2-2.1.el7                 base
libfontenc                x86_64                1.1.2-3.el7                   base
libjpeg-turbo             x86_64                1.2.90-5.el7                  base
libpng                    x86_64                2:1.5.13-7.el7_2             updates
libxcb                    x86_64                1.11-4.el7                    base
libxslt                   x86_64                1.1.28-5.el7                  base
lksctp-tools              x86_64                1.0.13-3.el7                  base
python-javapackages        noarch                3.4.1-11.el7                  base
python-lxml               x86_64                3.2.1-4.el7                   base
ttmkfdir                  x86_64                3.0.9-42.el7                  base
tzdata-java               noarch                2016b-1.el7                   updates
xorg-x11-font-utils        x86_64                1:7.5-20.el7                  base
xorg-x11-fonts-Type1       noarch                7.5-9.el7                     base

Transaction Summary
=====
Install 1 Package (+27 Dependent packages)

Total download size: 35 M
Installed size: 114 M
Is this ok [y/d/N]: y_
```

Image 4.2 Java Installation

## 4.8 Install Apache Tomcat

Tomcat is a servlet container designed by Apache to run Java HTTP web server. Install tomcat as below but it is necessary to point out that you must have installed Java prior of installing tomcat.

```
# yum install tomcat
```

After tomcat has been installed, start the tomcat service.

```
# systemctl start tomcat
```

Check Version of tomcat.

```
# /usr/sbin/tomcat version
```

Add service tomcat and default port (8080) through firewall and reload settings.

```
# firewall-cmd --zone=public --add-port=8080/tcp --permanent
# firewall-cmd --reload
```

Now it's time to secure tomcat server, create a user and a password to access and manage. We need to edit file '/etc/tomcat/tomcat-users.xml'. See the section which looks like:

```
<tomcat-users>
```

```
....
```

```
</tomcat-users>
```

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="manager-status"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<user username="tecmin" password="tecmin" roles="manager-gui,manager-
script,manager-jmx,manager-status,admin-gui,admin-script"/>
</tomcat-users>
```

Here we added user "tecmin" to administer/manage tomcat using password "tecmin". Stop and start the service tomcat so that the changes are taken into effect and enable tomcat service to start at system boot.

```
# systemctl stop tomcat
# systemctl start tomcat
# systemctl enable tomcat.service
```

```
CentOS 64-bit - VMware Workstation 12 Player (Non-commercial use only)

Verifying : tomcat-7.0.54-2.el7_1.noarch
Verifying : apache-commons-dbc-1.4-17.el7.noarch
Verifying : tomcat-lib-7.0.54-2.el7_1.noarch
Verifying : xerces-j2-2.11.0-17.el7_0.noarch
Verifying : tomcat-jsp-2.2-api-7.0.54-2.el7_1.noarch
Verifying : geronimo-jta-1.1.1-17.el7.noarch
Verifying : apache-commons-collections-3.2.1-22.el7_2.noarch
Verifying : log4j-1.2.17-15.el7.noarch
Verifying : xml-commons-resolver-1.2-15.el7.noarch

Installed:
tomcat.noarch 0:7.0.54-2.el7_1

Dependency Installed:
apache-commons-collections.noarch 0:3.2.1-22.el7_2
apache-commons-logging.noarch 0:1.1.2-7.el7
avalon-logkit.noarch 0:2.1-14.el7
geronimo-jta.noarch 0:1.1.1-17.el7
tomcat-el-2.2-api.noarch 0:7.0.54-2.el7_1
tomcat-servlet-3.0-api.noarch 0:7.0.54-2.el7_1
xml-commons-apis.noarch 0:1.4.01-16.el7
apache-commons-daemon.x86_64 0:1.0.13-6.el7
apache-commons-pool.noarch 0:1.6-9.el7
ecj.x86_64 1:4.2.1-8.el7
javamail.noarch 0:1.4.6-8.el7
tomcat-jsp-2.2-api.noarch 0:7.0.54-2.el7_1
xalan-j2.noarch 0:2.7.1-23.el7
xml-commons-resolver.noarch 0:1.2-15.el7
apache-commons-dbc.noarch
avalon-framework.noarch 0:4
geronimo-jms.noarch 0:1.1.1
log4j.noarch 0:1.2.17-15.el7
tomcat-lib.noarch 0:7.0.54
xerces-j2.noarch 0:2.11.0-1

Complete!
[root@localhost ~]# systemctl start tomcat
Failed to start tomcat.service: Unit tomcat.service failed to load: No such file or directory.
[root@localhost ~]# /usr/sbin/tomcat version
/usr/sbin/tomcat: line 21: .: /etc/sysconfig/: is a directory
Server version: Apache Tomcat/7.0.54
Server built: May 12 2015 08:07:35
Server number: 7.0.54.0
OS Name: Linux
OS Version: 3.10.0-327.10.1.el7.x86_64
Architecture: amd64
JVM Version: 1.8.0_71-b15
JVM Vendor: Oracle Corporation
[root@localhost ~]#
```

Image 4.3 Tomcat Installation

```
CentOS 64-bit - VMware Workstation 12 Player (Non-commercial use only)

distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<tomcat-users>
<!--
NOTE: By default, no user is included in the "manager-gui" role required
to operate the "/manager/html" web application. If you wish to use this app,
you must define such a user - the username and password are arbitrary.
-->
<!--
NOTE: The sample user and role entries below are wrapped in a comment
and thus are ignored when reading this file. Do not forget to remove
<!-- ... --> that surrounds them.
-->
<!--
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat" password="tomcat" roles="tomcat"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
<user username="role1" password="tomcat" roles="role1"/>
-->
<!-- <role rolename="admin"/> -->
<!-- <role rolename="admin-gui"/> -->
<!-- <role rolename="admin-script"/> -->
<!-- <role rolename="manager"/> -->
<!-- <role rolename="manager-gui"/> -->
<!-- <role rolename="manager-script"/> -->
<!-- <role rolename="manager-jmx"/> -->
<!-- <role rolename="manager-status"/> -->
<user name="panos" password="panos" roles="admin,manager,admin-gui,admin-script,manager-gui,manager-script,manager-jmx,manager-status" />
</tomcat-users>
:~$
```

Image 4.4 Setting new user for Tomcat

## 4.9 Install Nmap to Monitor Open Ports

Nmap for Network Mapper creates a map of the network by discovering host on which it is running as well as by analyzing network. nmap is not included in the default installation and you have to install it from repository.

```
# yum install nmap
```

List all open ports and corresponding services using them on host.

```
# nmap 127.0.0.1
```

You may also use firewall-cmd to list all the ports, however I find nmap more useful.

```
# firewall-cmd --list-ports
```

## 4.10 FirewallD Configuration

Net-filter it is a firewall in Linux. FirewallD is a dynamic daemon to manage firewall with support for networks zones. In RHEL/CentOS 7 and Fedora 21 iptables interface is being replaced by firewallD. FirewallD is installed by default on RedHat Enterprise Linux and its derivatives by default. With iptables every change in order to be taken into effect needs to flush all the old rules and create new rules.

However with firewallD, no flushing and recreating of new rules required and only changes are applied on the fly.

Check if FirewallD is running or not.

```
# systemctl status firewallD
```

OR

```
# firewall-cmd --state
```

Before heading up for firewallD configuration, I would like to discuss about each zones. By default there are some zones available. We need to assign the interface to the zone. A zone define that the zone was trusted or denied level to the interface to get connection. A zone can contain services & ports. Here, we're going describe each zones available in FirewallD.

1. Drop Zone: Any incoming packets are dropped, if we use this drop zone. This is same as we use to add iptables -j drop. If we use the drop rule, means there is no reply, only outgoing network connections will be available.



2. Block Zone: Block zone will deny the incoming network connections are rejected with an icmp-host-prohibited. Only established connections within the server will be allowed.
3. Public Zone: To accept the selected connections we can define rules in public zone. This will only allow the specific port to open in our server other connections will be dropped.
4. External Zone: This zone will act as router options with masquerading is enabled other connections will be dropped and will not accept, only specified connection will be allowed.
5. DMZ Zone: If we need to allow access to some of the services to public, you can define in DMZ zone. This too have the feature of only selected incoming connections are accepted.
6. Work Zone: In this zone, we can define only internal networks i.e. private networks traffic are allowed.
7. Home Zone: This zone is specially used in home areas, we can use this zone to trust the other computers on networks to not harm your computer as every zone. This too allow only the selected incoming connections.
8. Internal Zone: This one is similar to work zone with selected allowed connections.
9. Trusted Zone: If we set the trusted zone all the traffic are accepted.

Now you've better idea about zones, now let's find out some basic commands:

Get a list of all the zones.

```
# firewall-cmd --get-zones
```

To get details on a zone before switching.

```
# firewall-cmd --zone=work --list-all
```

To get default zone.

```
# firewall-cmd --get-default-zone
```

To switch to a different zone say 'work'.

```
# firewall-cmd --set-default-zone=work
```

To list all the services in the zone.

```
# firewall-cmd --list-services
```

To add a service say http, temporarily and reload firewalld.

```
# firewall-cmd --add-service=http  
# firewall-cmd --reload
```

To add a service say http, permanently and reload firewalld.

```
# firewall-cmd --add-service=http --permanent  
# firewall-cmd --reload
```

To remove a service say http, temporarily.

```
# firewall-cmd --remove-service=http  
# firewall-cmd --reload
```

To remove a service say http, permanently.

```
# firewall-cmd --zone=work --remove-service=http --permanent  
# firewall-cmd --reload
```

To allow a port (say 331), temporarily.

```
# firewall-cmd --add-port=331/tcp  
# firewall-cmd --reload
```

To allow a port (say 331), permanently.

```
# firewall-cmd --add-port=331/tcp --permanent  
# firewall-cmd --reload
```

To block/remove a port (say 331), temporarily.

```
# firewall-cmd --remove-port=331/tcp  
# firewall-cmd --reload
```

To block/remove a port (say 331), permanently.

```
# firewall-cmd --remove-port=331/tcp --permanent  
# firewall-cmd --reload
```

If I want to allow the services such as http and https, you use the following rules. First add the rule and make it permanent and reload the rules and check the status.

```
# firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24"
service name="http" accept'
# firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24"
service name="http" accept' --permanent
```

```
# firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24"
service name="https" accept'
# firewall-cmd --add-rich-rule 'rule family="ipv4" source address="192.168.0.0/24"
service name="https" accept' --permanent
```

Now, the Network range 192.168.0.0/24 can use the above service from my server. The option `--permanent` can be used in every rule, but we have to define the rule and check with the client access after that we have to make it permanent.

To disable firewalld.

```
# systemctl stop firewalld
# systemctl disable firewalld
# firewall-cmd --state
```

To enable firewalld.

```
# systemctl enable firewalld
# systemctl start firewalld
# firewall-cmd --state
```

To know more about Firewalld.

```
# man firewalld
```

## 4.11 Installing Wget

wget is a Linux command line based utility that retrieves (downloads) content from web servers. It is an important tool you must have to retrieve web contents or download any files using wget command.

```
# yum install wget
```

For more usage and practical examples on how to use wget command to download files on the terminal, read 10 Wget Command Examples.

## 4.12 Installing Webmin

Webmin is a Web based configuration tool for Linux. It acts as a central system to configure various system configuration like users, disk quota, services and configurations of HTTP server, Apache, MySQL, etc.

```
# wget http://prdownloads.sourceforge.net/webadmin/webmin-1.740-1.noarch.rpm
# rpm -ivh webmin-*.rpm
```

After webmin installation, you will get a message on terminal to login to your host (<http://ip-address:10000>) using your root password on port number 10000. If running a headless server you can forward the port and access it on a machine/server that is headed.

## 4.13 Enable Third Party Repositories

It is not a good idea to add untrusted repositories specially in production and it may be fatal. However just for example here we will be adding a few community approved trusted repositories to install third party tools and packages.

Add Extra Package for Enterprise Linux (EPEL) Repository.

```
# yum install epel-release
```

Add Community Enterprise Linux Repository.

```
# rpm -Uvh http://www.elrepo.org/elrepo-release-7.0-2.el7.elrepo.noarch.rpm
```

## 4.14 Install 7-zip Utility

In the CentOS Minimal Install you don't get utility like **unzip** or **unrar**. We have the option to install each utility as required or an utility that servers for all. **7-zip** is such an utility which compress and extract files of all known types.

```
# yum install p7zip
```

## 4.15 Install and Enable SELinux

**SELinux** which stands for **Security-Enhanced** Linux is a security module at kernel level.

```
# yum install selinux-policy
```

Check SELinux Mode.

```
# getenforce
```

The output is enforcing mode which means SELinux policy is in effect.

For debugging, set selinux mode to **permissive** temporarily. No need to reboot.

```
# setenforce 0
```

After debugging set selinux to **enforcing** again without rebooting.

```
# setenforce 1
```

## 4.16 Install Linux Malware Detect (LMD)

**Linux Malware Detect (LMD)** is an open source Linux malware scanner released under the GNU GPLv2 license, that is specially designed for threats faced in hosting environments.

LMD is not available from online repositories, but is distributed as a tarball from the project's web site. The tarball containing the source code of the latest version is always available at the following link, where it can be downloaded with:

```
# wget http://www.rfxn.com/downloads/maldetect-current.tar.gz
```

Then we need to unpack the tarball and enter the directory where its contents were extracted. There we will find the installation script, install.sh.

```
# tar -xvf maldetect-current.tar.gz
```

```
# ls -l | grep maldetect
```

If we inspect the installation script, which is only 75 lines long (including comments), we will see that it not only installs the tool, but also performs a pre-check to see if the

default installation directory (/usr/local/maldetect) exists. If not, the script creates the installation directory before proceeding.

Finally, after the installation is completed, a daily execution via cron is scheduled by placing the cron.daily script in /etc/cron.daily. This helper script will, among other things, clear old temporary data, check for new LMD releases, and scan the default Apache and web control panels (i.e., CPanel, DirectAdmin, to name a few) default data directories.

That being said, we run the installation script as usual:

```
# ./install.sh
```

You can also install and Use (LMD) with ClamAV as Antivirus Engine

```
# yum install clamd
```

## **5.Installing LAMP (Linux, Apache, MariaDB, PHP/PhpMyAdmin) in CentOS 7.0**

A "LAMP" stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the Linux operating system, with the Apache web server. The site data is stored in a MySQL database (using MariaDB), and dynamic content is processed by PHP.

In this guide, we'll get a LAMP stack installed on the last release of Red Hat Enterprise Linux 7.0 and CentOS 7.0, with the mention that both distributions have upgraded httpd daemon to Apache HTTP 2.4. CentOS will fulfill our first requirement: a Linux operating system.

### **5.1 Install Apache HTTP Server**

No matter for what purpose you will be using the server, in most of the cases you need a HTTP server to run websites, multimedia, client side script and many other things. The Apache web server is currently the most popular web server in the world, which makes it a great default choice for hosting a website.

Developer	March 2016	Percent	April 2016	Percent	Change
Apache	83,825,658	49.16%	82,446,619	49.15%	-0.01
nginx	28,026,677	16.44%	28,196,262	16.81%	0.37
Microsoft	17,228,197	10.10%	16,887,242	10.07%	-0.04
Google	13,545,864	7.94%	12,968,162	7.73%	-0.21

**Table 5.1 Web server developers: Market share of active sites** [\[3\]](#)

We can install Apache easily using CentOS's package manager, `yum`.

```
# yum install httpd
```

If you would like to change default port (80) of Apache HTTP Server to any other port. You need to edit the configuration file `/etc/httpd/conf/httpd.conf` and search for the line that starts typically like:

```
LISTEN 80
```

Change port number '80' to any other port (say 3221), save and exit.  
Add the port you just opened for Apache through firewall and then reload firewall.

Allow service http through firewall (Permanent).

```
# firewall-cmd --add-service=http
```

Allow port 3221 through firewall (Permanent).

```
# firewall-cmd -permanent -add-port=3221/tcp
```

Reload firewall.

```
# firewall-cmd --reload
```

After making all above things, now it's time to restart Apache HTTP server, so that the new port number is taken into effect.

```
# systemctl restart httpd.service
```

Now add the Apache service to system-wide to start automatically when system boots.

```
# systemctl start httpd.service  
# systemctl enable httpd.service
```

Now verify the Apache HTTP Server by using links command line tool as shown in the below screen.

```
# links "your static IP address"
```

## 5.2 Install PHP

PHP is a server-side scripting language for web based services. It is frequently used as general-purpose programming language as well. Install PHP on CentOS Minimal Server as.

```
# yum install php
```

After installing php, make sure to restart Apache service to render PHP in Web Browser.

```
# systemctl restart httpd.service
```

Next, verify PHP by creating following php script in the Apache document root directory.

```
# echo -e "<?php\nphpinfo();\n?>" > /var/www/html/phpinfo.php
```

Now view the PHP file, we just created (phpinfo.php) in Linux Command Line as below.

```
# php /var/www/html/phpinfo.php  
OR  
# links http:// "your static IP address"/phpinfo.php
```



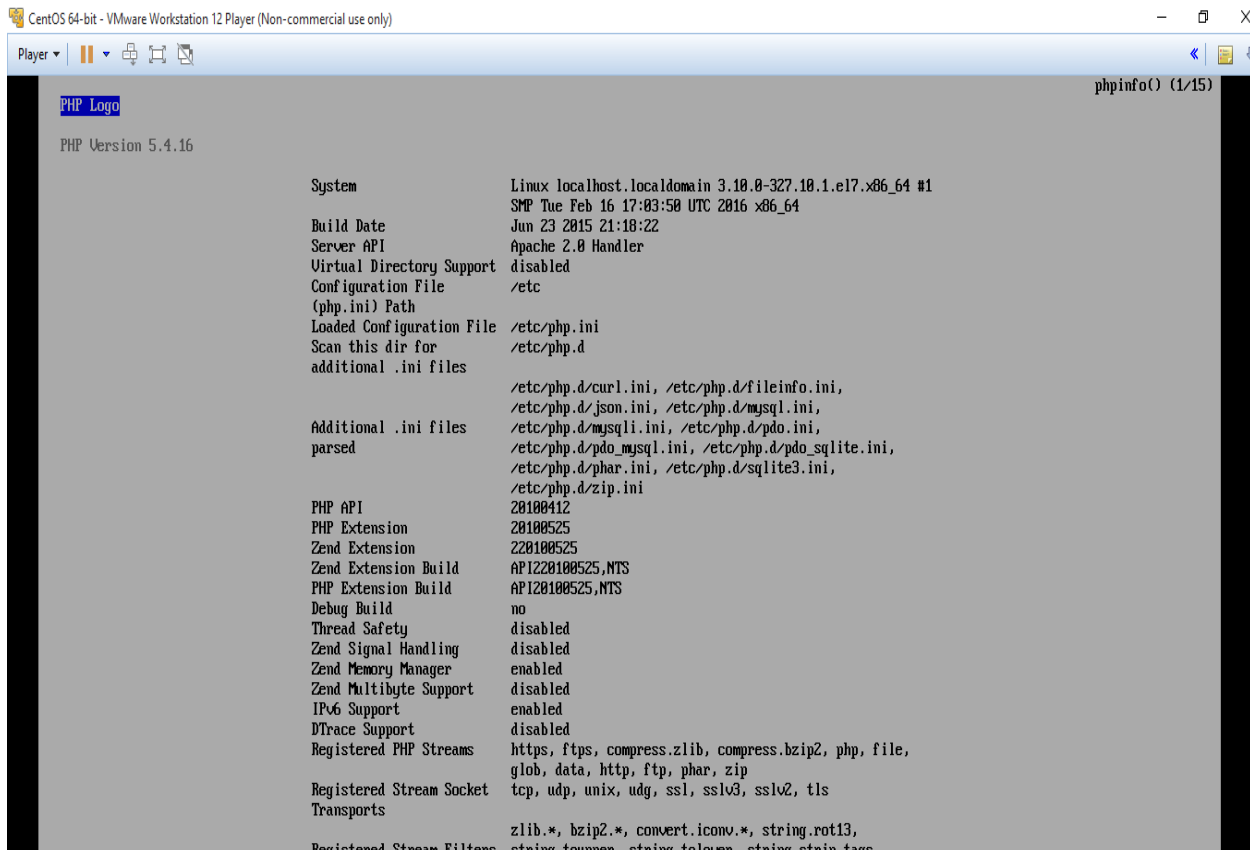


Image 5.1 PHP Installation

## 5.3 Install MariaDB Database

MariaDB is a fork of MySQL. RedHat Enterprise Linux and its derivatives have shifted to MariaDB from MySQL. It is the Primary Database management System. It is again one of those tools which is necessary to have and you will need it sooner or later no matter what kind of server you are setting. Install MariaDB on CentOS Minimal Install server as below.

```
# yum install mariadb-server mariadb
```

Start and configure MariaDB to start automatically at boot.

```
# systemctl start mariadb.service
# systemctl enable mariadb.service
```

Allow service mysql (mariadb) through firewall.

```
# firewall-cmd --add-service=mysql
```

Now it's time to secure MariaDB server.

```
# /usr/bin/mysql_secure_installation
```

To test database functionality login to MariaDB using its root account and exit using quit statement.

```
# mysql -u root -p
# MariaDB > SHOW VARIABLES;
# MariaDB > quit
```

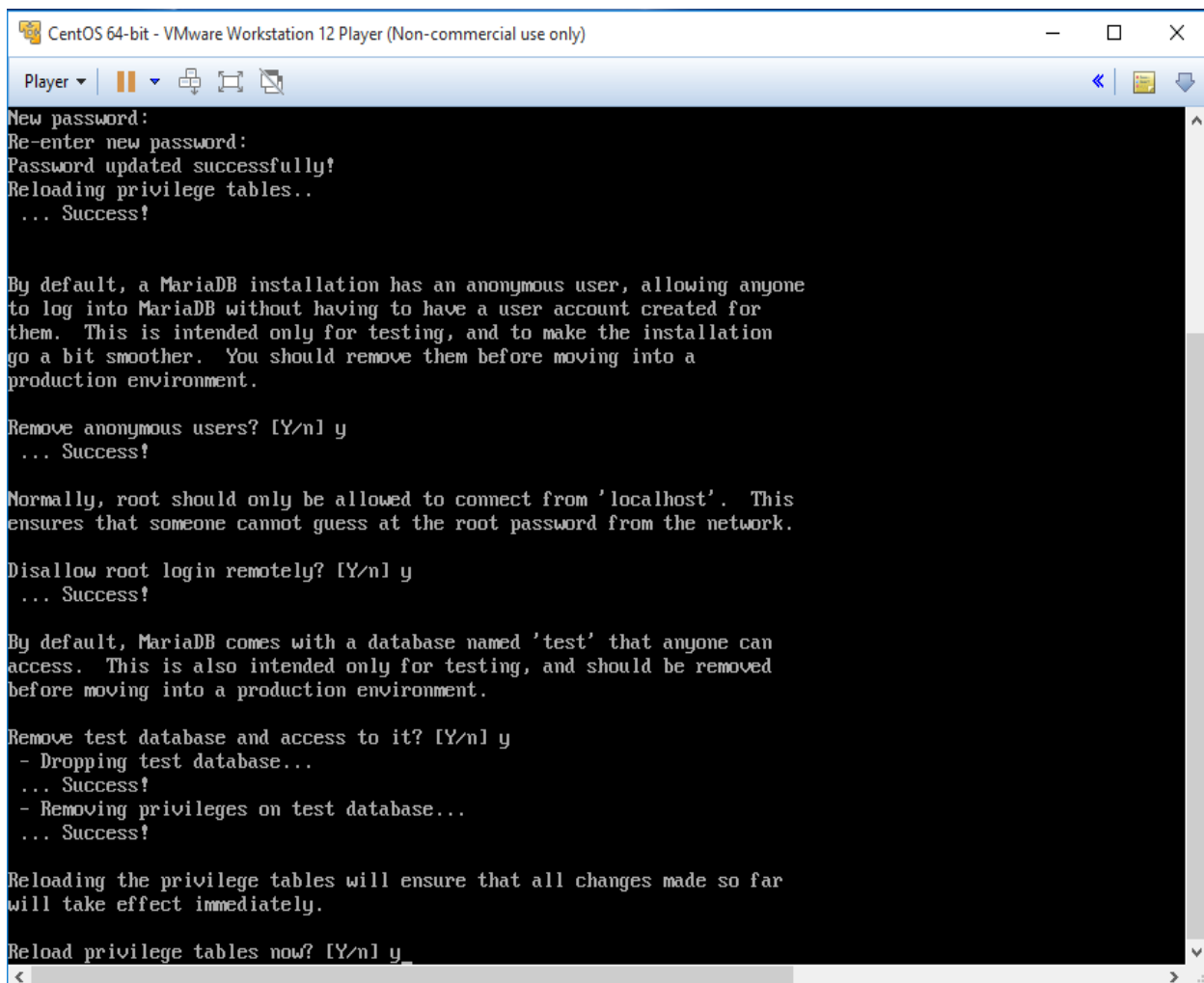


Image 5.2 MariaDB Installation

## 5.4 Install PHPMyAdmin

By default official RHEL 7.0 or CentOS 7.0 repositories doesn't provide any binary package for PhpMyAdmin Web Interface. If you are uncomfortable using MySQL command line to manage your database you can install PhpMyAdmin package by enabling CentOS 7.0 rpmforge repositories using the following command.

```
# yum install http://pkgs.repoforge.org/rpmforge-release/rpmforge-release-0.5.3-1.el7.rf.x86_64.rpm
```

After enabling rpmforge repository, next install PhpMyAdmin.

```
# yum install phpmyadmin
```

Next configure PhpMyAdmin to allow connections from remote hosts by editing `phpmyadmin.conf` file, located on `Apache conf.d` directory, commenting the following lines.

```
# vi /etc/httpd/conf.d/phpmyadmin.conf
```

Use a `#` and comment this lines.

```
# Order Deny,Allow
# Deny from all
# Allow from 127.0.0.1
```

To be able to login to PhpMyAdmin Web interface using cookie authentication method add a blowfish string to `phpmyadmin config.inc.php` file like in the screenshot below using the generate a secret string, restart Apache Web service and direct your browser to the URL address `http://server_IP/phpmyadmin/`.

```
# nano /etc/httpd/conf.d/phpmyadmin.conf
# systemctl restart httpd
```

Enable LAMP System-wide

If you need MariaDB and Apache services to be automatically started after reboot issue the following commands to enable them system-wide.

```
# systemctl enable mariadb
# systemctl enable httpd
```

```
CentOS 64-bit - VMware Workstation 12 Player (Non-commercial use only)

Alias /phpMyAdmin /usr/share/phpMyAdmin
Alias /phpmyadmin /usr/share/phpMyAdmin

<Directory /usr/share/phpMyAdmin/>
    AddDefaultCharset UTF-8

    <IfModule mod_authz_core.c>
        # Apache 2.4
        <RequireAny>
            Require ip 192.168.1.6
            Require ip ::1
        </RequireAny>
    </IfModule>
    <IfModule !mod_authz_core.c>
        # Apache 2.2
        Order Deny,Allow
        Deny from All
        Allow from 192.168.1.6
        Allow from ::1
    </IfModule>
</Directory>

<Directory /usr/share/phpMyAdmin/setup/>
    <IfModule mod_authz_core.c>
        # Apache 2.4
        <RequireAny>
            Require ip 192.168.1.6
            Require ip ::1
        </RequireAny>
    </IfModule>
    <IfModule !mod_authz_core.c>
        # Apache 2.2
        Order Deny,Allow
        Deny from All
        Allow from 192.168.1.6
        Allow from ::1
    </IfModule>
</Directory>
```

Image 5.3 PhpMyAdmin Configuration A

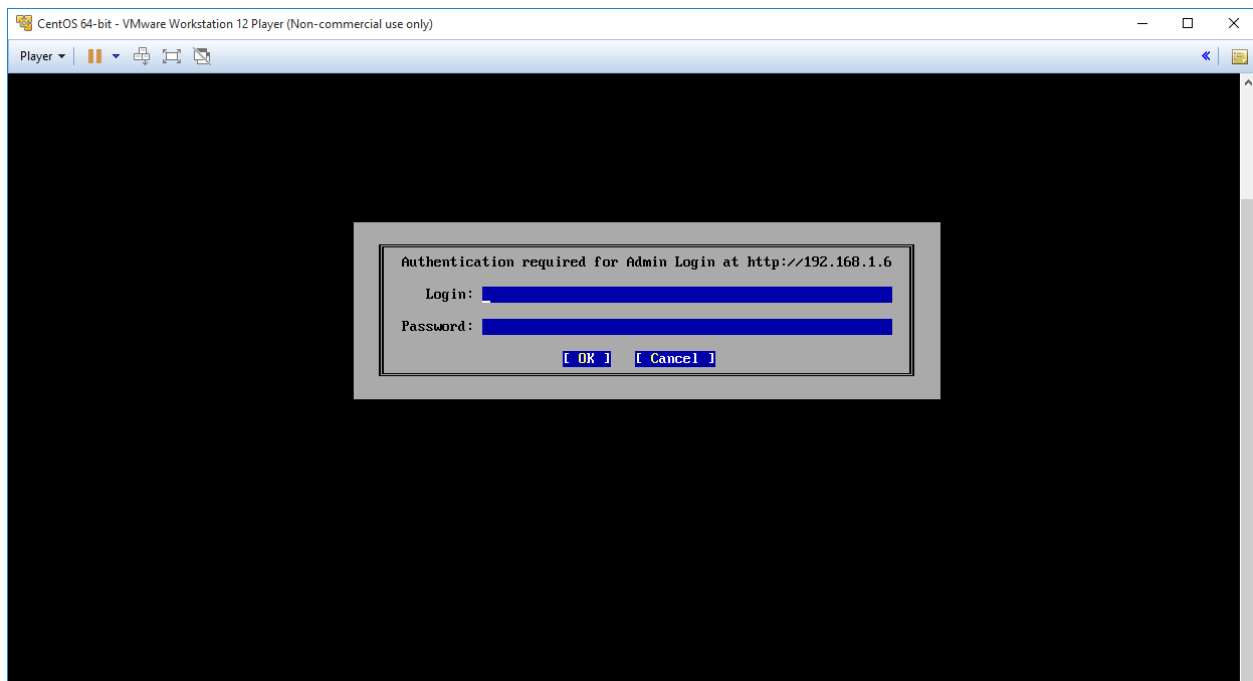


Image 5.4 PhpMyAdmin Login

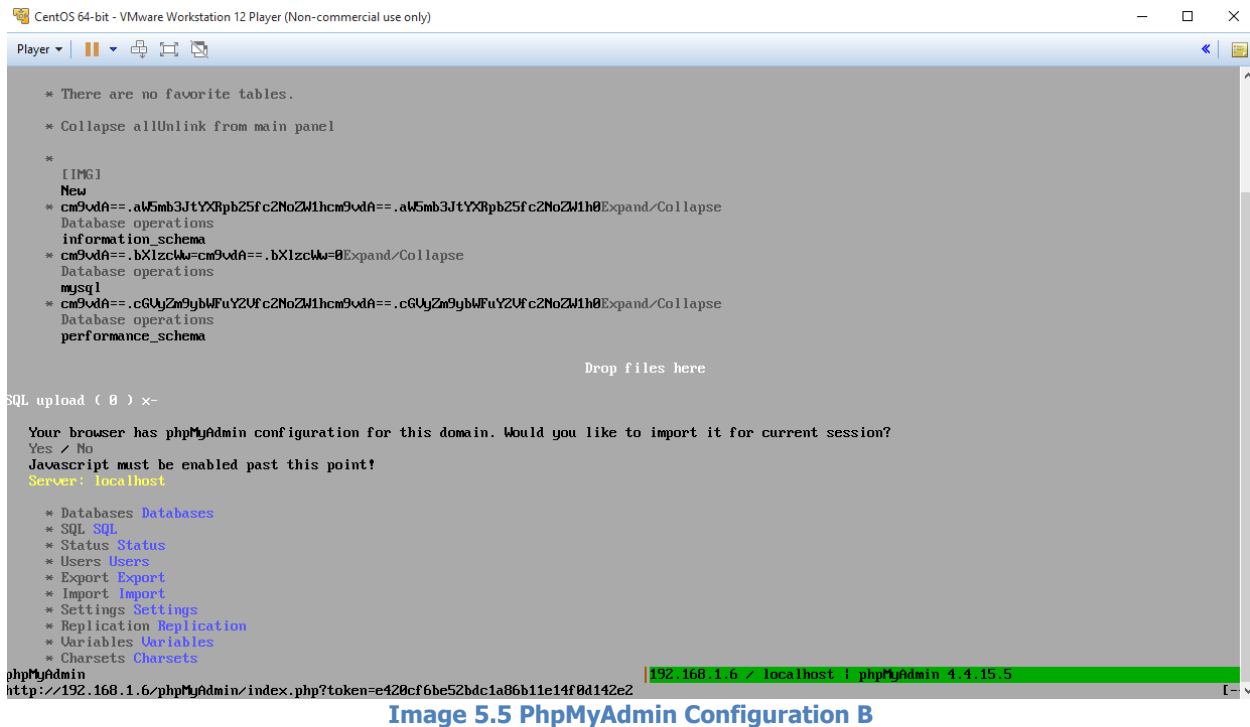


Image 5.5 PhpMyAdmin Configuration B

That is all it takes for a basic LAMP installation on CentOS 7.0.

## 6. How to Install Mod\_Security with Apache on CentOS 7

Mod\_Security is a Web Application Firewall that execute as a Module on your Web Server and provides protection against various attacks to our web applications. It monitors HTTP traffic and performs real time analysis. It's a product developed by Breach Security and is available a free software under the GNU License. It is Available for Apache, Nginx and IIS.

Mod\_Security can be deployed and integrated in our current Web Servers infrastructure, meaning that we do not have to modify our internal Network, we don't add any point of failure, we can benefit from load balancing and scalability and we would not have any issues with compress or encrypted Data. Mod\_Security is a valuable security tool and have proven to be effective. If we want to protect our web applications this is a tool that deserves your attention.

Important files to Remember:

**Mod Security Config File** – /etc/httpd/conf.d/mod\_security.conf  
**Debug Log** – /var/log/httpd/modsec\_debug.log

**Audit log** – /var/log/httpd/modsec\_audit.log  
**Rules** – /etc/httpd/modsecurity.d/activated\_rules

## 6.1 Installation and Configuration of mod\_security & mod\_evasive

It is important to have enabled the EPEL repository in the CentOS/RHEL server before the installation of the following packages,

```
# rpm -ivh http://dl.fedoraproject.org/pub/epel/7/x86\_64/e/epel-release-7-5.noarch.rpm
```

And since we already have, we can proceed:

Log in to the server as user 'root' and make sure that all packages are up to date:

```
# yum -y update
```

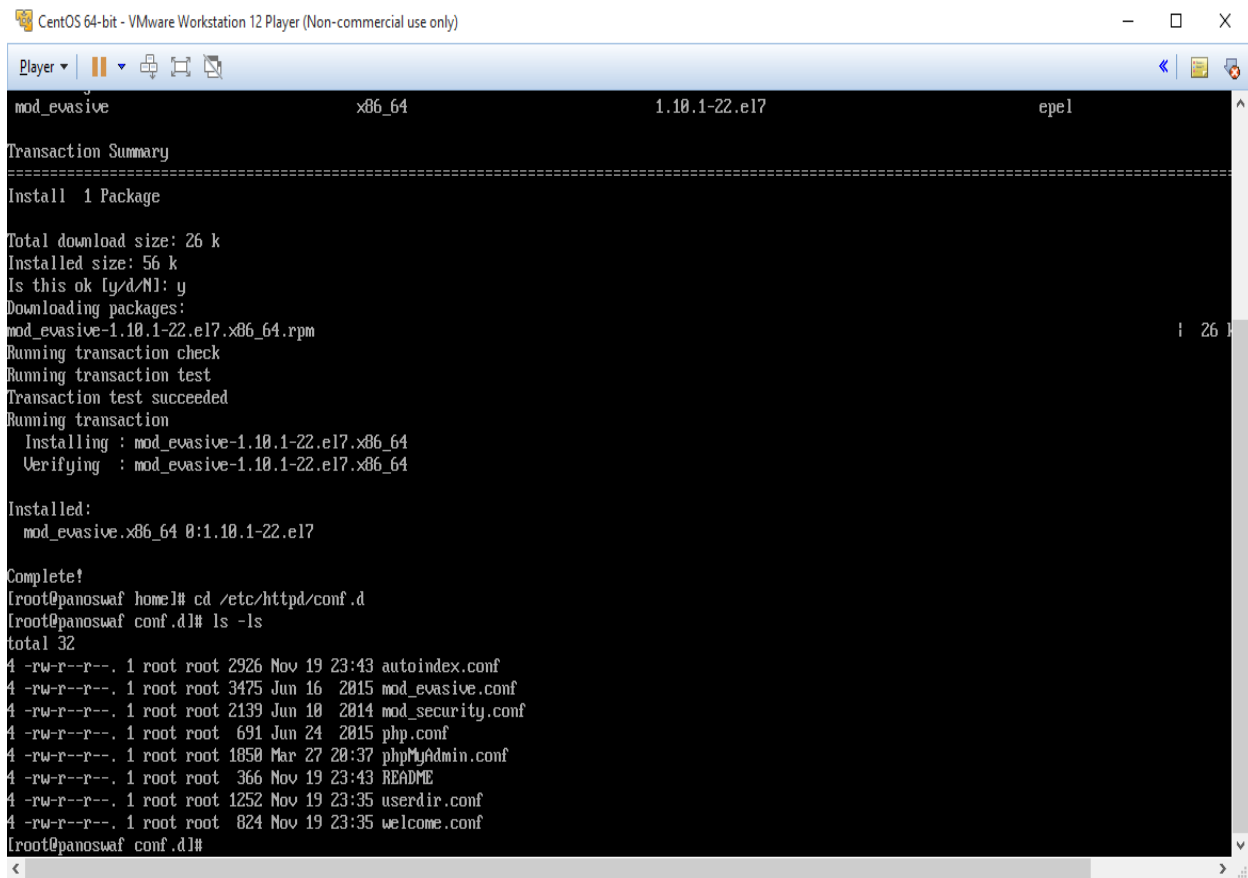
Before we start the installation of mod\_security, we need to install the following dependencies first

```
# yum install gcc make httpd-devel libxml2 pcre-devel libxml2-devel curl-devel git
```

Then install these packages [4]

```
# yum install mod_security  
# yum install mod_evasive
```

After the installation is complete, you will find the main configuration files inside /etc/httpd/conf.d



```
CentOS 64-bit - VMware Workstation 12 Player (Non-commercial use only)
mod_evasive x86_64 1.10.1-22.el7 epel

Transaction Summary
=====
Install 1 Package

Total download size: 26 k
Installed size: 56 k
Is this ok [y/d/N]: y
Downloading packages:
mod_evasive-1.10.1-22.el7.x86_64.rpm | 26 k
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : mod_evasive-1.10.1-22.el7.x86_64
  Verifying  : mod_evasive-1.10.1-22.el7.x86_64

Installed:
  mod_evasive.x86_64 0:1.10.1-22.el7

Complete!
[root@panoswaf home]# cd /etc/httpd/conf.d
[root@panoswaf conf.d]# ls -ls
total 32
4 -rw-r--r--. 1 root root 2926 Nov 19 23:43 autoindex.conf
4 -rw-r--r--. 1 root root 3475 Jun 16 2015 mod_evasive.conf
4 -rw-r--r--. 1 root root 2139 Jun 10 2014 mod_security.conf
4 -rw-r--r--. 1 root root 691 Jun 24 2015 php.conf
4 -rw-r--r--. 1 root root 1850 Mar 27 20:37 phpMyAdmin.conf
4 -rw-r--r--. 1 root root 366 Nov 19 23:43 README
4 -rw-r--r--. 1 root root 1252 Nov 19 23:35 userdir.conf
4 -rw-r--r--. 1 root root 824 Nov 19 23:35 welcome.conf
[root@panoswaf conf.d]#
```

**Image 6.1 Installation of mod\_security and mod\_evasive**

Or download the mod\_security source code (this includes only the mod\_security, not the mod\_evasive module) from their official website to your server

```
# cd /opt/
# wget https://www.modsecurity.org/tarball/2.9.1/modsecurity-2.9.1.tar.gz
```

Extract the downloaded archive and change the current working directory to the newly extracted directory

```
# tar xzfv modsecurity-2.9.1.tar.gz
# cd modsecurity-2.9.1
```

Now, let's configure, compile and install mod\_security from the source code

```
# ./configure
# make
# make install
```

Copy the default `mod_security` configuration and the *unicode.mapping* file to the necessary Apache directory

```
# cp modsecurity.conf-recommended /etc/httpd/conf.d/modsecurity.conf
# cp unicode.mapping /etc/httpd/conf.d/
```

With this step, `mod_security` is installed on your server. Now we need to configure the Apache web server. Open the web server configuration file and add the following line

```
# nano /etc/httpd/conf/httpd.conf
# LoadModule security2_module modules/mod_security2.so
# LoadModule unique_id_module modules/mod_unique_id.so
```

Now you need to make sure that Apache loads both modules when it starts. Look for the following lines (or add them if they are not present) in `mod_security.conf` and `mod_evasive.conf`, respectively:

```
# LoadModule security2_module modules/mod_security2.so
# LoadModule evasive20_module modules/mod_evasive20.so
```

Save the changes and restart Apache

```
# /etc/init.d/httpd restart
```

Download and configure OWASP (Open Web Application Security Project) core rule set for a base configuration

```
# cd /etc/httpd
# git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git
# mv owasp-modsecurity-crs modsecurity-crs
# cd modsecurity-crs
# cp modsecurity_crs_10_setup.conf.example modsecurity_crs_10_config.conf
```

Open the Apache configuration file again, and add the following lines at the end of the file

```
# Include modsecurity-crs/modsecurity_crs_10_config.conf
# Include modsecurity-crs/base_rules/*.conf
```

Save the file and restart the web server again

```
/etc/init.d/httpd restart
```



## 7. Risk Assessment & Mitigation

### 7.1 Installation of Web Application DVWA –Damn Vulnerable Web Application-

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment and help web developers better understand the processes of securing web applications. By using DVWA, you can practice and exploit some of the most common web vulnerabilities, with various difficulty levels, with a simple interface [\[5\]](#).

Simply unzip dvwa.zip, place the unzipped files in your public html folder, then point your browser to:

```
# http://[IP address of your machine]/dvwa/setup.php.
```

The most recent version can be found at <http://www.dvwa.co.uk/>, so we will start by downloading and installing the DVWA into this file,

```
# cd /var/www/html
# wget https://github.com/RandomStorm/DVWA/archive/v1.9.zip
# unzip v1.9.zip
# rm -rf v1.9.zip
```

The configuration directory for DVWA is,

```
# cd /var/www/html/dvwa/config
```

Also, the configuration file for DVWA that handles the database communication from the Web App is,

```
# nano config.inc.php
```

To set up the database, simply click on the Setup DVWA button in the main menu, then click on the Create / Reset Database button. This will create / reset the database for you with some data in.

If you receive an error while trying to create your database, make sure your database credentials are correct within `./config/config.inc.php`.

The variables are set to the following by default:

```
$_DVWA[ 'db_user' ] = 'root';  
$_DVWA[ 'db_password' ] = 'p@ssw0rd';  
$_DVWA[ 'db_database' ] = 'dvwa';
```

Some other configurations may be needed depending on your Operating System as well as your PHP version, so you may wish to alter the default configuration. The location of the files will be different on a per-machine basis. Note that you are unable to use PHP v7.0 or later with DVWA.

Regarding the User Credentials:

Change the default variables and set to them to yours:

```
$_DVWA[ 'db_user' ] = 'root'; - Or your mysql user with root privileges  
$_DVWA[ 'db_password' ] = ' '; - Leave it empty or set it to your mysql password
```

Also, specifically for CentOS 7 version due to some connectivity issues, the following modifications needed to be done:

Open the main database configuration file `/etc/my.conf`:

```
# nano /etc/my.conf
```

Add the following line in order mariadb to accept connections from any address:

```
# bind-address=0.0.0.0
```

And put in comments:

```
# #socket=/var/lib/mysql/mysql.sock
```

Also set the `db_server` IP address within `./config/config.inc.php` to "localhost":

```
$_DVWA[ 'db_server' ] = 'localhost';
```

Regarding the Folder Permissions:

`./hackable/uploads/` - Needs to be writable by the web service (for File Upload).  
`./external/phpids/0.6/lib/IDS/tmp/phpids_log.txt` - Needs to be writable by the web service (if you wish to use PHPIDS).

Regarding PHP configurations:

```
# allow_url_include = on - Allows for Remote File Inclusions (RFI) [allow_url_include]
# allow_url_fopen = on - Allows for Remote File Inclusions (RFI) [allow_url_fopen]
# safe_mode = off - (If PHP <= v5.4) Allows for SQL Injection (SQLi) [safe_mode]
# magic_quotes_gpc = off - (If PHP <= v5.4) Allows for SQL Injection (SQLi)
[magic_quotes_gpc]
# display_errors = off - (Optional) Hides PHP warning messages to make it less verbose
[display_errors]
```

Regarding the ReCaptcha Keys go into the file config/config.inc.php:

`$_DVWA[ 'recaptcha_public_key' ]` & `$_DVWA[ 'recaptcha_private_key' ]` - These values need to be generated from:

# <https://www.google.com/recaptcha/admin/create>

The Default Credentials, which can easily be brute forced, are:

```
# Default username = admin
# Default password = password
```

Finally, Login URL: `http://[IP address of your machine]/dvwa/login.php`

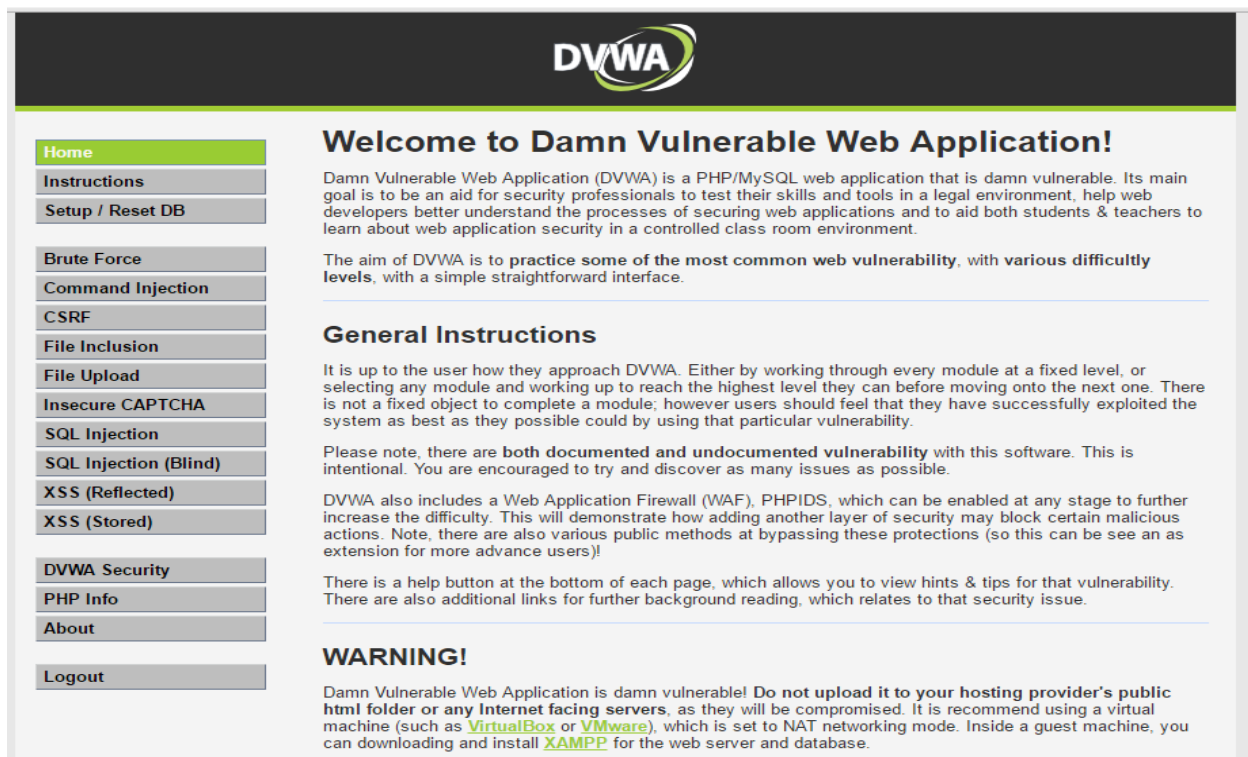


Image 7.1 DVWA Interface

## 7.2 Splunk Security Installation

You can install Splunk Enterprise on Linux using RPM or DEB packages or a tar file [\[6\]](#).

For tar file installation of Splunk Enterprise on a Linux system, expand the tar file into an appropriate directory using the tar command:

```
# tar xvfz splunk_package_name.tgz
```

The default installation directory is /splunk in the current working directory. To install into /opt/splunk, use the following command:

```
# tar xvfz splunk_package_name.tgz -C /opt
```

*Note that when you install Splunk Enterprise with a tar file:*

- *Splunk Enterprise does not create the splunk user. If you want Splunk Enterprise to run as a specific user, you must create the user manually before you install.*
- *Ensure that the disk partition has enough space to hold the uncompressed volume of the data you plan to keep indexed.*

Now for RedHat RPM installation, ensure that the splunk build rpm package you want is available locally on the target server. Verify that the file is readable and executable by the Splunk user. If needed change access:

```
# chmod 744 splunk_package_name.rpm
```

To install the Splunk RPM in the default directory /opt/splunk:

```
# rpm -i splunk_package_name.rpm
```

To install Splunk in a different directory, use the --prefix flag:

```
#rpm -i --prefix=/opt/new_directory splunk_package_name.rpm
```

To enable Splunk Enterprise to start the system at boot by adding it to /etc/init.d/ Run this command as root or sudo and specify the user that Splunk Enterprise should run as.

```
# ./splunk enable boot-start -user splunkuser
```

You can start Splunk Enterprise as any user on the local system. If you run it as a non-root user, make sure that it has the appropriate permissions to read the inputs that you specify. Refer to the instructions for running Splunk Enterprise as a non-root user.

To start Splunk Enterprise from the command-line interface, run the following commands from \$SPLUNK\_HOME/bin directory, where \$SPLUNK\_HOME is the directory into which you installed Splunk Enterprise:

```
# ./splunk start
```

By convention, this document uses:

\$SPLUNK\_HOME to identify the path to your Splunk Enterprise installation.

\$SPLUNK\_HOME/bin/ to indicate the location of the command-line interface.

There are also some Startup options. The first time you start Splunk Enterprise after a new installation, you must accept the license agreement. To start Splunk Enterprise and accept the license in one step:

```
$SPLUNK_HOME/bin/splunk start --accept-license
```

To launch Splunk Web and log in follow these 2 steps below. After you start Splunk Enterprise and accept the license agreement, you can launch Splunk Web:

- In a browser window, access Splunk Web at `http://<hostname>: port`. Hostname is the host machine. Port is the port you specified during the installation (the default port is 8000).
- Splunk Web prompts you for login information before it launches. The default is user name admin and password "changeme". If you switch to Splunk Free, you bypass this logon page in future sessions.

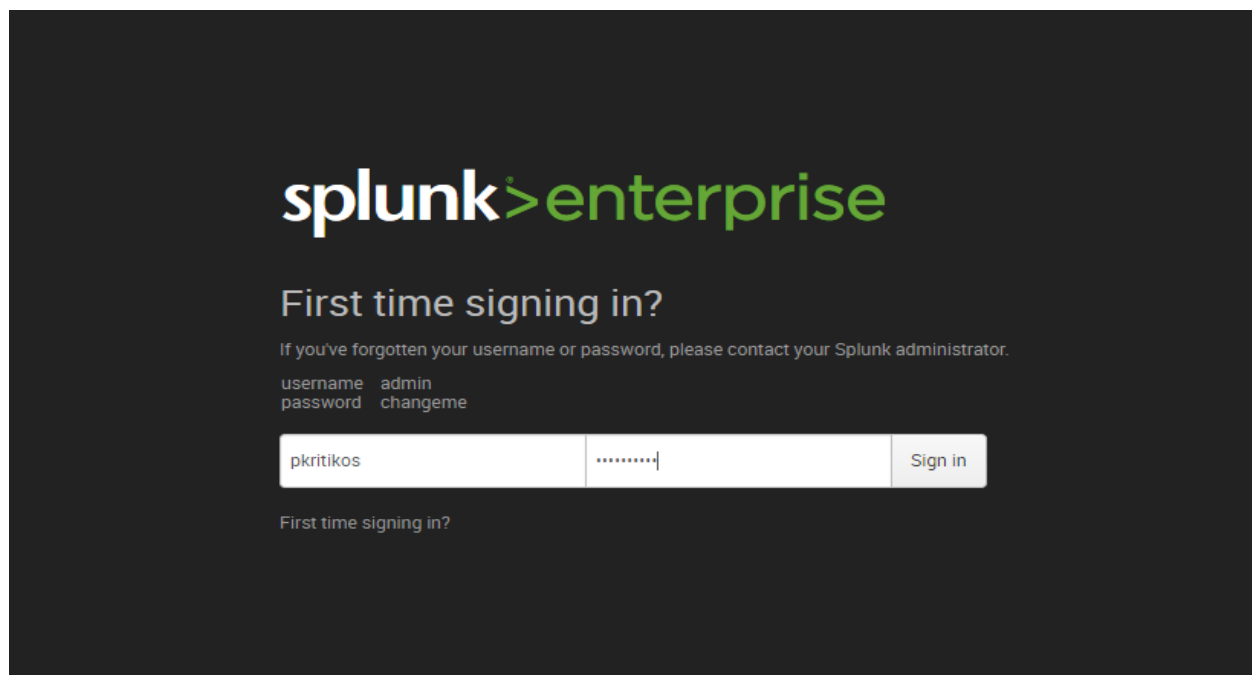


Image 7.2 Splunk Security

## 7.3 Network & Web Application Scan

Before we start scanning our system for vulnerabilities with our vulnerabilities scanners, first we need to have a good understanding of the audit logs, not only from the system and Apache but also those that produced by the Mod\_Security waf module.

### 7.3.1 Mod\_Security Audit Logs

Before we start scanning our system for vulnerabilities, first we need to have a good understanding of the audit logs, not only from the system and Apache but also those that produced by the mod\_security waf module.

Apache uses two kinds of logs: error logs and access logs. Access logs record requests made to the server, while the error log records problems with the server.

On CentOS systems, the location of the error log is specified in `/etc/httpd/conf/httpd.conf` by the directive `ErrorLog logs/error_log`. And, because CentOS is configured so that `/etc/httpd/logs` is a symbolic link to `/var/log/httpd`, the error logs are sent to

```
# /var/log/httpd/error_log.
```

Like syslog messages, Apache generates error messages at different levels: debug, info, notice, warn, error, crit, alert, and emerg. The level recorded in the error log is set by the value of `LogLevel`; all of the discussed distributions set this to warn by default. The access log(s) record requests made of the server. The format of these logs is customizable via the `LogFormat` directive. In its most common use, `LogFormat` takes two arguments: a format string to determine what is logged, and a name for that logging format. On CentOS, in `/etc/httpd/conf/httpd.conf`, we have four common formats: combined, common, referer, and agent with the directives [7]:

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

*Note: The word "referer" is, in fact, misspelled. It was misspelled in the original 1996 RFC for HTTP/1.0, RFC 1945, available at <http://tools.ietf.org/html/rfc1945>, and the new spelling has stuck. It is still in use in the June 2014 RFC 7231 (<http://tools.ietf.org/html/rfc7231>), which notes that referer has been misspelled.*

Components of a format string include the following [7]:

- %b Response size (bytes) not including headers
- %h Name or IP address of the remote host
- %l The reported remote log name (generally just "-")
- %p The port on the server
- %r The first line of the request
- %s The status code returned
- %t Time
- %u The reported remote user name (generally just "-")
- %U The URL path requested
- %v The server name
- %{Referer}i The referer<sup>4</sup> reported by the client
- %{User-Agent}i The user-agent reported by the client

If a format string directive includes ">" like %>s, then whenever the request has been internally redirected, the log entry should contain the final value.

The CustomLog directive takes as arguments file location and a defined log format, then tells Apache to record logs to that file with that format. On CentOS, the primary configuration file /etc/httpd/conf/httpd.conf contains the line

```
# CustomLog logs/access_log combined
```

Thus, the log file /var/log/httpd/access\_log records requests in the combined log format.

The table that follows shows how to capture full HTTP transaction data by using the Mod\_Security audit engine and its directives.

Directive	Description
<b>SecAuditEngine</b>	Controls the audit log engine; possible values On, Off, or RelevantOnly
<b>SecAuditLog</b>	Path to an audit log file
<b>SecAuditLog2</b>	Path to another audit log file (copy)
<b>SecAuditLogParts</b>	Specifies which part of a transaction will be logged
<b>SecAuditLogRelevantStatus</b>	Specifies which response statuses will be considered relevant
<b>SecAuditLogStorageDir</b>	Path where concurrent audit log files will be stored
<b>SecAuditLogType</b>	Specifies the type of audit log to use: Serial or Concurrent

**Table 7.1 Audit log directives [8]**

If you want to provide the greatest amount of data for incident response processes, you should enable full audit logging of both HTTP request and response traffic. However, because requests with bodies will increase the amount of data that needs to be logged, as well as the logging of response bodies, we will not enable full audit logging.

Logically, each audit log entry is a single file. When serial audit logging is used, all entries will be put within one file, but with concurrent audit logging, one file per entry is used. Looking at a single audit log entry, you will find that it consists of multiple independent segments (parts). A segment begins with a boundary and ends when the next segment begins. The only exception is the terminating segment (Z), which consists only of the boundary. The idea behind the use of multiple segments is to allow each audit log entry to contain potentially different information. Only the parts A and Z are mandatory; the use of the other parts is controlled with the SecAuditLogParts directive.

Directive SecAuditLogParts defines the separate transactional elements that are captured. The Table below, "Audit log parts", contains the list of all audit log parts, along with a description of their purpose.

Part letter	Description
<b>A</b>	Audit log header (mandatory)
<b>B</b>	Request headers
<b>C</b>	Request body
<b>D</b>	Reserved
<b>E</b>	Response body
<b>F</b>	Response headers
<b>G</b>	Reserved
<b>H</b>	Audit log trailer, which contains additional data
<b>I</b>	Compact request body alternative (to part C), which excludes files
<b>J</b>	Information on uploaded files (available as of version 2.6.0)
<b>K</b>	Contains a list of all rules that matched for the transaction
<b>Z</b>	Audit log footer - Final boundary (mandatory)

Table 7.2 Audit log parts [\[8\]](#)



### 7.3.2 Mod\_Security Audit Log Example

As an example of typical access log entries, here are the results of protocol violation and sql injection attack, as captured from the security event manager Splunk:



### Image 7.3 Logs example of SQLi attack

Every audit log entry begins with part A, which contains the basic information about the transaction: time, unique ID, source IP address, source port, destination IP address, and destination port:

```
[Fri May 27 01:06:23.084662 2016] [:error] [pid 3408] [client 192.168.1.5]
```

Part B contains the request headers and nothing else:

```
[hostname "192.168.1.6"]
[uri "/DVWA-1.9/index.php"]
[unique_id "V0dzX8IJtrLiws84wCLkrgAAAAU"]
```

Part C contains the raw request body, typically that of a POST request:

```
ModSecurity: Warning. Pattern match
"(?:([|]|'|\"|xc2|xb4|xe2|x80|x99|xe2|x80|x98|(|)(|))]*?)|b([|]|d|w|
++)([|]|'|\"|xc2|xb4|xe2|x80|x99|xe2|x80|x98|(|)(|))]*?)(?:|=|<=>/r?lik
e/sounds|||s+like/regexp)([|]|'|\"|xc2|xb4|xe2|x80|x99|xe2|x80|x98|(|)(|)
)]*?)|2|b(?:!=|<=>=/<>/|/|^/is||s+not ... " at
REQUEST_COOKIES:splunkd_8000. [file
"/etc/httpd/modsecurity.d/sqlcustomrules.conf"]
```



# [line "64"]: The specific line in the aforementioned file.

# [id "27950901"]: The unique id of the triggered rule.

# [rev "2"]: Rule revision, how many times it has changed.

# [msg "SQL Injection Attack: SQL Tautology Detected by PanosWaf"]: Human-readable message, as specified by the msg action

# [data "Matched Data: 1Ae^fhNrIGP8RqD\_u9E7LUdfgaVoecHZZwo16gfUZ found within REQUEST\_COOKIES:splunkd\_8000:1Ae^fhNrIGP8RqD\_u9E7LUdfgaVoecHZZwo16gfUZ^2WjzriB1EZfc92jCzuhSF\_pqrdrFnuWStPoecU02FVKRPUDcz8gsVdM2CvphpYlr\_yilFxPpRmK^gOD47mzXar7FPLtFF"]: The request data that matched with the rule.

# [severity "CRITICAL"]: Event severity as text, as specified by the severity action. The possible values (with their corresponding numerical values in brackets) are EMERGENCY (0), ALERT (1), CRITICAL (2), ERROR (3), WARNING (4), NOTICE (5), INFO (6) and DEBUG (7).

# [ver "OWASP\_CRS/2.2.9"]: The version of the rules.

# [maturity "9"]: How old is the rule.

# [accuracy "8"]: How many false positives it produces. The higher the number the fewer the false positives are. The range is from 1-9.

# [hostname "192.168.1.6"]: The IP address of the target machine.

# [uri "/DVWA-1.9/index.php"]: The request uri.

# [unique\_id "V0dzX8IJtrLiws84wCLkrgAAAAU"]: Unique event ID, generated automatically.

# [tag "OWASP\_CRS/WEB\_ATTACK/SQL\_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP\_TOP\_10/A1"] [tag "OWASP\_AppSensor/CIE1"] [tag "PCI/6.5.2"]: These field tags are referring to the attack categorization by some global accepted organizations. More specifically:

- **Web Application Security Consortium.**

<b>Item</b>	<b>WASC ID</b>
<b>Insufficient Authentication</b>	WASC-01
<b>Insufficient Authorization</b>	WASC-02
<b>Integer Overflows</b>	WASC-03
<b>Insufficient Transport Layer Protection</b>	WASC-04
<b>Remote File Inclusion</b>	WASC-05
<b>Format String</b>	WASC-06
<b>Buffer Overflow</b>	WASC-07
<b>Cross-site Scripting</b>	WASC-08
<b>Cross-site Request Forgery</b>	WASC-09
<b>Denial of Service</b>	WASC-10
<b>Brute Force</b>	WASC-11
<b>Content Spoofing</b>	WASC-12
<b>Information Leakage</b>	WASC-13
<b>Server Misconfiguration</b>	WASC-14
<b>Application Misconfiguration</b>	WASC-15
<b>Directory Indexing</b>	WASC-16
<b>Improper Filesystem Permissions</b>	WASC-17
<b>Credential/Session Prediction</b>	WASC-18
<b>SQL Injection</b>	WASC-19
<b>Improper Input Handling</b>	WASC-20
<b>Insufficient Anti-Automation</b>	WASC-21
<b>Improper Output Handling</b>	WASC-22
<b>XML Injection</b>	WASC-23
<b>HTTP Request Splitting</b>	WASC-24
<b>HTTP Response Splitting</b>	WASC-25
<b>HTTP Request Smuggling</b>	WASC-26
<b>HTTP Response Smuggling</b>	WASC-27
<b>Null Byte Injection</b>	WASC-28
<b>LDAP Injection</b>	WASC-29
<b>Mail Command Injection</b>	WASC-30
<b>OS Commanding</b>	WASC-31
<b>Routing Detour</b>	WASC-32
<b>Path Traversal</b>	WASC-33
<b>Predictable Resource Location</b>	WASC-34
<b>SOAP Array Abuse</b>	WASC-35
<b>SSI Injection</b>	WASC-36
<b>Session Fixation</b>	WASC-37
<b>URI Redirector Abuse</b>	WASC-38
<b>XPath Injection</b>	WASC-39

<b>Insufficient Process Validation</b>	WASC-40
<b>XML Attribute Blowup</b>	WASC-41
<b>Abuse of Functionality</b>	WASC-42
<b>XML External Entities</b>	WASC-43
<b>XML Entity Expansion</b>	WASC-44
<b>Fingerprinting</b>	WASC-45
<b>XQuery Injection</b>	WASC-46
<b>Insufficient Session Expiration</b>	WASC-47
<b>Insecure Indexing</b>	WASC-48
<b>Insufficient Password Recovery</b>	WASC-49

**Table 7.3 Web Application Security Consortium**

- **OWASP**

<b>OWASP Top 10</b>	<b>Testing Requirement</b>
<b>A1</b>	Injection (For example, SQL injection; command injection; CRLF injection; SSI injection; XPath injection)
<b>A2</b>	Broken authentication and session management (For example, weak passwords; cleartext password transmission; session IDs in URL; session fixation)
<b>A3</b>	Cross Site Scripting
<b>A4</b>	Insecure direct object references
<b>A5</b>	Security misconfiguration (For example, software patches; default passwords; error message information leakage)
<b>A6</b>	Sensitive data exposure (For example, unencrypted content; cleartext password transmission; SSL weak algorithms; session cookies without "secure" flag; invalid certificates)
<b>A7</b>	Missing function level access control (For example, direct URL access)
<b>A8</b>	Cross-site request forgery (CSRF)
<b>A9</b>	Using components with known vulnerabilities (For example, vulnerable framework libraries)
<b>A10</b>	Unvalidated redirects and forwards

**Table 7.4 OWASP**

- **PCI Security Standards**

PCI Standard	Testing Requirement
6.5.1	Injection (SQL, LDAP, and Xpath flaws)
6.5.2	Buffer overflows
6.5.3	Insecure cryptographic storage
6.5.4	Insecure communications/transport layer protection
6.5.5	Information leakage and improper error handling
6.5.6	All high vulnerabilities (Reference: PCI Req. 6.2)
6.5.7	Cross site scripting (XSS)
6.5.8	Improper access controls
6.5.9	Cross site request forgery (CSRF)

**Table 7.5 PCI Standards**

### 7.3.3 Web Application Scan with Acunetix Vulnerability Scanner

Acunetix is a powerful and well-known Vulnerability Scanner that automatically crawls and scans off-the-shelf and custom-built websites and web applications for over 3000 web vulnerabilities. It also features its own crawling and scanning engine that fully replicates user interaction inside of a browser by executing and analyzing JavaScript. Acunetix is the industry leader in detecting the largest variety of SQL Injection and XSS vulnerabilities, including Out-of-band SQL Injection and DOM-based XSS. Finally, in order to keep track of the vulnerabilities detected, Acunetix includes various extensive reports, a set of Internal Management reports as well as a range of Compliance and Classification reports for regulatory standards and best practice guidelines to help manage escalation and remediation of vulnerabilities while assisting in task prioritization [9].

The images below illustrate the results of a web vulnerability scanning of DVWA and on how Splunk Security logs the events as well as the fact that Mod\_Security detects it. Acunetix identifies vulnerabilities of different threat levels; for example, the issue that the user credentials are sent in clear text which is a threat level Medium (2).

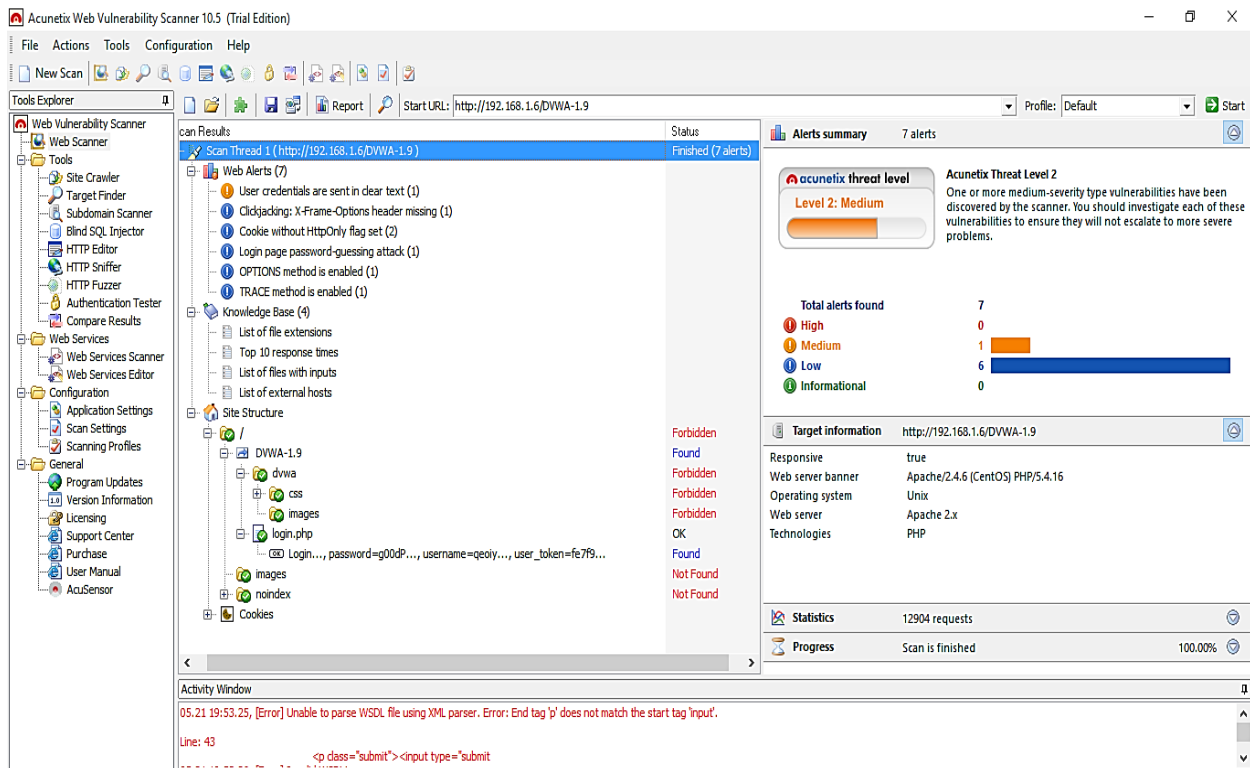


Image 7.4 Acunetix Web Scan 1

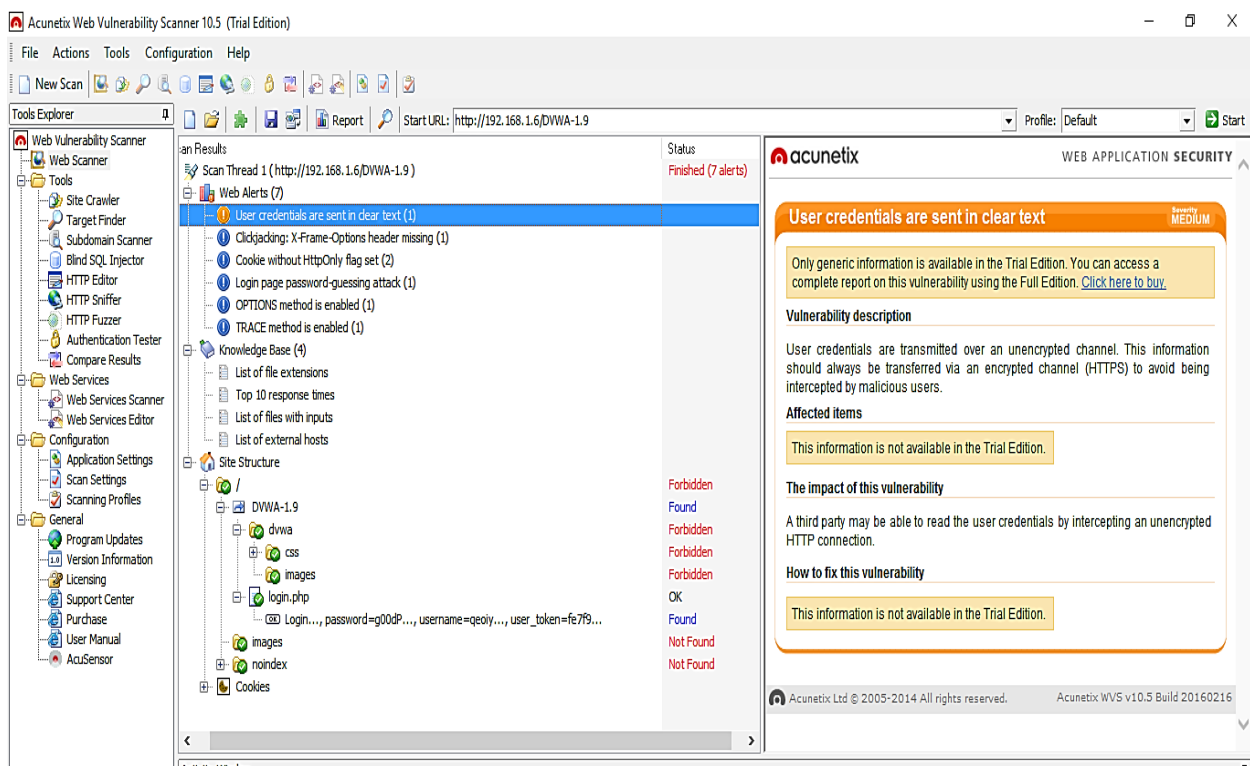
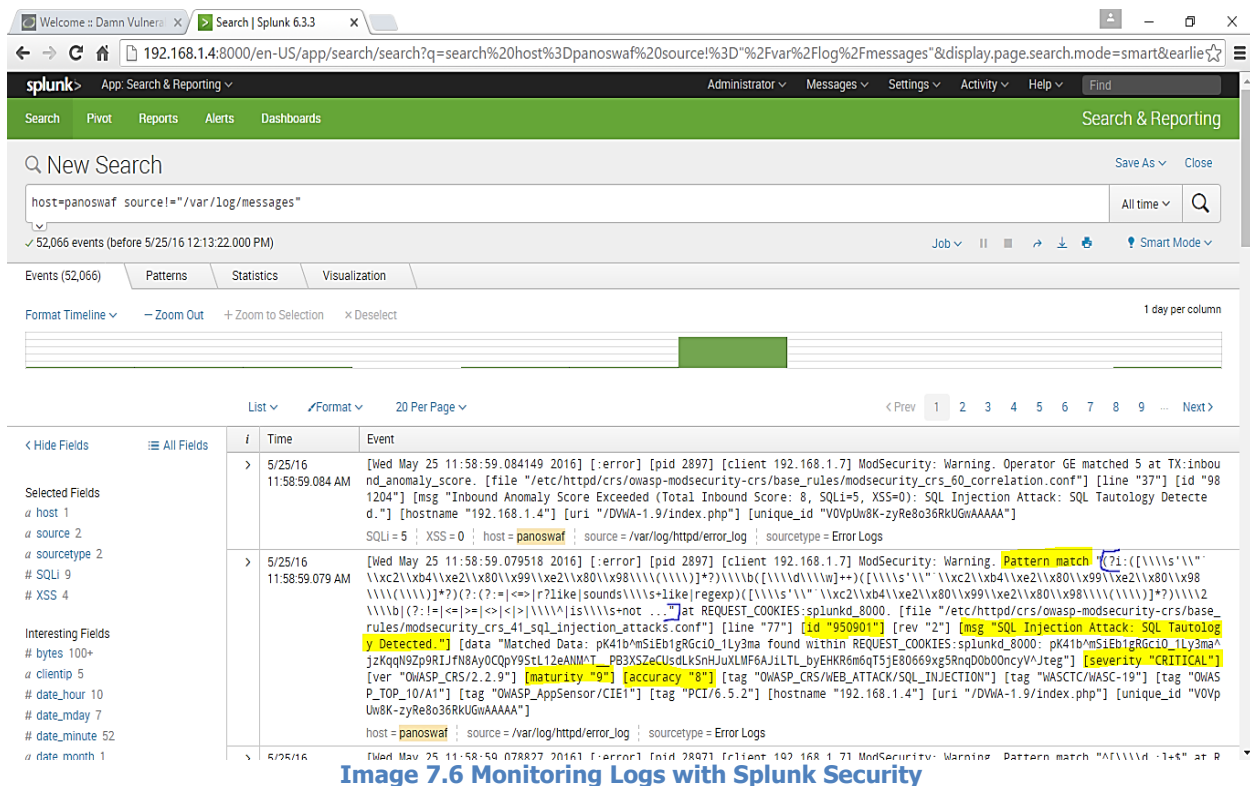


Image 7.5 Acunetix Web Scan 2





### 7.3.4 Web Application Scan with Nessus Vulnerability Scanner

Nessus has been the most widely deployed solution for vulnerability, configuration and compliance assessments. Nessus prevents network attacks by identifying the vulnerabilities and configuration issues that hackers use to penetrate your network. It supports the widest range of network devices, operating systems, databases, applications in physical, virtual and cloud infrastructures.

Nessus also supports non-credentialed, remote scans; credentialed, local scans for deeper, granular analysis of assets; and offline auditing on a network device's configuration. Regarding threat detection, Nessus scans for viruses, malware, backdoors, hosts communicating with botnet-infected systems, known/unknown processes as well as web services linking to malicious content. Lastly, it creates reports with exploitability, severity modification and scan scheduling [10].

The images below illustrate the different scanning options provided by Nessus, the results of a basic network and a web application scanning of DVWA. Using Splunk, we create a timely visualization of the logs that parsed according to HTTP status codes. Nessus, in both scans, identifies vulnerabilities of different threat levels; for example, vulnerabilities in MySQL database, SSH protocol, HTTP type and version, etc.



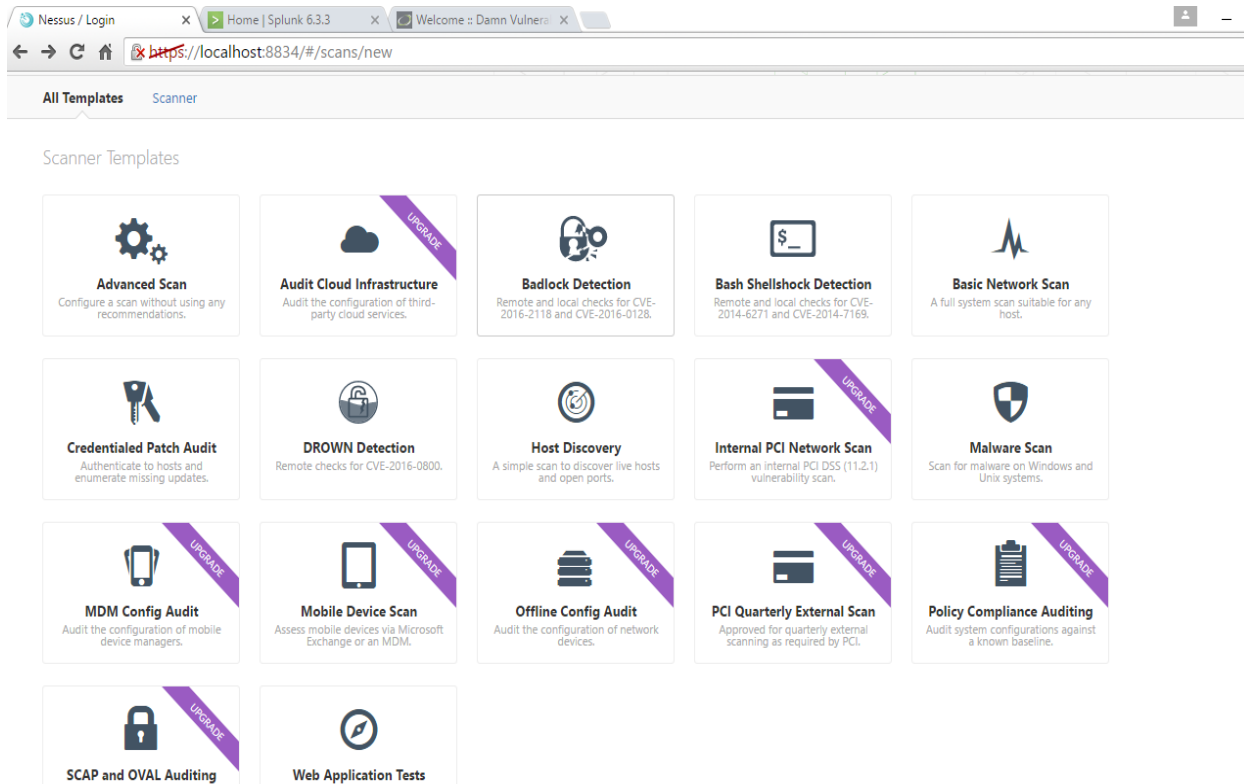


Image 7.7 Templates of Nessus Scans

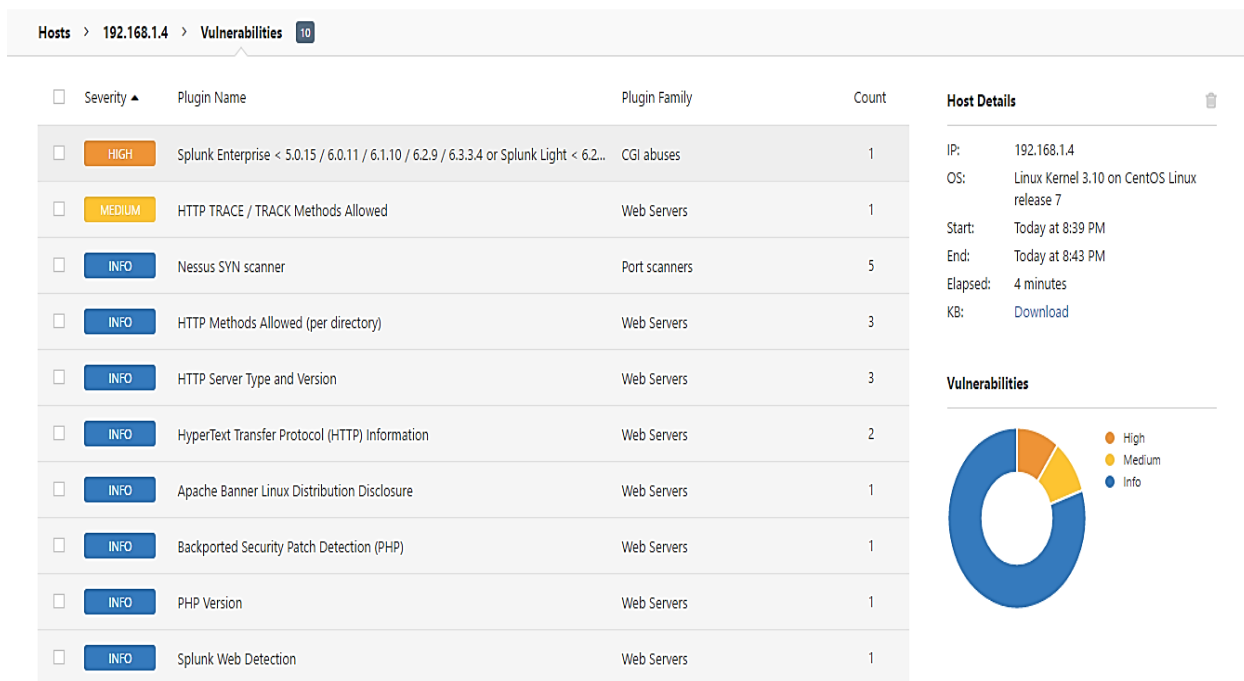


Image 7.8 Basic Network Scan

Hosts
>
192.168.1.4
>
Vulnerabilities
32

Severity	Plugin Name	Plugin Family	Count
HIGH	MySQL Unpassworded Account Check	Databases	1
HIGH	Splunk Enterprise < 5.0.15 / 6.0.11 / 6.1.10 / 6.2.9 / 6.3.3.4 or Splunk Light < 6.2...	CGI abuses	1
MEDIUM	HTTP TRACE / TRACK Methods Allowed	Web Servers	1
MEDIUM	MySQL Protocol Remote User Enumeration	Databases	1
MEDIUM	SSH Weak Algorithms Supported	Misc.	1
LOW	SSH Server CBC Mode Ciphers Enabled	Misc.	1
LOW	SSH Weak MAC Algorithms Enabled	Misc.	1
INFO	Nessus SYN scanner	Port scanners	5
INFO	Service Detection	Service detection	4
INFO	HTTP Server Type and Version	Web Servers	3
INFO	HyperText Transfer Protocol (HTTP) Information	Web Servers	2
INFO	Apache Banner Linux Distribution Disclosure	Web Servers	1

Host Details

IP: 192.168.1.4

MAC: 00:0c:29:cc:68:18

OS: Linux Kernel 3.10 on CentOS Linux release 7

Start: Today at 8:09 PM

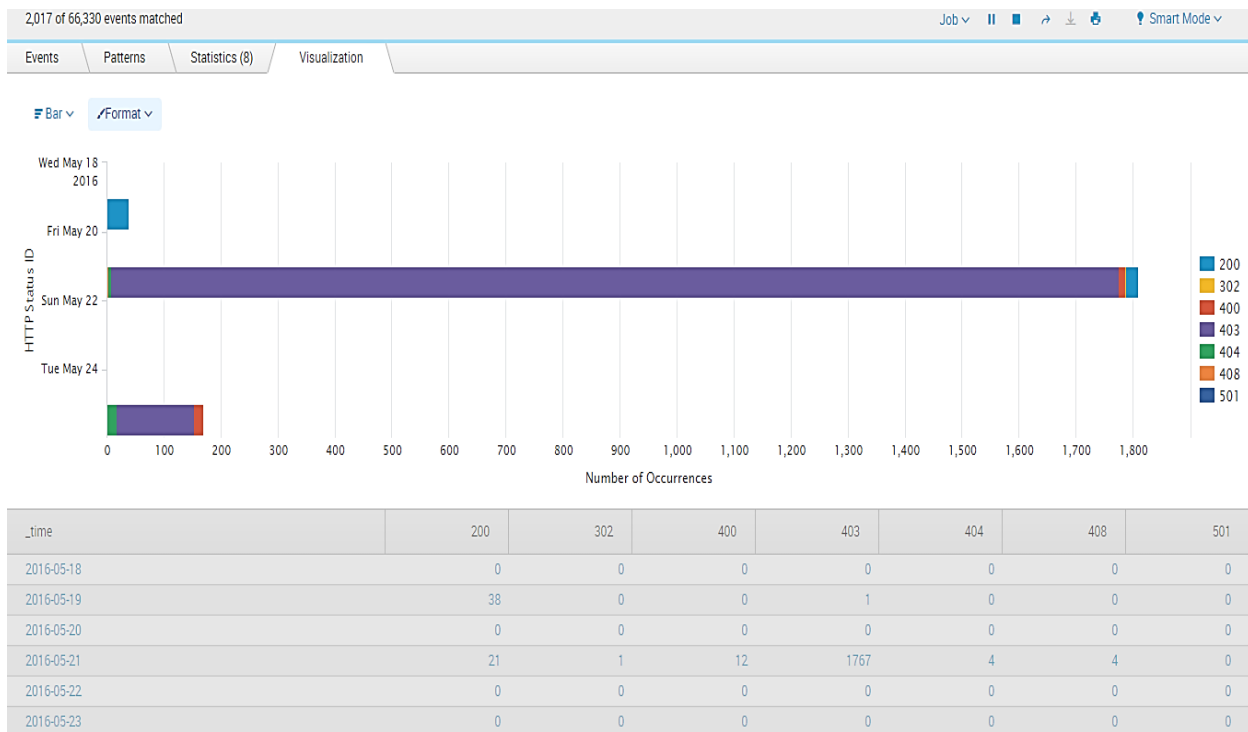
End: Today at 8:19 PM

Elapsed: 11 minutes

KB: [Download](#)

Vulnerabilities

High  
Medium  
Low  
Info



## 7.4 Identification and Mitigation of Web Attacks

Before we continue with some of the most widely used web attacks, first we need to explain the Mod\_Security rules, their characteristics and how they operate as well as how to modify them or create new ones.

### 7.4.1 Handling Mod\_Security Rules and False Positives

Since Mod\_Security rules are open source, this allows us the capability to see exactly what the rule is matching on as well as to create our own rules. At the start of the installation, we set Mod\_Security to DetectionOnly mode using the SecRuleEngine DetectionOnly command. We run Mod\_Security in a detection only mode for a while and simultaneously we review the events generated and the way the rules are triggered. Then, we decide if any modification to the rule set should be made before moving to protection mode, setting SecRuleEngine On.

It is inevitable that we will run into some false positive hits when using web application firewalls. This is not something that is unique to Mod\_Security. All web application firewalls will generate false positives from time to time. The following information will help to guide you through the process of identifying, fixing, implementing and testing new custom rules to address false positives.

Every rule set can have false positive in new environments False Positives happen with Mod\_Security Core Rules (CRS) mainly of the fact that the rules are "generic" in nature for successfully detecting the vast majority of attacks. Also due to the versatility of each web application and the different services that run on them there cannot be a "holy grail" of rule sets. In most cases, we need to customize our rule set according to the web applications we want to protect.

However, we need to be cautious and not be too hasty to remove a rule. If a particular rule is generating a few false positives that does not mean that we should remove the rule entirely. Remember, these rules were created for a reason. They are intended to block a known attack. But, for the purposes of this paper, we will use mainly our own customized rules based on Mod\_Security Core Rule Set (CRS).

In order to verify if we indeed have a false positive, we need to review our logs. This means that we need to look in the audit\_log or error\_log file first to see what the Mod\_Security message states. It will provide information as to which rule triggered. The last place to look, and actually the best source of information, is the modsec\_debug.log file. This file can show everything that Mod\_Security is doing, especially if you turn up the SecDebugLogLevel to 9. Keep in mind, however, that increasing the verbosity of the debug log does affect performance. So in this paper, it was decided to keep the debug logs to a minimum.

In general, it is recommended to try to limit the alteration of the Core Rules as much as possible. The more the rule files are altered, the less likely and harder it will be to upgrade to the newer releases since all the customizations will need to be recreated. What is proposed in this paper, is that to try to contain the changes to our own custom rules file(s) that are particular to web application. This is where we would want to add new signatures and also create rules to exclude false positives from the normal Core Rules files. There are two main ways to integrate the custom rules so that they work with the Core Rules.

## 1. Adding new white-listing rules

If you need to add new white-listing rules so that you can, for instance, allow a specific client IP address to pass through all of the Mod\_Security rules you should place this type of rule after the `modsecurity_crs_10_config.conf` file but before the other Core Rules. This is accomplished by creating a new rule file e.g. `modsecurity_customrules.conf` and place it in the same directory as the other Core Rules. This is assuming you are using the Apache Include directive to call up the Core Rules like this,

```
<IfModule security2_module>
```

```
Include mod_security.conf/rules/*.conf
```

```
</IfModule>
```

## 2. Adding new negative policy rules

If you need to add new negative policy rules, such as when you need to update a Core Rule that is causing a false positive, you should add these rules to a new rule file that come after all of the other Core Rules. Call this new file something like `modsecurity_crs_60_customrules.conf`. Just make sure that number in the filename is higher than any other rules file so it is read last. The rationale for placing these types of rules after the other rules is that you can then match up these new replacement rules with corresponding `SecRuleRemoveById` directives that will then disable the specific Core Rule(s) that are causing false positives. It is important to note that you need to use `SecRuleRemoveById` after ModSecurity has knowledge of the Rule ID you are actually removing. That is why this directive should be called up in your custom rules file that comes at the end.

In this paper, we chose for our purposes the first way and added new white-listing rules as the images that follow illustrate:

```

GNU nano 2.3.1                               File: mod_security.conf

LoadModule security2_module modules/mod_security2.so

<IfModule mod_security2.c>

    SecRuleEngine Off

    # ModSecurity Core Rules Set configuration
    IncludeOptional modsecurity.d/*.conf
    IncludeOptional modsecurity.d/activated_rules/*.conf

    # Default recommended configuration
    #SecRuleEngine DetectionOnly
    SecRequestBodyAccess On
    SecRule REQUEST_HEADERS:Content-Type "text/xml" \
        "id:'200000',phase:1,t:none,t:lowercase,pass,nolog,ctl:requestBodyProcessor=XML"
    SecRequestBodyLimit 13107200
    SecRequestBodyNoFilesLimit 131072
    SecRequestBodyInMemoryLimit 131072
    SecRequestBodyLimitAction Reject
    SecRule REQBODY_ERROR "!@eq 0" \
        "id:'200001', phase:2,t:none,log,deny,status:400,msg:'Failed to parse request body.',logdata:'%{reqbody}'"
    SecRule MULTIPART_STRICT_ERROR "!@eq 0" \
        "id:'200002',phase:2,t:none,log,deny,status:44,msg:'Multipart request body \
        failed strict validation: \

```

Image 7.11 Include Custom Rules with Apache Include Directive

```

[root@panoswaf bin]# systemctl restart httpd
[root@panoswaf bin]# cd ../../../../
[root@panoswaf /]# cd /etc/httpd/
[root@panoswaf httpd]# ll
total 16
drwxr-xr-x. 2 root root   35 May 15 21:54 conf
drwxr-xr-x. 2 root root 4096 May 15 22:29 conf.d
drwxr-xr-x. 2 root root 4096 May 15 21:54 conf.modules.d
drwxr-xr-x. 3 root wheel  47 Mar 29 21:05 crs
lrwxrwxrwx. 1 root root   19 May 15 21:54 logs -> ../../var/log/httpd
drwxr-xr-x. 3 root root 4096 May 26 18:02 modsecurity.d
lrwxrwxrwx. 1 root root   29 May 15 21:54 modules -> ../../usr/lib64/httpd/modules
-rw-r--r--. 1 root root   44 Mar 27 20:45 pma_pass
lrwxrwxrwx. 1 root root   18 May 15 21:54 run -> /run/httpd
[root@panoswaf httpd]# cd modsecurity.d/
[root@panoswaf modsecurity.d]# ll
total 160
drwxr-xr-x. 2 root root    6 Jun 10 2014 activated_rules
-rw-r--r--. 1 root wheel 1489 May 26 18:22 csrfcustomrules.conf
-rw-r--r--. 1 root wheel 2222 May 26 18:02 doscustomrules.conf
-rw-r--r--. 1 root wheel 4034 May 26 19:32 lficustomrules.conf
-rw-r--r--. 1 root wheel 4896 May 26 19:33 rficustomrules.conf
-rw-r--r--. 1 root wheel 43171 May 26 18:17 sqlcustomrules.conf
-rw-r--r--. 1 root wheel 96786 May 26 18:43 xssccustomrules.conf
[root@panoswaf modsecurity.d]# _

```

Image 7.12 Created New Custom Rules Files

At this point, let us present an example of a custom rule for Cross-site Scripting (XSS) Attacks based on the Mod\_Security core rule set:

*SecRule REQUEST\_FILENAME/ARGS/ARGS\_NAMES/REQUEST\_HEADERS*

```
"(?:\b(?:on(?:?:mo/key(?:?:press/down/up)/c(?:?:hange/lick)/s(?:?:elec/ubmi)t(?:?:un)?load/
dragdrop/resize/focus/blur)\b|W*?=/abort\b)/(?:l(?:?:owsrc\b|W*?\b(?:?:java/vb)script
/shell)/ivescript)/(?:href/url)\b|W*?\b(?:?:java/vb)script/shell)/mocha):
/type\b|W*?\b(?:?:text\b(?:?:|W*?\b(?:?:j(?:?:ava)?/ecma)script\b| [vbscript])
/application\b|W*?\b(?:?:java/vb)script\b)/s(?:?:?:tyle\b|W*?=. *|bexpression\b|W*
/timeout\b|W*?)\b|/rc\b|W*?\b(?:?:java/vb)script/shell/http:)/(?:c(?:?:opyparentfolder
/reatetextrange)/get(?:?:special/parent)folder
/<(?:?:body\b.*?\b(?:?:backgroun/onloa)d/input\b.*?|btype\b|W*?\bimage)\b|!/[CDAT
A]/script/meta)/.?(?:?:execscrip/addimpor)t(?:?:fromcharcod/cooki)e/innerhtml)\b)" |

"log, id:27950014,severity:2,msg:'Cross-site Scripting (XSS) Attack Vectors in Cookie
Values Detected by PanosWaf'"
```

This rule is inspecting the variables REQUEST\_FILENAME, ARGS, ARGS\_NAMES and REQUEST\_HEADERS in the entire html cookie and it is checking the cookie values for matches with a variety of possible combinations of vectors used on XSS attack.

For example, it looks for strings like:

*c(?:?:hange/lick):* A string that starts with "c" and ends with either "hange" or "lick", like change or click

*(?:?:java/vb)script/shell/http):* Strings that include and starts with "java" or "vb" and ends with "script", "shell" or "http", like javascript, javashell or vbscript.

Finally, we are also updating the "id" meta-data action by changing to a new number that represents our custom rule range and we present our own message.

## 7.4.2 SQL Injection Attacks

SQL injection is a code injection technique, used to attack data-driven applications, in which nefarious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker). SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server [11].

The images below illustrate the way we can monitor and analyze the logs by using the Splunk Security as well as create dashboards for identifying attack patterns, successful attack rates and how well are our waf rules working.

```
[Fri May 27 01:07:18.387216 2016] [:error] [pid 3408] [client 192.168.1.5] ModSecurity: Warning. Pattern match "(^[\\"'`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98;]+|[\\"'`\\xc2\\xb4\\xe2\\x80\\x99\\xe2\\x80\\x98;]+$)" at ARGS:password_new. [file "/etc/httpd/modsecurity.d/sqlcustomrules.conf"] [line "49"] [id "27981318"] [rev "2"] [msg "SQL Injection Attack: Common Injection Testing Detected by PanosWaf"] [data "Matched Data: ' found within ARGS:password_new: 5e7ko08[79']" [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "192.168.1.6"] [uri "/DVWA-1.9/vulnerabilities/csrf/"] [unique_id "V0dzlsIJtrl iws84wCLksAAAAAU"]
```

```
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs
```

```
[Thu May 26 22:26:29.965046 2016] [:error] [pid 14379] [client 192.168.1.5] ModSecurity: Warning. Pattern match "(?i:\\\\b(?:s(?:t(?:d(?:dev_pop|samp)?|r(?:_to_date|cmp))|u(?:b(?:str(?:ing(_index)?|(?dat|time)|m)|e(?:c(?:_to_time|ond)|ssion_user)|y s(?:tem_user|date)|ha(1|2)?|oundex|chema|ig?n|pace|qrt)|i(?:s(null|_free_lock|ipv4_compat|ipv4_mapped|ipv4) ...)" at ARGS:id_forum. [file "/etc/httpd/modsecurity.d/sqlcustomrules.conf"] [line "115"] [id "27950001"] [rev "2"] [msg "SQL Injection Attack Detected by PanosWaf"] [data "Matched Data: UNION/**/SELECT found within ARGS:id_forum: -1/**/UNION/**/SELECT 1758597684--"] [severity "CRITICAL"] [ver "OWASP_CRS/2.2.9"] [maturity "9"] [accuracy "8"] [tag "OWASP_CRS/WEB_ATTACK/SQL_INJECTION"] [tag "WASCTC/WASC-19"] [tag "OWASP_TOP_10/A1"] [tag "OWASP_AppSensor/CIE1"] [tag "PCI/6.5.2"] [hostname "192.168.1.6"] [uri "/forum.php"] [unique_id "V0d N5S7wEXY3C@rYyXfK0AAAAA"]
```

```
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs
```

Image 7.13 Logs from Splunk of SQL Injection Attacks

This image below shows the top uri found among the SQL injection attempts in 1 day span.

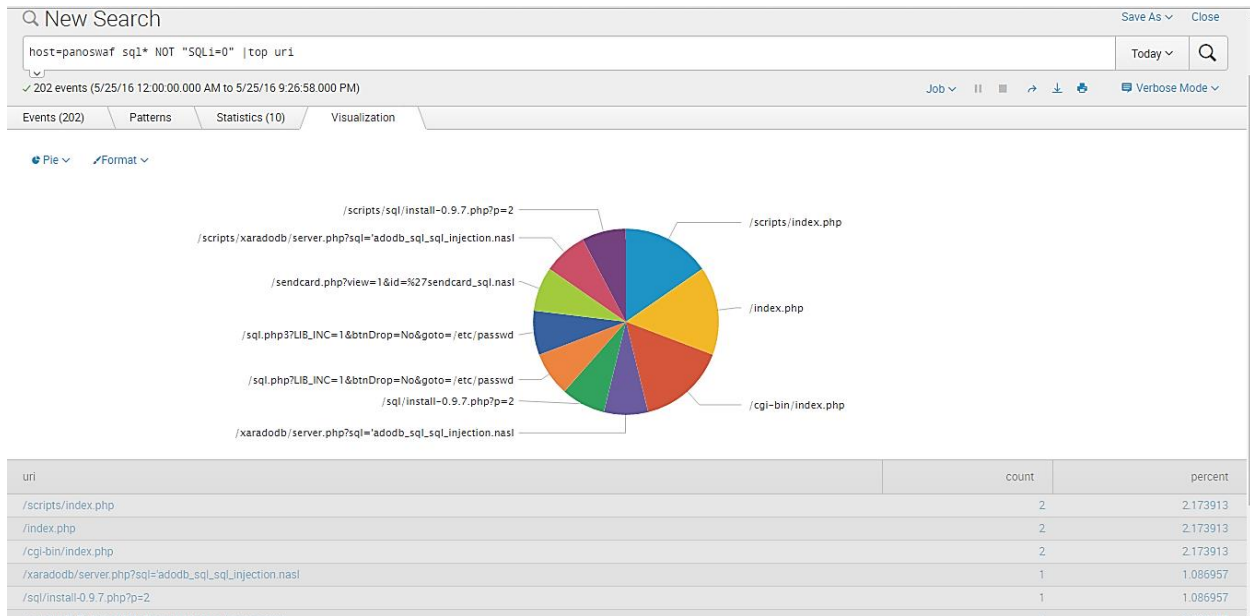


Image 7.14 Top URI from the SQL Injection Attacks

Finally, the images below show the success rate of the SQL Injection attacks before and after our Mod\_Security waf implementation, where clearly most of the attempts are blocked successfully as well as the new file that we created containing our custom rules for detecting SQL Injection Attacks.



Image 7.15 Success Rate of SQL Injection Attacks before and after our WAF Implementation



```
GNU nano 2.3.1 File: sqlcustomrules.conf Modified
<IfModule mod_security2.c>
SecRuleEngine DetectionOnly
SecRequestBodyAccess On
SecResponseBodyAccess On
SecResponseBodyMimeType text/plain text/html text/xml applications/octet-stream
SecDataDir /tmp
</IfModule>

#
# --[ This Rule Detects SQL Comment Sequences ]--
#
SecRule REQUEST_COOKIES!REQUEST_COOKIES:/__utm/!REQUEST_COOKIES:/_pk_ref/!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS!XML:/* "(^!?!?\\n*/!';]--!-[\\s\\r\\n\\u\\f!)(?:-$

#
# --[ This Rule Detects Common SQL Hex Evasion Methods ]--
#
SecRule REQUEST_COOKIES!REQUEST_COOKIES:/__utm/!REQUEST_COOKIES:/_pk_ref/!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS!XML:/* "(?:i:(?:a|!^\\d))0x[a-f\\d]{3,}[a-f\\d]*$

#
# --[ This Rule Detects String Injections ]--
#
SecRule REQUEST_COOKIES!REQUEST_COOKIES:/__utm/!REQUEST_COOKIES:/_pk_ref/!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS!XML:/* "(^[\\'\"''';];!+!\\'\"''';!+!$)" "phase:2$

#
# --[ This Rule Detects SQL Operators ]--
#
SecRule REQUEST_COOKIES!REQUEST_COOKIES:/__utm/!REQUEST_COOKIES:/_pk_ref/!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS!XML:/* "(?:i:(\\N=|\\N&|\\N!|>>|<<|=|<|>|<=$

#
# --[ This Rule Detects SQL Tautologies ]--
#
```

Image 7.16 Custom File for Rules Detecting SQL Injection Attacks

### 7.4.3 Cross-site Scripting (XSS) Attacks

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted web sites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page [\[12\]](#).

The first two images below illustrate the dashboards that we created to identify the XSS attack patterns and successful attack rates before and after our WAF implementation. We can clearly discern from the Max XSS Attacks dashboard that after our WAF implementation the max(XSS) – top number of successful XSS attacks – has significantly decreased.



[illegible]

```

#
# --[ This Rule Detects Script Based XSS Attacks - Category 1 ]--
#
SecRule ARGS "(?i)(<script[^>]*>|</script>|<script[^>]*>|</script>|<script[^>]*>|</script>|<script[^>]*>|</script>)"@m
#
# --[ This Rule Detects Different Event Handlers used for XSS Attacks - Category 2 ]--
# XSS vectors making use of event handlers like onerror, onload etc., e.g., <body onload="alert(1)">
#
SecRule ARGS "(?i)(\\\"'`;<br>\\x00-9a=}|on\\w+.*=)\"" "id:'27973337',phase:2,t:none,rev:'1',ver:'OWASP_CRS/2.2.9',maturity:'1',accuracy:'8',t:urlDecodeUni,t:htmlEnti$
#
# --[ This Rule Detects Javascript Based XSS Attacks - Category 3 ]--
# XSS vectors making use of Javascripts URIs, e.g., <p style="background:url(javascript:alert(1))">
#
SecRule ARGS "(?i)((?:=|UWS.*RNS.*LNS.*\\O.S.*[\\"])*S.*S.*C.S.*RNS.*LNS.*P.S.*T.S.*:|colon|{\\s}\\s|allowscriptaccess|\\s}\\s|src|\\s}\\s|data:text\\/html|\\s}\\s|)$
#
# These Rules Detect Combinations of XSS Attacks -event handlers, scripts, uris,...-
#
SecRule REQUEST_COOKIES:!REQUEST_COOKIES:/_utm/;!REQUEST_COOKIES/_pk_ref/;!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS:XQL/* "%bm jscrip<script>onsubmit copyparentfolder$
"phase:2,id:'27981136',rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'8',accuracy:'8',t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,pass,nolog,set$
#
SecRule REQUEST_COOKIES:!REQUEST_COOKIES:/_utm/;!REQUEST_COOKIES/_pk_ref/;!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS:XQL/* "%bgetparentfolder\b" \
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'8',accuracy:'8',capture,t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,ctl:auditLogParts=$
#
SecRule REQUEST_COOKIES:!REQUEST_COOKIES:/_utm/;!REQUEST_COOKIES/_pk_ref/;!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS:XQL/* "%bonmousedown\b\\w*?=" \
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'8',accuracy:'8',capture,t:none,t:htmlEntityDecode,t:compressWhiteSpace,t:lowercase,ctl:auditLogParts=$
#
SecRule REQUEST_COOKIES:!REQUEST_COOKIES:/_utm/;!REQUEST_COOKIES/_pk_ref/;!REQUEST_COOKIES_NAMES!ARGS_NAMES!ARGS:XQL/* "%bsrc\b\\w*?nshell:" \

```

Page | 59

#### 7.4.4 File Inclusion Attacks

File inclusion vulnerability allows an attacker to access unauthorized or sensitive files available on the web server or to execute malicious files on the web server by making use of the 'include' functionality. This vulnerability is mainly due to a bad input validation mechanism, wherein the user's input is passed to the file include commands without proper validation. The impact of this vulnerability can lead to malicious code execution on the server or reveal data present in sensitive files, etc [\[13\]](#). There are two types of file inclusion, Remote File Inclusion (RFI) and Local File Inclusion (LFI).

Remote File Inclusion (RFI) is a type of vulnerability most often found on websites. It allows an attacker to include a remote file, usually through a script on the web server. The vulnerability occurs due to the use of user-supplied input without proper validation.

The Local File Inclusion (LFI) vulnerability is a process of including the local files available on the server. This vulnerability occurs when a user input contains the path to the file that has to be included. When such an input is not properly sanitized, the attacker may give some default file names and access unauthorized files, or an attacker may also make use of directory traversal characters and retrieve sensitive files available in other directories.

The first image that follows shows some of the logs that captured from the Splunk where we can identify the RFI attack that took place. The second image is of the dashboard that we created to identify the LFI attack patterns by monitoring the Query statistics and successful attack rates before and after our WAF implementation. We can clearly discern from the Top LFI Attacks dashboard that after our WAF implementation the top number of successful LFI attacks has drastically declined.

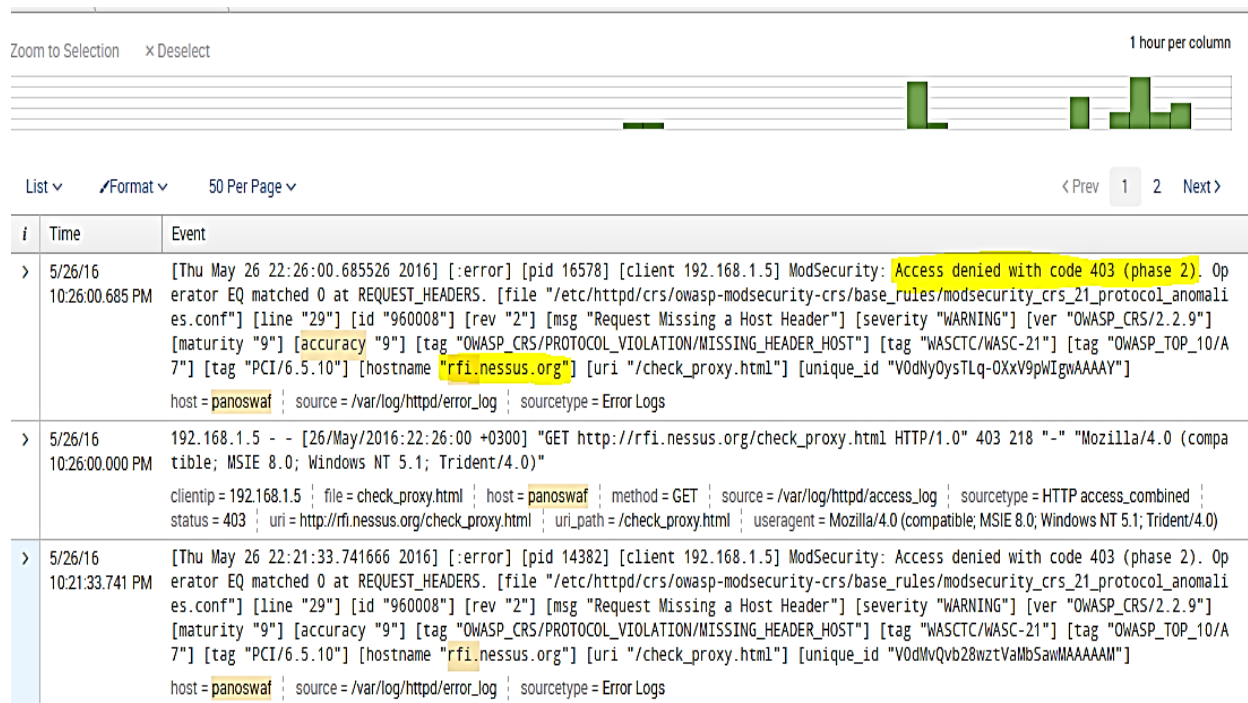


Image 7.21 Logs from Splunk of RFI Attacks



Image 7.22 LFI Attacks Overview before and after WAF Implementation

Furthermore, the last two images illustrate the new files that we created containing our custom rules for detecting the LFI and RFI Attacks respectively.

```

GNU nano 2.3.1                                File: lricustomrules.conf                                Modified
#
# These Rules are based on the ruleset that was created by Trustwave SpiderLabs Research Team and includes data from:
#
#   http://www.emergingthreats.net/
#

SecRule REQUEST_FILENAME "!@pmFromFile modsecurity_46_slr_et_lfi.data" "id:'2000001',phase:2,nolog,pass,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePat$

# These Rules Detect container.php theme_directory parameter local file inclusion

SecRule REQUEST_LINE "@contains /container.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,logdata:'%(TX.0)',seve$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule ARGS:theme_directory "@contains ../" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected Acute Control Panel container.php theme_directory parameter$

# These Rules Detect header.php theme_directory parameter local file inclusion

SecRule REQUEST_LINE "@contains /header.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,logdata:'%(TX.0)',severit$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule ARGS:theme_directory "@contains ../" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected Acute Control Panel header.php theme_directory parameter lo$

# These Rules Detect main.inc.php mj_config parameter local file inclusion

SecRule REQUEST_LINE "@contains /main.inc.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,logdata:'%(TX.0)',sever$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule ARGS:mj_config[src_path] "@contains ../" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected Basebuilder main.inc.php mj_config Parameter Local File$

# These Rules Detect block_center_down.php local file inclusion

SecRule REQUEST_LINE "@contains /block_center_down.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,logdata:'%(TX.$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule ARGS:row_mysql_blocks_center_down[file] "@contains ../" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected Blogplus block_center_down.php Local Fil$

#SecMarker END_SLR_ET_LFI_RULES

```

Image 7.23 Custom File for Rules Detecting LFI Attacks

```

GNU nano 2.3.1                                File: rfcustomrules.conf                                Modified
#
# These Rules are based on the ruleset that was created by Trustwave SpiderLabs Research Team and includes data from:
#
#   http://www.emergingthreats.net/
#

SecRule REQUEST_FILENAME "!@pmFromFile modsecurity_46_slr_et_rfi.data" "id:'2000003',phase:2,nolog,pass,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePat$

# These Rules Detect Path Parameter Remote File Inclusion Attempt

SecRule REQUEST_LINE "@contains /ardeaCore/lib/core/ardeaInit.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,log$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule &TX:'/RFI.*ARGS:pathForArdeaCore/' "@gt 0" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected ArdeaCore pathForArdeaCore Parameter Remote File Incl$

# These Rules Detect standard.php page_include Parameter Remote File Inclusion

SecRule REQUEST_LINE "@contains /layouts/standard.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,logdata:'%(TX.0$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule &TX:'/RFI.*ARGS:page_include/' "@gt 0" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected standard.php page_include Parameter Remote File Inclusion$

# These Rules Detect pageDescriptionObject.php LibDir Parameter Remote File Inclusion Attempt

SecRule REQUEST_LINE "@contains /lib/page/pageDescriptionObject.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,l$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule &TX:'/RFI.*ARGS:LibDir/' "@gt 0" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected pageDescriptionObject.php LibDir Parameter Remote File Inclusio$

# These Rules Detect layoutHeaderFuncs.php LibDir Parameter Remote File Inclusion Attempt

SecRule REQUEST_LINE "@contains /lib/layout/layoutHeaderFuncs.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,log$
SecRule REQUEST_LINE "@contains GET " "chain"
SecRule &TX:'/RFI.*ARGS:LibDir/' "@gt 0" "ctl:auditLogParts=+E,setvar:'tx.msg=PanosWaf detected layoutHeaderFuncs.php LibDir Parameter Remote File Inclusion At$

# These Rules Detect layoutManager.php LibDir Parameter Remote File Inclusion Attempt

SecRule REQUEST_LINE "@contains /lib/layout/layoutManager.php" "chain,phase:2,block,t:none,t:urlDecodeUni,t:htmlEntityDecode,t:normalisePathWin,capture,logdata:$
SecRule REQUEST_LINE "@contains GET " "chain"

```

Image 7.24 Custom File for Rules Detecting RFI Attacks

## 7.4.5 Cross-Site Request Forgery (CSRF) Attacks

Cross-site request forgery, also known as one-click attack or session riding and abbreviated as CSRF or XSRF, is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the website trusts. Unlike cross-site scripting (XSS), which exploits the trust of a user that has for a particular site, CSRF exploits the trust that a site has in a user's browser.

CSRF forces an end user to execute unwanted actions on a web application in which they are currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application [14].

The first image that follows illustrates the dashboard that we created to identify the CSRF attack patterns and successful attack rates before and after our WAF implementation, mentioned in the middle of the image as "PanosWaf". We can clearly discern from the Top CSRF Attacks dashboard that after our WAF implementation the top number of successful CSRF attacks has significantly decreased. We have underlined the two top numbers 90 and 15 in each case.

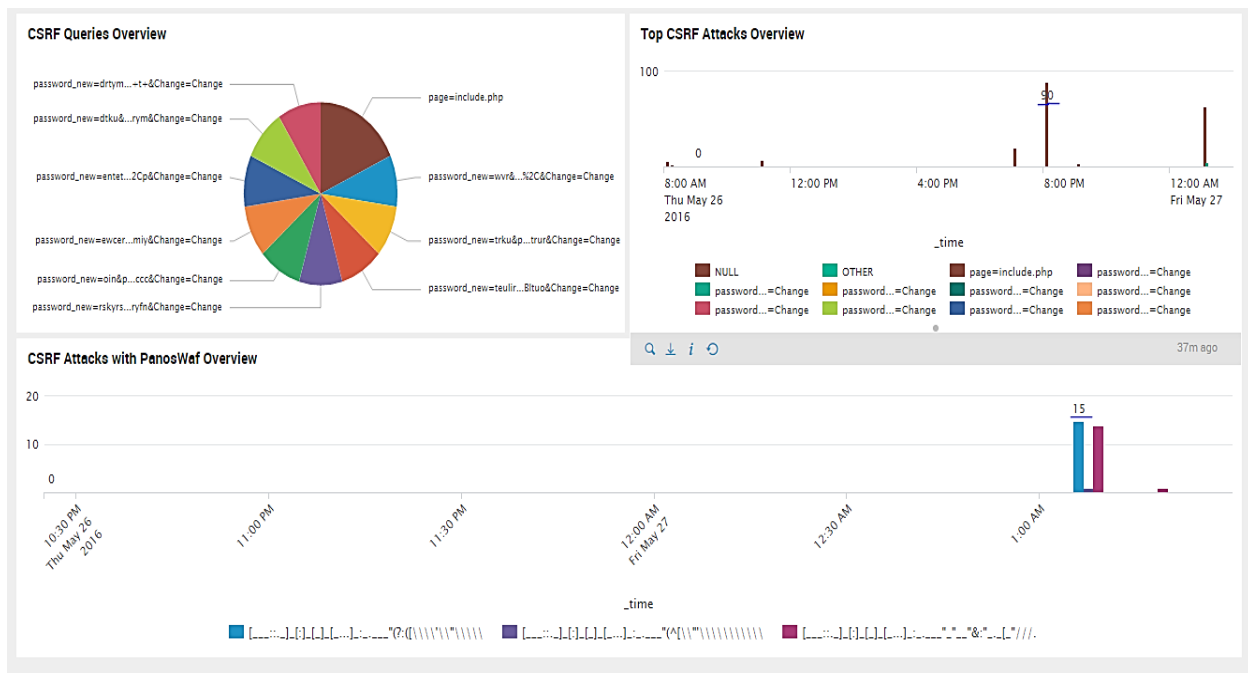


Image 7.25 CSRF Attacks Overview before and after WAF Implementation



The second image below shows some of the logs that captured from the Splunk where we can identify the SCRF attack that took place.

```
5/27/16 [Fri May 27 01:07:58.025053 2016] [:error] [pid 3409] [client 192.168.1.5] ModSecurity: Warning. Match of "eq 1" against "&ARGS:CSR
1:07:58.025 AM F_TOKEN" required. [file "/etc/httpd/modsecurity.d/csrfcustomrules.conf"] [line "27"] [id "27981143"] [msg "CSRF Attack Detected by
PanosWaf - Missing CSRF Token."] [hostname "192.168.1.6"] [uri "/DVWA-1.9/vulnerabilities/csrf/"] [unique_id "V0dzurpcPPFcyh@kjBf0n
QAAAAAY"]
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs

5/27/16 [Fri May 27 01:07:54.404257 2016] [:error] [pid 3409] [client 192.168.1.5] ModSecurity: Warning. Match of "eq 1" against "&ARGS:CSR
1:07:54.404 AM F_TOKEN" required. [file "/etc/httpd/modsecurity.d/csrfcustomrules.conf"] [line "27"] [id "27981143"] [msg "CSRF Attack Detected by
PanosWaf - Missing CSRF Token."] [hostname "192.168.1.6"] [uri "/DVWA-1.9/vulnerabilities/csrf/"] [unique_id "V0dzurpcPPFcyh@kjBf0n
AAAAAY"]
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs

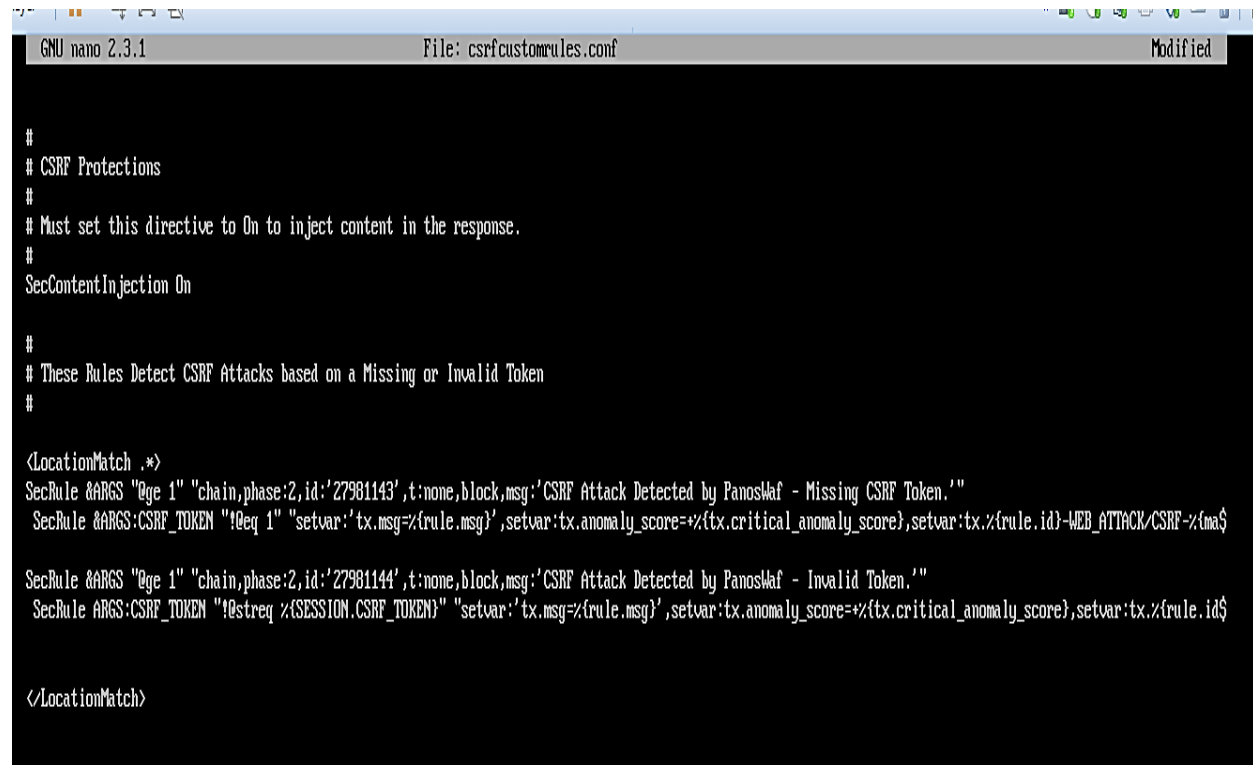
5/27/16 [Fri May 27 01:07:54.404257 2016] [:error] [pid 3409] [client 192.168.1.5] ModSecurity: Warning. Match of "eq 1" against "&ARGS:CSR
1:07:54.404 AM F_TOKEN" required. [file "/etc/httpd/modsecurity.d/csrfcustomrules.conf"] [line "27"] [id "27981143"] [msg "CSRF Attack Detected by
PanosWaf - Missing CSRF Token."] [hostname "192.168.1.6"] [uri "/DVWA-1.9/vulnerabilities/csrf/"] [unique_id "V0dzurpcPPFcyh@kjBf0n
AAAAAY"]
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs

5/27/16 [Fri May 27 01:07:45.679426 2016] [:error] [pid 2906] [client 192.168.1.5] ModSecurity: Warning. Match of "eq 1" against "&ARGS:CSR
1:07:45.679 AM F_TOKEN" required. [file "/etc/httpd/modsecurity.d/csrfcustomrules.conf"] [line "27"] [id "27981143"] [msg "CSRF Attack Detected by
PanosWaf - Missing CSRF Token."] [hostname "192.168.1.6"] [uri "/DVWA-1.9/vulnerabilities/csrf/"] [unique_id "V0dzsU0FdLORY4B5lCt9e
gAAAAAM"]
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs

5/27/16 [Fri May 27 01:07:45.679426 2016] [:error] [pid 2906] [client 192.168.1.5] ModSecurity: Warning. Match of "eq 1" against "&ARGS:CSR
1:07:45.679 AM F_TOKEN" required. [file "/etc/httpd/modsecurity.d/csrfcustomrules.conf"] [line "27"] [id "27981143"] [msg "CSRF Attack Detected by
PanosWaf - Missing CSRF Token."] [hostname "192.168.1.6"] [uri "/DVWA-1.9/vulnerabilities/csrf/"] [unique_id "V0dzsU0FdLORY4B5lCt9e
gAAAAAM"]
host = panoswaf | source = /var/log/httpd/error_log | sourcetype = Error Logs
```

Image 7.26 Logs from Splunk of CSRF Attacks

Finally, the last image illustrates the new file that we created containing our custom rules for detecting the CSRF Attacks.



```
GNU nano 2.3.1 File: csrfcustomrules.conf Modified

#
# CSRF Protections
#
# Must set this directive to On to inject content in the response.
#
SecContentInjection On

#
# These Rules Detect CSRF Attacks based on a Missing or Invalid Token
#

<LocationMatch .*>
SecRule &ARGS "eq 1" "chain,phase:2,id:'27981143',t:none,block,msg:'CSRF Attack Detected by PanosWaf - Missing CSRF Token.'"
SecRule &ARGS:CSR_TOKEN "!eq 1" "setvar:'tx.msg=%{rule.msg}',setvar:tx.anomaly_score=%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-WEB_ATTACK/CSRF-%{ima$

SecRule &ARGS "eq 1" "chain,phase:2,id:'27981144',t:none,block,msg:'CSRF Attack Detected by PanosWaf - Invalid Token.'"
SecRule &ARGS:CSR_TOKEN "!streq %{$SESSION.CSRF_TOKEN}" "setvar:'tx.msg=%{rule.msg}',setvar:tx.anomaly_score=%{tx.critical_anomaly_score},setvar:tx.%{rule.id$

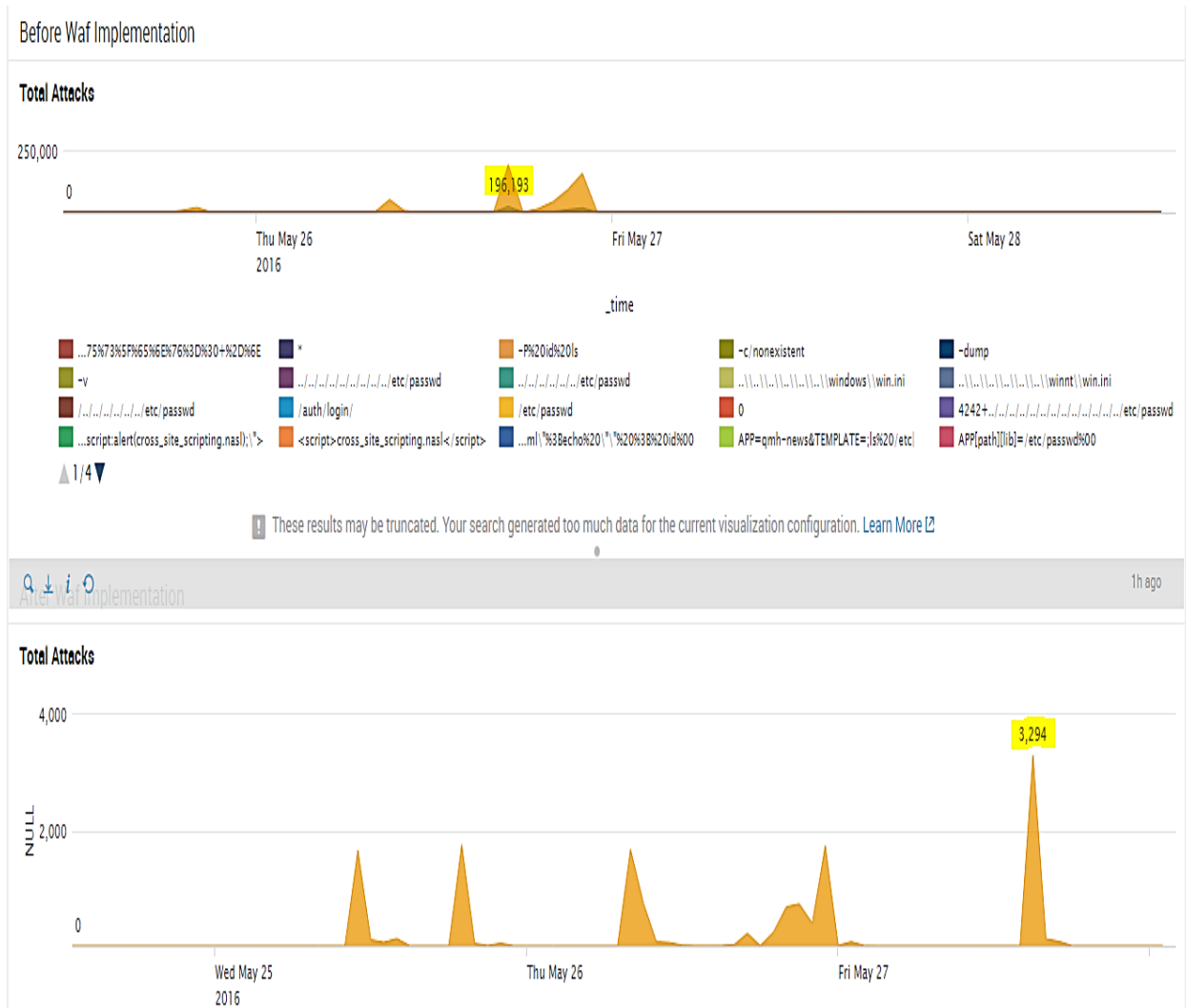
</LocationMatch>
```

Image 7.27 Custom File for Rules Detecting CSRF Attacks



### 7.4.6 Total Attacks

At this point, we can totally summarize the different attacks that took place in a dashboard called "Total Attacks". The image that follows illustrates the dashboard that we created to identify the successful attack rates from all the attacks that took place before and after our WAF implementation. We can clearly distinguish how greatly the top number of successful attacks has dropped after our WAF implementation. We have marked the two top numbers 196,193 and 3,294 in each case.



### Image 7.28 Total Attacks Overview before and after WAF Implementation

## 8. Conclusions

In the presence of multiple known and unknown threats from widespread vulnerabilities on web applications and attackers continuously attempting to exploit them, the need for robust, universal, flexible, efficient and easy to use defensive tools is more than obvious. Mod\_Security module and generally Web Application Firewalls are an excellent addition to secure web servers from application layer attacks.

A well configured WAF can protect web servers from almost all kind of application layer attacks including XSS, SQL Injection, and Parameter Manipulation attacks. With its ability to filter on any HTTP, HTTPS GET and POST request, we can address all malicious requests targeting our web servers. Support of Regex (Regular Expressions) allows security admin to create more powerful rule base. WAF is not intended to take place of Firewalls, IDS or IPS, it has an ability to filter HTTP traffic. Moreover, all data is security relevant and should be indexed. IT security teams need to properly investigate security incidents and identify threats as well as to include more than security data from traditional security products such as firewalls, IDS or anti-malware. The data indexed also needs to include "non-security" data from sources such as OS logs, LDAP/AD, DNS, NetFlow and email/web servers. To detect advanced threats, all non-security and security data must reside in a single repository that is monitored in real time. Real-time analytics can accelerate incident investigations and automatically detect the anomalies that may or not be advanced threats. Statistics can help with this detection by looking for events that are standard deviations off the norm. Correlations can also help by detecting combinations of events that are rarely seen and are suspicious. Solutions for information and event monitoring and management such as Splunk Security can provide exactly all that.

Overall, WAF is a great concept to secure web servers from Layer 7 attacks and it can secure badly coded applications to some extent. Also, solutions like Splunk can provide information and aid in security incidents handling which otherwise may be impossible. However, we have merely scratched the surface of what is possible with using Mod\_Security WAF module and Splunk as defensive mechanisms. They may become some of your favorite tools in your arsenal. Not only they do make attackers visible, but they also let you respond with a tangible impact on the malicious client. So let us summarize in a few bullets our findings.

Regarding Splunk,

- All the original data is retained and can be searched on without reliance on vendors
- Its search language can do automated base lining and the calculation of anomalies as well as advanced correlations.
- It is easy to operate and create appropriate reports and various visualizations of data.

- It has the capability to help you detect known threats but also unknown through the indexing of “non-security” data.

Regarding Mod\_Security,

- It is necessary to adapt the WAF filter (rules & exclusions) to the particular web application that is being protected.
- WAF does not eliminate vulnerabilities; it just screens the attack vector and helps mitigate the security risks.
- It has functional limitations; WAF is not able to protect a web application from all possible vulnerabilities, especially if not configured correctly.
- It is certainly a huge asset for fortifying your web security, however creating custom rules, exclusions and fixing false positives is not simple enough. It requires deep knowledge as well as a lot of and constant tuning before it can be used to block attacks effectively and efficiently.

## 9. References

- [1]. <https://www.centos.org/>
- [2]. <https://httpd.apache.org/>
- [3]. <http://news.netcraft.com/archives/2016/04/21/april-2016-web-server-survey.html>
- [4]. [http://xmodulo.com/harden-apache-web-server-mod\\_security-mod\\_evasive-centos.html](http://xmodulo.com/harden-apache-web-server-mod_security-mod_evasive-centos.html)
- [5]. <https://github.com/RandomStorm/DVWA>
- [6]. <http://docs.splunk.com/Documentation/Splunk/6.0.3/Installation/InstallonLinux>
- [7]. Cyber Operations: Building, Defending, and Attacking Modern Computer Networks, Chapter 11 – Logs & Logging, Mike O' Leary, Department of Mathematics, Towson University, Towson, MD, US, 2015
- [8]. ModSecurity Handbook, Chapter 4 - Logging, by Ivan Ristiæ, 2010
- [9]. <http://www.acunetix.com/>
- [10]. <http://www.tenable.com/products/nessus/nessus-professional>
- [11]. [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- [12]. [https://www.owasp.org/index.php/Cross-site\\_scripting](https://www.owasp.org/index.php/Cross-site_scripting)
- [13]. <http://resources.infosecinstitute.com/file-inclusion-attacks/>
- [14]. [https://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery\\_%28CSRF%29](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29)