

Security of Learning Management Systems

Yılmaz Demirci

Submitted to the
Institute of Graduate Studies and Research
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
September 2013
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazım Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Asst. Prof. Dr. Mustafa Rıza
Supervisor

Examining Committee

1. Asst. Prof. Dr. Arif Akkeleş

2. Asst. Prof. Dr. Mustafa Rıza

3. Asst. Prof. Dr. Müge Saadetoğlu

ABSTRACT

In this thesis I discuss the security issues of the Moodle Learning Management System. Therefore first the security vulnerabilities of web applications in general are discussed and then the security risks of Moodle and its solutions are presented. Furthermore a complete step-by-step installation guideline is proposed in order to create a Moodle installation with maximum security.

Keywords: Moodle, security of Moodle, security risks of 2013

ÖZ

Bu tezde Moodle Öğrenme Yönetim Sistemi' nin güvenlik sorunlarını ele aldım. Bu sebeple web uygulamalarının genel güvenlik açıkları öncelikle ele alındı ve sonra Moodle' ın güvenlik riskleri ve bunların çözümleri sunuldu. Ayrıca, maksimum güvenlikte bir Moodle kurulumu yapmak için adım adım tam bir kurulum rehberi önerildi.

Anahtar Kelimeler: Moodle, Moodle güvenliği, 2013 yılının güvenlik riskleri

To My Father

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my family for the continuous support to my education, for their patience, motivation, enthusiasm, and immense love.

I would like to express my deepest gratitude to my supervisor, Asst. Prof. Dr. Mustafa Rıza, for his continuous support, guidance as well as encouragement throughout this study.

Besides, I would also like to thank the members of my thesis committee and the substitute member of my thesis defence committee: Asst. Prof. Dr. Müge Saadetoğlu, Asst. Prof. Dr. Arif Akkeleş and Asst. Prof. Ersin Kuset Bodur, for their encouragement, insightful comments, and hard but constructive questions.

My sincere thanks also go to Müge Saadetoğlu for her support in and out of the faculty along with her contributions to my master education.

Last but not least; I would like to thank my friends and EMU Mathematics Club: Asri Karadeniz, Emre Bulut, Hülya Demez, İbrahim Avcı, Recep Duranay, Sedef Sultan Emin and Yeşim Bekar for their endless insight, patience, and support.

TABLE OF CONTENTS

| | |
|--|-----|
| ABSTRACT | iii |
| ÖZ | iv |
| DEDICATION | v |
| ACKNOWLEDGEMENTS | vi |
| LIST OF FIGURES | xii |
| LIST OF ABBREVIATIONS | xiv |
| INTRODUCTION | 1 |
| 1.1 Server OS: Debian 7.1 Wheezy | 3 |
| 1.2 Web Server: Apache2 | 5 |
| 1.3 DBMS: MySQL | 6 |
| 1.4 Conclusion | 6 |
| SECURITY RISKS | 8 |
| 2.1 Injection | 8 |
| 2.1.1 Sample Scenario: SQL Injection | 9 |
| 2.2 Broken Authentication and Session Management | 9 |
| 2.2.1 Sample Scenario: Session Hijacking | 10 |
| 2.3 Cross-Site Scripting (XSS) | 11 |
| 2.3.1 Stored Cross-Site Scripting | 11 |
| 2.3.2 Reflected Cross-Site Scripting | 12 |
| 2.3.3 DOM Based Cross-Site Scripting | 12 |

| | |
|--|----|
| 2.3.4 Sample Scenario: Stored XSS..... | 12 |
| 2.4 Insecure Direct Object References | 13 |
| 2.4.1 Sample Scenario: Missing Access Control | 13 |
| 2.5 Security Misconfiguration | 14 |
| 2.5.1 Sample Scenario: Server Misconfiguration | 14 |
| 2.6 Sensitive Data Exposure..... | 14 |
| 2.6.1 Sample Scenario: Stealing of a Backup File..... | 15 |
| 2.7 Missing Function Level Access Control | 15 |
| 2.7.1 Sample Scenario: Missing Function Level Access Control | 16 |
| 2.8 Cross-Site Request Forgery (CSRF) | 16 |
| 2.8.1 Sample Scenario: CSRF Attack..... | 16 |
| 2.9 Using Components with Known Vulnerabilities..... | 17 |
| 2.9.1 Sample Scenario: MySQL - Bug #64884 | 18 |
| 2.10 Invalidated Redirects and Forwards | 18 |
| 2.10.1 Sample Scenario: Redirects | 18 |
| ASSESSMENT OF SECURITY RISKS IN MOODLE..... | 19 |
| 3.1 Injection..... | 19 |
| 3.1.1 General Aspects | 19 |
| 3.1.2 How Moodle Avoids the Risk | 19 |
| 3.2 Broken Authentication and Session Management..... | 21 |
| 3.2.1 General Aspects | 21 |
| 3.2.2 How Moodle Avoids the Risk | 21 |

| | |
|--|----|
| 3.3 Cross-Site Scripting (XSS)..... | 26 |
| 3.3.1 General Aspects | 26 |
| 3.3.2 How Moodle Avoids the Risk | 26 |
| 3.4 Insecure Direct Object References | 27 |
| 3.4.1 General Aspects | 27 |
| 3.4.2 How Moodle Avoids the Risk | 27 |
| 3.5 Security Misconfiguration | 28 |
| 3.5.1 General Aspects | 28 |
| 3.5.2 How Moodle Avoids the Risk | 29 |
| 3.6 Sensitive Data Exposure..... | 30 |
| 3.6.1 General Aspects | 30 |
| 3.6.2 How Moodle Avoids the Risk | 30 |
| 3.7 Missing Function Level Access Control | 30 |
| 3.7.1 General Aspects | 30 |
| 3.7.2 How Moodle Avoids the Risk | 31 |
| 3.8 Cross-Site Request Forgery (CSRF) | 31 |
| 3.8.1 General Aspects | 31 |
| 3.8.2 How Moodle Avoids the Risk | 31 |
| 3.9 Using Components with Known Vulnerabilities..... | 33 |
| 3.9.1 General Aspects | 33 |
| 3.9.2 How Moodle Avoids the Risk | 33 |
| 3.10 Invalidated Redirects and Forwards | 34 |

| | |
|--|----|
| 3.10.1 General Aspects | 34 |
| 3.10.2 How Moodle Avoids the Risk | 34 |
| SECURING MOODLE INSTALLATION..... | 35 |
| 4.1 Secure Server Installation..... | 35 |
| 4.2 Secure Moodle Installation..... | 35 |
| 4.2.1 Preparation | 35 |
| 4.2.2 Start Your Installation..... | 36 |
| 4.2.3 Configuration of Site Wide Settings..... | 39 |
| 4.3 Suggestions..... | 41 |
| 4.3.1 Change Your Default Theme | 41 |
| 4.3.2 Take Regular Backups | 41 |
| 4.3.3 Site Wide Settings..... | 41 |
| SERVER INSTALLATIONS | 44 |
| 5.1 Setting up Virtual Machine | 44 |
| 5.2 Installing Debian Wheezy | 47 |
| 5.3 Installation of Other Components Using Automated Scripts..... | 50 |
| 5.3.1 Running Basic Apache, MySQL and PHP Installation Script..... | 50 |
| 5.3.2 An Automated Full ISPConfig 3 Installer Script..... | 52 |
| CONCLUSION..... | 53 |
| REFERENCES..... | 55 |
| APPENDICES | 58 |
| Appendix A: Ettercap Attack Logs | 59 |

Appendix B: LAMP_install.sh 60

LIST OF FIGURES

| | |
|--|----|
| Figure 1 - Moodle Statistics | 2 |
| Figure 2 – Usage of Server-Side Programming Languages for Websites | 2 |
| Figure 3 - Usage of OS for Websites | 4 |
| Figure 4 - Usage of Subcategories of UNIX..... | 4 |
| Figure 5 - Usage of Subcategories of Linux | 5 |
| Figure 6 - Usage of Web Servers for Websites..... | 6 |
| Figure 7 - Sample of SQL Injection..... | 9 |
| Figure 8 – A Sample HTTP Header Request..... | 10 |
| Figure 9 – A Sample HTTP Header Response with Set-Cookie | 10 |
| Figure 10 - Sample of Missing Access Control | 13 |
| Figure 11 - CSRF Attack Example | 17 |
| Figure 12 - Special Characters Used in Injection..... | 19 |
| Figure 13 - Surnames with Apostrophes and Script Tag Management | 20 |
| Figure 14 - HTML Source Code of Rich Text in Figure 13 | 20 |
| Figure 15 - Forcing Password to Change..... | 21 |
| Figure 16 - A Successful Man-in-the-Middle Attack | 22 |
| Figure 17 - POSTDATA Structure | 24 |
| Figure 18 - Invalid Login Error Message..... | 24 |
| Figure 19 - Script Installation | 25 |
| Figure 20 - Command to Run Our Script..... | 25 |
| Figure 21 - Broken Password and Created Files..... | 26 |
| Figure 22 - Access Denied Error Page..... | 28 |
| Figure 23 - A Sample HTTP Header from an Unsecured Server | 29 |

| | |
|---|----|
| Figure 24 - Error Message Related with Permissions..... | 31 |
| Figure 25 - URL with Session Key | 31 |
| Figure 26 - Confirmation Step for Deleting a User..... | 32 |
| Figure 27 - Codes behind the Figure 14..... | 32 |
| Figure 28 – Path to Data Directory | 37 |
| Figure 29 - Database Settings | 37 |
| Figure 30 - Admin Details | 38 |
| Figure 31 - A Sample Notifications Page | 42 |
| Figure 32 - VirtualBox Manager Window | 45 |
| Figure 33 - Settings Window | 46 |
| Figure 34 - Choosing Installation Medium | 47 |
| Figure 35 - Starting Script Installation..... | 51 |
| Figure 36 - Apache Works | 52 |

LIST OF ABBREVIATIONS

| | |
|--------|--|
| ACL | Access Control List |
| CD | Compact Disk |
| CSRF | Cross-Site Request Forgery |
| CSV | Comma Separated Values |
| DBMS | Database Management System |
| DHCP | Dynamic Host Configuration Protocol |
| DOM | Document Object Model |
| DVD | Digital Versatile Disk |
| FTP | File Transfer Protocol |
| GRUB | Grand Unified Bootloader |
| HTML | Hyper Text Markup Language |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Secure Hypertext Transfer Protocol |
| IIS | Internet Information Services |
| IP | Internet Protocol |
| ISP | Internet Service Provider |
| IT | Information Technology |
| LAMP | Linux, Apache, MySQL, PHP |
| LMS | Learning Management System |
| LVM | Logical Volume Manager |
| MOOC | Massive Open Online Course |
| MOODLE | Modular Object-Oriented Dynamic Learning Environment |
| OS | Operating System |

| | |
|-------|---------------------------------------|
| OWASP | Open Web Application Security Project |
| PHP | Hypertext Preprocessor |
| SSL | Secure Socket Layer |
| SQL | Structured Query Language |
| URL | Uniform Resource Locator |
| WWW | World Wide Web |
| XSS | Cross-Site Scripting |

Chapter 1

INTRODUCTION

Distance Learning becomes more and more important. Academically and financially strong universities all over the world form consortia for Massive Open Online Course (MOOC) delivery like edX [1] or Coursera [2]. The basic idea here is to change the economy of course delivery as pointed out in [3]. On the other hand Learning Management Systems become inevitable in secondary and higher education and schools as well as universities have to provide digital content as support to their in class teachings. There are various LMS's available on the market, custom made, commercial, and open source. When we check the user statistics of LMS systems, we will find [4] that the top 5 used LMS systems are Moodle with about 60 million users, SumTotal with about 32 million users, Blackboard/WebCT with about 20 million users, Edmodo with about 10 million users, and Interactyx with about 10 million users. We can easily identify that Moodle has the highest market penetration in terms of number of users. This makes Moodle especially interesting for various attack methods. There is unfortunately very little information available about security breach problems of commercial LMSs like Blackboard/WebCT. Therefore we will concentrate in this thesis on the security of Moodle (version 2.4), or on how various attack methods can be prevented.

Moodle claims that the number of users is over 70 million in 2013, as we can see in Figure 1 below.

| | |
|------------------|-------------|
| Registered sites | 86,289 |
| Countries | 237 |
| Courses | 7,802,353 |
| Users | 73,084,114 |
| Teachers | 1,297,013 |
| Enrolments | 76,243,719 |
| Forum posts | 129,985,398 |
| Resources | 69,503,576 |
| Quiz questions | 196,459,208 |

We perform regular bulk checking of sites to make sure they still exist, so occasionally you may see reductions in the count

Figure 1 - Moodle Statistics

More statistics on Moodle can be checked from [5]. Moodle is developed using the PHP as the server side programming language. In Figure 2 we can see the statistics of the different server side programming languages used on the websites [6].

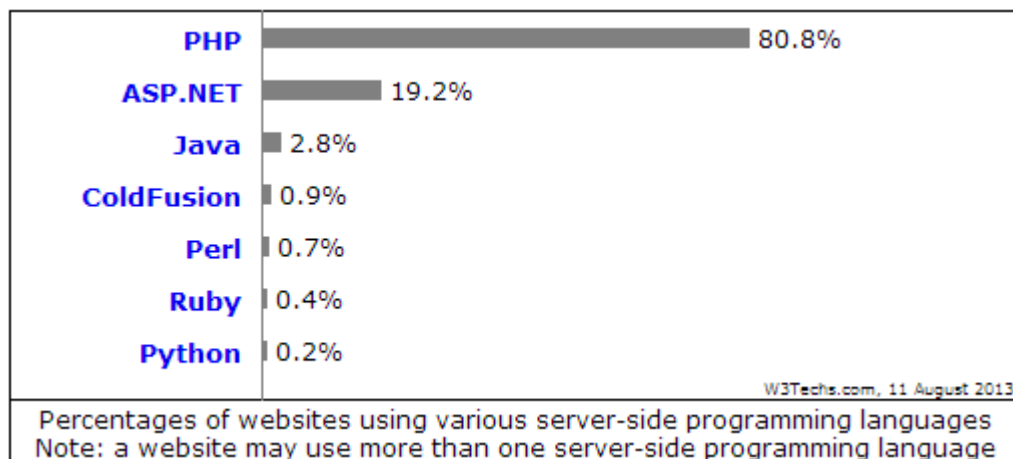


Figure 2 – Usage of Server-Side Programming Languages for Websites

As we are interested in the security of the Moodle LMS, we cannot consider the software isolated from the environment it is installed on. We can consider Moodle as

a part of an IT system. Without doubt the protection of any IT system is very important for various reasons, e.g. the service the IT system is offering, privacy of data, consistency of data, etc. Of course, IT systems have a public interface over the internet, but are not as secure as a bank's safe. Anybody can attempt to access, if they know their IP or URL addresses. Anyhow, we can secure these systems by applying some strict rules on them. These rules affect the configuration of the components of the server and the web applications used for the interaction with the users. So, choosing these components is important to make the protection persistent. Therefore we will consider the systems widely used on the Internet. The reason to go for the market leaders for each component is the following. The highest number of attacks can always be observed for the market leaders, and therefore security vulnerabilities are publicized very early, and fixes for these vulnerabilities become available very soon. Security by obscurity is not the choice of our security concept. We will only use well-known components that are widely used in the market. In the following we will present the statistics of the software components and use these for the decision of the components of our sample system. The statistics used for the decision of the components generated is based on some criteria which can be found in [7].

1.1 Server OS: Debian 7.1 Wheezy

The most important component of a server is its operating system. As we can see in Figure 3 around 2/3 of the servers run a UNIX based operating system whereas only around 1/3 run a Windows Server operating system. Other OS's are negligible [8].

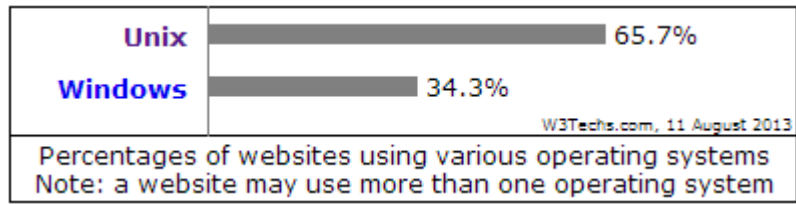


Figure 3 - Usage of OS for Websites

There are various UNIX based operating systems. Over the years open source Linux became the most important UNIX derivative on the market as we can see in Figure 4 [9].

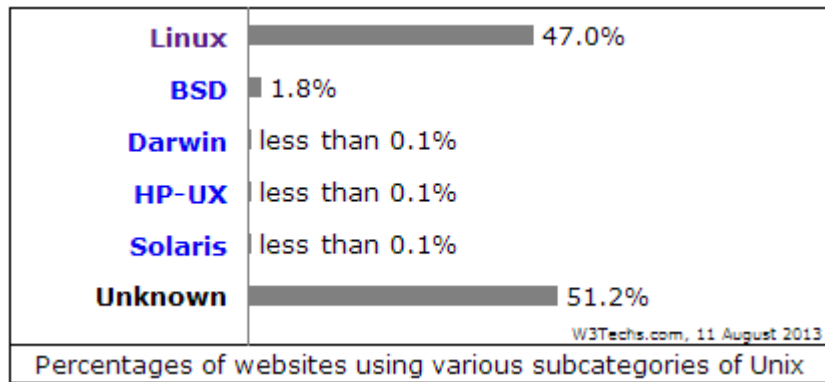


Figure 4 - Usage of Subcategories of UNIX

The distribution of usage of the Linux installations is presented in Figure 5 [10]. With 32.6% Debian is leading the table of the most used Linux distribution.

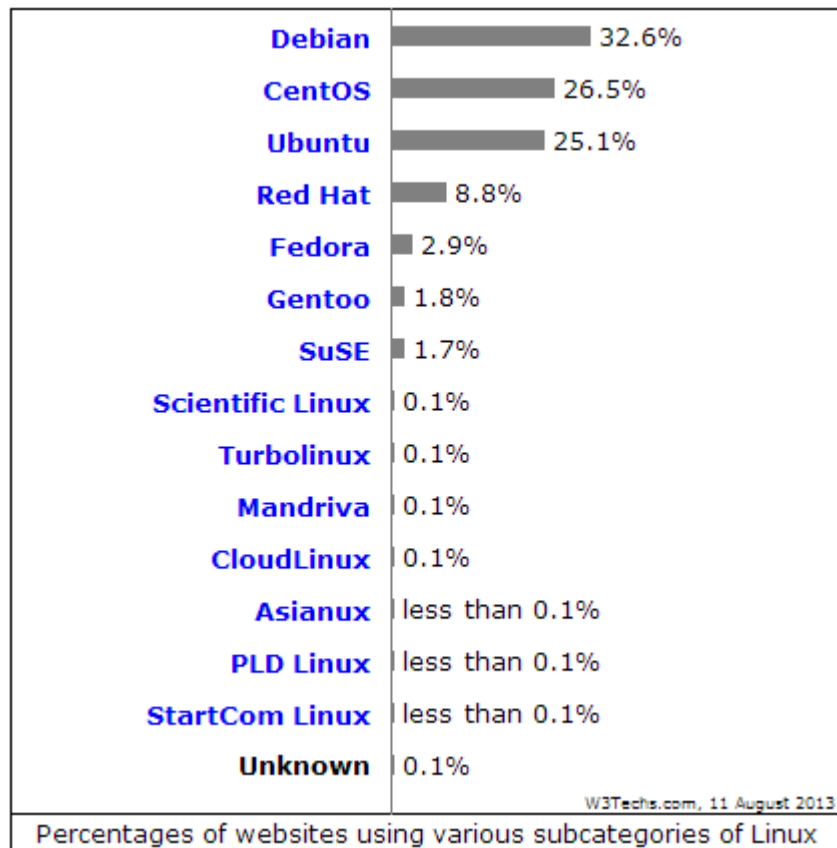


Figure 5 - Usage of Subcategories of Linux

1.2 Web Server: Apache2

Another important component is the web server. Statistics in Figure 6 visualizes the usage of them. Apache is in the front of all by far with 65.1%, which is about four times of nearest competitor Microsoft IIS as shown in Figure 6 [11].

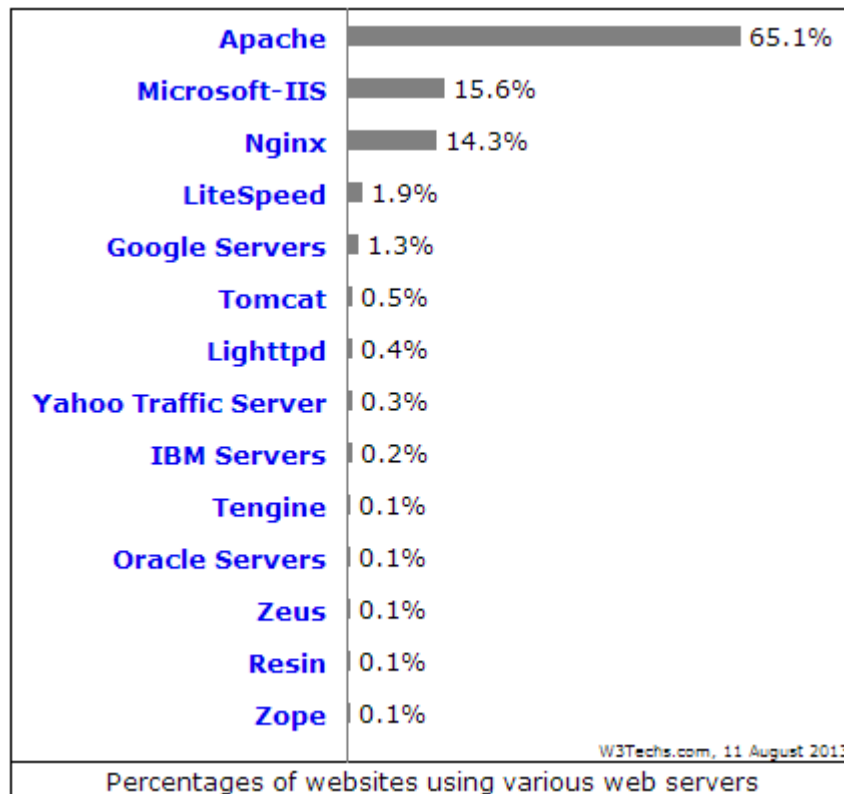


Figure 6 - Usage of Web Servers for Websites

1.3 DBMS: MySQL

Unfortunately, I did not find any statistics for DBMS usage, but it is well-known that PHP is closely tied to MySQL. Thus, you can take Figure 2 as its statistics.

1.4 Conclusion

As a result of these thoughts and statistics, I found a tutorial which combines all these things together, without any troublesome progress. The tutorial uses ISP software called ISPCConfig 3 [12]. You can manage all of the things above by using ISPCConfig 3.

Based on this I will discuss in Chapter 2 Security Risks, discussing top ten security risks available in the literature and in the internet. In Chapter 3, the security risks of Moodle are discussed. Chapter 4 presents how Moodle should be installed properly

to avoid the security risks. Chapter 5 shows explicitly how the server installation should be conducted to avoid security breaches. Finally we conclude with a summary of all findings in this thesis, in Chapter 6.

Chapter 2

SECURITY RISKS

Higher awareness regarding the security risks of web applications is necessary. The analysis of all possible attack methods is not sufficient; therefore more generally the risks of web applications will be the driving motivation in the following. In this manner, I chose the project *OWASP Top Ten*. This project has produced a book called “OWASP Top 10 - 2013” [13]. The ideas presented in [13] are used as a guide in this thesis, where the listed security risks are discussed in more details compared to the source.

2.1 Injection

Injection is also an attack type, where an attacker injects his code into a web application and sends data using the same way of submitting of usual data (username, email address, etc.) to the web application via a web browser. If the web application is vulnerable to injections, then the web application executes attacker’s code. With the execution of the injected code, all data accessible by the web application can be stolen, modified, or deleted. In some cases, this can result in data loss, data corruption, or denial of access.

They are often found in queries (like SQL, Xpath, NoSQL, etc.), OS commands, XML parsers and SMTP headers.

2.1.1 Sample Scenario: SQL Injection

```
1 <?php
2 //...
3 $query = "SELECT * FROM users WHERE username='".$user_name.'";
4 //...
5 ?>
```

Figure 7 - Sample of SQL Injection

Assume that, we have a web application that is written with PHP and the SQL query in the original code as in Figure 7. We have a variable called *user_name* and its value comes from an HTML form filled by a user.

An attacker can easily modify this variable as *' or '1'='1* using the form field related with the username. If there is no filter operation for the entered data, then our SQL command acts as *SELECT * FROM users WHERE username="" or '1'='1* and this query will return all of users' data.

2.2 Broken Authentication and Session Management

All web applications need such systems for user authentication to track the users via sessions through the web application. These systems are generally custom made for each web application. As a standard process is not used, developers cannot prevent flaws in the authentication and session management system of the web application (logout, password management, remember me, secret question, timeouts, etc.).

Generally, privileged accounts like administrator or root accounts are targeted. If an attacker succeeds to hijack a user's session, the attacker gets the same user rights and can act as the user. An anonymous external attacker or a user who wants to get access of another account can use this flaw. Therefore this flaw results in the risk of broken authentication and session management.

2.2.1 Sample Scenario: Session Hijacking

Let us assume that the user is in a public place and connected to the Internet via a hotspot like many others including also our attacker. The attacker is sniffing the network that the user is connected to. So the attacker is able to monitor all network traffics between the user and the server executing the users' requests.

After the user logs in to the web application, the server will send back a session id that makes the user unique on this server. The attacker will catch this session id from your HTTP header response (first time) from the server or other parts (URL, HTTP header request, etc.) depending on the web application's configuration. Assume that this application uses cookies. A sample HTTP header request is shown in Figure 8 below.

| Request Header Name | Request Header Value |
|---------------------|--|
| Host | fas.emu.edu.tr |
| User-Agent | Mozilla/5.0 (Windows NT 6.1; WOW64; rv:22.0) Gecko/20100101 Firefox/22.0 |
| Accept | text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 |
| Accept-Language | en-US,en;q=0.5 |
| Accept-Encoding | gzip, deflate |
| Connection | keep-alive |

Figure 8 – A Sample HTTP Header Request

| Response Header Name | Response Header Value |
|----------------------|---|
| Status | OK - 200 |
| Content-Type | text/html; charset=UTF-8 |
| Content-Encoding | gzip |
| Vary | Accept-Encoding |
| Server | Microsoft-IIS/7.5 |
| X-Powered-By | PHP/5.3.14, ASP.NET |
| Set-Cookie | CAKEPHP=16f7fgujqr7ll6hgdme0a75i67; expires=Sun, 04-Aug-2013 15:44:53 GMT; path=... |
| Date | Sun, 04 Aug 2013 11:44:53 GMT |
| Content-Length | 3730 |

Figure 9 – A Sample HTTP Header Response with Set-Cookie

The corresponding HTTP header response shown in Figure 9 also includes the *Set-Cookie* field with the session key, and its expiration date. Now the attacker knows your session information and can hijack easily the users session. Using a cookie editor tool, the attacker simply assigns the new cookie that includes the user's session id and acts as the user until the user logs out from the web application. If the user does not log out from the web application, this cookie is valid until its expiration date and time in the *Set-Cookie* field.

2.3 Cross-Site Scripting (XSS)

Another risk that appears is that Cross-Site Scripting allows attackers to execute some malicious code on the victim's browser. Cross-site scripting facilitates hijacking of user sessions, deface websites, insert hostile content, redirect users, etc. by an attacker.

External users, internal users and administrators can supply this type of untrusted data including malicious code. Because of this, the web application needs proper validation for the data coming from outside and developers and administrators need to be aware of this risk while adding some data/content to the system.

There are three known types of cross-site scripting flaws: Stored, reflected and DOM based.

2.3.1 Stored Cross-Site Scripting

If dangerous script code is stored as ordinary content in the database used by the web application, then this is called stored XSS. This malicious script code exists within contents (forum posts, comments, etc.) and executes every time during the related page content loaded by the user.

2.3.2 Reflected Cross-Site Scripting

In this type of XSS, the malicious script code is reflected to the server. The user is not aware about the XSS and continues as normal. E.g. the user just clicks on a link from an email, or submits a special crafted form, and then the injected script code reaches the XSS vulnerable web application, which reflects our malicious script code back to the user's browser and the user's browser executes this code.

2.3.3 DOM Based Cross-Site Scripting

This XSS type, in contrast to both reflected and stored XSS attacks, makes use of a flaw in web application, where the client side script modifies the DOM environment the malicious script code is injected, because in this method the attacker uses an original client side script that is part of the web application. This original client side script modifies DOM environment in that web application and our malicious script code injected to the web application via original client side script. After that it is executed by the user's browser.

2.3.4 Sample Scenario: Stored XSS

Assume that an attacker hijacks the user's session information via sniffing the network. Then, he can access the user's profile page to find a vulnerable field. In case the attacker finds a vulnerable field the attacker injects his malicious code into this field and submits the form. The whole data will be saved into the database of the web application.

These profile data will be presented by the user's profile page, so that the attacker will wait for another authenticated user to visit the user's profile page. Every request to the user's profile page will also execute the attacker's malicious script code.

2.4 Insecure Direct Object References

Developers generally use references to change an object's name or key. This is easy to implement, but many applications do not verify if the user is authorized or not for this object.

The attacker can easily guess other values for the object and steal all the data related to this object. If the attacker is an authenticated user, then this kind of attack is easier to perform.

2.4.1 Sample Scenario: Missing Access Control

Assume that we have a vulnerable web application and the user has already opened the link: <http://www.sample.com/user/list.php?role=user>. This file (list.php) could be like in Figure 10.

```
1 <?php
2 if (is_authenticated()) {
3     # code...
4
5     if ($_GET['role'] == "admin") {
6         create_editable_list();
7     } else
8         if ($_GET['role'] == "user") {
9             create_list();
10        } else {
11            redirect("error.php", $_GET['role'] . " is not a valid role!");
12        }
13
14    # code...
15 } else {
16     redirect("login.php", "You should login to system to see user list!");
17 }
18 ?>
```

Figure 10 - Sample of Missing Access Control

The attacker can guess other values for the *role* variable. As we see in the code above in Figure 10, the web application just checks if the user is authenticated or not. An authenticated user can simply modify the *role* variable in the URL. If the attacker

modifies the link in the form *http://www.sample.com/user/list.php?role=admin*, then the access to *create_editable_list()* function is granted, which should normally not be available.

Probably, this will make the attacker edit users such as deleting all, creating new one, etc.

2.5 Security Misconfiguration

This risk is related to every server application, like the web server, database, framework, and application server. These applications need to be updated regularly and configured properly to secure the server. Everything not needed, e.g. services, ports, etc. should be removed or disabled. Cooperation is needed between developers and system administrators for guaranteeing entire stack configured properly. Otherwise, this will conclude with a flaw.

Some automated tools may help for detecting missing patches, misconfigurations, default accounts, unnecessary services, etc. In general, this should be automated to ensure high protection and reduce the workload of the administrator.

2.5.1 Sample Scenario: Server Misconfiguration

Assume that directory listing is not disabled on your server. An attacker browses simply directory of your web application and steals your codes. Of course, he reaches your configuration files too. Learning database credentials may occur with a huge data compromise.

2.6 Sensitive Data Exposure

This is not a very common risk, but we need to secure sensitive data like personal information, credit card numbers, user credentials, etc. Danger is not limited to the

system borders. Also backups of these sensitive data are vulnerable. The easiest approach to secure sensitive data is to store the data encrypted, but this may not be sufficient sometimes for highly sensitive data.

Usage of weak key generation and management, and weak algorithm for ciphering is endangering these sensitive data. Because, breaking old/weak algorithm or weak keys is not a big issue for attackers nowadays. They can simply break and read all sensitive data.

2.6.1 Sample Scenario: Stealing of a Backup File

Assume regular backups of the web application are taken and stored in directory named *backups*. This folder exists in the directory of your web application.

If the directory listing for this folder is not disabled explicitly the attacker can easily create a directory tree of the web application using a brute force method tool. Now, the attacker has access to most of the files and folders including *backups*.

If there is no other protection for these backup files, the attacker can directly read content of the sensitive data from the backup file.

Proposal for the solution: Although this seems to be very practical and straight forward, the storage of backups should be independent from the web server and should be kept on a separate storage location.

2.7 Missing Function Level Access Control

Web applications may not always control the function level access and the authorization of the user. After the log in, an authorized user can simply change the

URL or a variable to reach a private function. This function operates without authorization of the user that is not normally having access to its usage.

2.7.1 Sample Scenario: Missing Function Level Access Control

This risk is close to 2.4 Insecure Direct Object References. The only difference is that this occurs in the function level, so that we continue from the sample scenario of 2.4.1 Sample Scenario: Missing Access Control.

If *create_editable_list()* function has also an access control mechanism implemented, the attacker will not be able to access this function. As a result of this, he will be redirected to an error page.

2.8 Cross-Site Request Forgery (CSRF)

This risk is caused by the attack method CSRF. In this method, the attacker creates a forged link and tries to send this link to the web application via an authenticated user. In order to succeed, the attacker may hide this link into image tag in the web application or can try to apply a XSS attack.

In this way the attacker can manage to access some privileged function operation via an authenticated admin user. Here, the web application trusts the data that comes with an authenticated user session and executes corresponding operation. The user will not realize this attack immediately, because he is just loading a page in the web application.

2.8.1 Sample Scenario: CSRF Attack

Assume that there is a web application available on: *http://www.sample.com* allowing comments to guests and publishes instantly their comments. Also, the

attacker knows the web application's link mechanism and includes that link inside an image tag on a posted comment as in Figure 11.

```
1 
```

Figure 11 - CSRF Attack Example

When an authenticated admin user loads the related page after the publication of this comment, the admin's browser will try to call this image and will request the information in the link. This request is taken by the web application and executed, because an admin user who is normally allowed to execute such an operation executes it, the user with the id *123* will be deleted.

2.9 Using Components with Known Vulnerabilities

A server is a composition of many components interacting with each other. Generally, each component is developed independently by different communities. Furthermore, these base components run with full privileges. A vulnerability of such a component can easily become critical. An attacker can manage to exploit vulnerability for a component to get full control over the component and the system. In this case sensitive data can be accessed and corrupted, or the attacker can start an attempt to take over the server.

Fixing such vulnerability depends on the community developing and supporting this component. If the administrator of the system is aware of such vulnerabilities, the administrator has to scan regularly for updates and bug fixes of the component and take more frequent backups of the data during this period. It is advisable to register the important component's developer email lists to receive the security updates as soon as they are available.

2.9.1 Sample Scenario: MySQL - Bug #64884

This MySQL bug [14] allows an attacker to bypass authentication in specific environments by repeatedly trying to authenticate with the same incorrect password. In this case this bug could be exploited if the bug fix is not applied and an attacker can get full control over the MySQL server.

2.10 Invalidated Redirects and Forwards

Especially on discussion platforms like forums, social networks, blogs, etc. users interact with each other and share information. In some cases they can share links to other web sites. This is called a redirect operation.

Another similar type operation is called forward operation. In this case, these redirections occur inside the web application and users move from one web page to another web page of the same web application.

In both cases, proper validation is needed for redirects and forwards. Otherwise, users can be redirected to phishing or forwarded to the unauthorized sites.

2.10.1 Sample Scenario: Redirects

Assume we have a web application and it has redirect mechanism to other web sites. A user clicks on a link (EMU Department of Mathematics) from our web application and redirect mechanism catches this request as *http://www.sample.com/redirect.php?url=www.evil.com*.

Now we are restricted with the capabilities of our *redirect.php* file. If it has white/black list check capability, then it may decide to redirect or not. Of course, we assumed there is no such capability and the user will be redirected to the phishing site.

Chapter 3

ASSESSMENT OF SECURITY RISKS IN MOODLE

In the previous chapter the security risks for web applications in general were discussed on the base of the top ten security risks of 2013 [13]. In the following Moodle will be assessed according to these security risks. Basic attacks for the risks discussed before will be applied to a Moodle test system and the way Moodle handles these security risks will be discussed.

3.1 Injection

3.1.1 General Aspects

Methods that are used to apply injections need some special characters like in Figure 12.

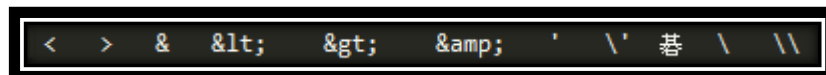



Figure 12 - Special Characters Used in Injection

These special characters could be used for injection via a form field or an URL parameter. Moodle should handle such kind of data without any loss. Since, some form fields accepts rich texts in Moodle, it must be able to keep whole data without affecting its structure.

3.1.2 How Moodle Avoids the Risk

Moodle employs a proper validation for inputs. It takes out script tags and avoids apostrophe sign character in some special cases like surnames in Figure 13.

Eugene Gladstone O'Neill



Eugene Gladstone O'Neill (October 16, 1888 – November 27, 1953) was an [Irish American](#) playwright and Nobel laureate in Literature.

Normal style tags like stylize bold and links are shown above.

`<script>alert('This will display as it is with script tags.');`

Country: Cyprus

City/town: Famagusta - `alert('This will display without script tags.');`

Course profiles: [Test Course 1](#)

First access: Wednesday, 7 August 2013, 4:40 PM (1 day)

Last access: Thursday, 8 August 2013, 5:33 PM (43 secs)

Figure 13 - Surnames with Apostrophes and Script Tag Management

Moodle inserts backslash in front of the apostrophes sign like “O\Neill” and escapes from the risk before running SQL command. This is done by concatenating strings. All input data that is properly filtered are passed to an array and concatenated according to the SQL query schema.

Other than rich text fields, tags like `<script></script>` are removed automatically. In rich text fields, Moodle handles such tags to style the text or protect their structures as it is in the Figure 1. You can check third line in the rich text area for script tag visualization and *City/town* part to realize script tags that has been removed by Moodle.

Figure 14 shows how Moodle handles script tags. Moodle converts “<” and “>” characters to the corresponding HTML entities respectively, “<” and “>,” which replaces them with a visual element.

```
HTML source editor  Word wrap
<p><strong>Eugene Gladstone O'Neill</strong> (October 16, 1888 – November 27, 1953) was an <a
title="Irish American" href="http://en.wikipedia.org/wiki/Irish_American">Irish
American</a> playwright and Nobel laureate in Literature.</p>
<p>Normal style tags like stylize bold and links are shown above.</p>
<p><script>alert('This will display as it is with script tags.');

```

Figure 14 - HTML Source Code of Rich Text in Figure 13

3.2 Broken Authentication and Session Management

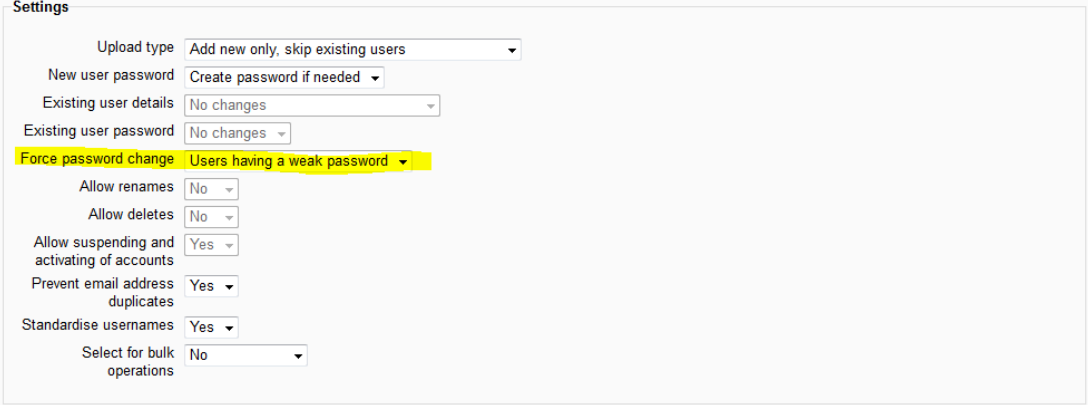
3.2.1 General Aspects

Moodle has a lot of options to authenticate a user, but only default authentication method will be considered in the following. This allows an authorized user like administrator to add them and they will be logged into system with their predefined usernames and passwords.

Another issue arises when SSL is not used during login process or entire session period. Otherwise, some attack methods like man-in-the-middle, session fixation, etc. can be applied easily to the targeted site.

Breaking authentication using brute force is also an important and easy method conducted by attackers. In this method, all possible password combinations are tried until the attacker logs in successfully.

3.2.2 How Moodle Avoids the Risk



The image shows a screenshot of the Moodle user settings page. The 'Settings' section is visible, containing several dropdown menus. The 'Force password change' option is highlighted in yellow and is set to 'Users having a weak password'. Other settings include 'Upload type' (Add new only, skip existing users), 'New user password' (Create password if needed), 'Existing user details' (No changes), 'Existing user password' (No changes), 'Allow renames' (No), 'Allow deletes' (No), 'Allow suspending and activating of accounts' (Yes), 'Prevent email address duplicates' (Yes), 'Standardise usernames' (Yes), and 'Select for bulk operations' (No).

| Setting | Value |
|---|-----------------------------------|
| Upload type | Add new only, skip existing users |
| New user password | Create password if needed |
| Existing user details | No changes |
| Existing user password | No changes |
| Force password change | Users having a weak password |
| Allow renames | No |
| Allow deletes | No |
| Allow suspending and activating of accounts | Yes |
| Prevent email address duplicates | Yes |
| Standardise usernames | Yes |
| Select for bulk operations | No |

Figure 15 - Forcing Password to Change

While an authorized user will upload a CSV file that keeps data of other users, then Moodle offers some extra options to enhance password security like in the Figure 15. The authorized user can force to change passwords for users who have a weak password or all of them. Password changes will be prompted in their first logins.

It is not important to set strong passwords, if you are not enabled SSL for your Moodle site. Unfortunately, Moodle uses SSL only for login page and SSL must be enabled explicitly. Thus, your users' credentials are secured, but their session information still available to be hijacked by an attacker.

If SSL is not activated, then your users' credentials are not secured. I have performed man-in-the-middle attack to get an admin user credentials. Figure 16 shows that a successful attack with admin credentials.

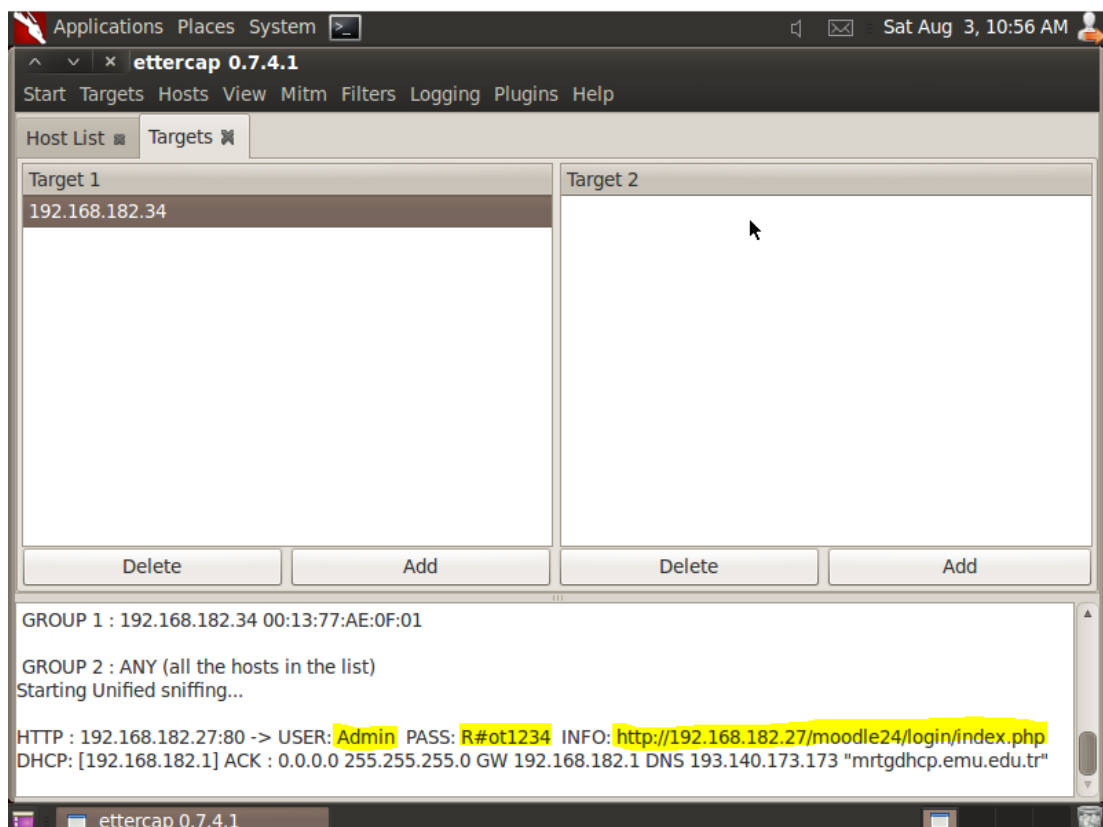


Figure 16 - A Successful Man-in-the-Middle Attack

Ettercap 0.7.4.1 is used to perform this attack on the Linux system (Back Track 5 R3). The scenario is very simple to only show how easy it is. I just set *ettercap* up to listen my IP address which is 192.168.182.34 and log in directly to Moodle site as an admin user. It caught successfully my username and password (See Appendix A: Ettercap Attack Logs).

Brute Force Method: Moodle can block a user, if he has reached the attempt limit for login. This system is based on session key and needs cookies, so that a simple script can surpass such a protection. Additionally, Moodle has options for informing bad login attempts and their default is set to inform nobody. This means no limit for login attempts. I also tried applying a tutorial from computersecuritystudent.com [15].

Using Tamper Data for Firefox web browser, I caught “*POSTDATA*” structure by using random values *test* for the username and *12345* for the password and also I noted the “*Referer*” part (Figure 17). Another need is an error message, if the login failed to understand password is wrong (Figure 18).

| Request Header Name | Request Header Value |
|---------------------|--|
| Host | 192.168.182.27 |
| User-Agent | Mozilla/5.0 (Windows NT 6.1; WOW64; rv:22.0) Gecko/... |
| Accept | text/html,application/xhtml+xml,application/xml;q=0... |
| Accept-Language | en-US,en;q=0.5 |
| Accept-Encoding | gzip, deflate |
| Referer | http://192.168.182.27/moodle24/login/index.php |
| Cookie | MoodleSession=lgopamklb7ok093tgncigf8vh0 |
| Connection | keep-alive |
| Content-Type | application/x-www-form-urlencoded |
| Content-Length | 28 |
| POSTDATA | username=test&password=12345 |

Figure 17 - POSTDATA Structure

Returning to this web site?

Login here using your username and password
(Cookies must be enabled in your browser) ?

Invalid login, please try again

Username

Password

Remember username

[Forgotten your username or password?](#)

Some courses may allow guest access

Figure 18 - Invalid Login Error Message

After that, I set up our *crack web forms* script as in Figure 19.

```
root@bt: /pentest/passwords/cwf
File Edit View Terminal Help
root@bt:/pentest/passwords# mkdir cwf
root@bt:/pentest/passwords# cd cwf
root@bt:/pentest/passwords/cwf# wget http://www.computersecuritystudent.com/SECURITY_TOOLS/MUTILLIDAE/MUTILLIDAE_2511/lesson4/cwf.v2.tar.gz
--2013-08-03 12:00:57-- http://www.computersecuritystudent.com/SECURITY_TOOLS/MUTILLIDAE/MUTILLIDAE_2511/lesson4/cwf.v2.tar.gz
Resolving www.computersecuritystudent.com... 75.11.117.94
Connecting to www.computersecuritystudent.com|75.11.117.94|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15977 (16K) [application/x-gzip]
Saving to: `cwf.v2.tar.gz'

100%[=====>] 15,977 --.-K/s in 0.001s
2013-08-03 12:00:58 (28.6 MB/s) - `cwf.v2.tar.gz' saved [15977/15977]

root@bt:/pentest/passwords/cwf# tar -xzf cwf.v2.tar.gz
root@bt:/pentest/passwords/cwf# ls
crack_web_form.pl  cwf.v2.tar.gz  password.txt
root@bt:/pentest/passwords/cwf#
```

Figure 19 - Script Installation

Generally, this kind of attack performs for admin users, but I did not want to modify *password.txt* file that comes with our compressed script file. And I tried to break *user1*'s password. Then, I wrote code to start attack with known data (yellow texts in Figure 20).

```
root@bt: /pentest/passwords/cwf
File Edit View Terminal Help
root@bt:/pentest/passwords/cwf# ./crack_web_form.pl -U user1 -http "http://192.168.182.27/moodle24/login/index.php" -data "username=USERNAME&password=PASSWORD" -F "Invalid login, please try again"
```

Figure 20 - Command to Run Our Script

After about 5 minutes, it broke *user1*'s password at attempt: 432 and created two files; cookie of this session and logs of this attack operation called respectively "*crack_cookies.txt*" and "*crack-log.txt*". These are highlighted with yellow in Figure 21.


```
root@bt: /pentest/passwords/cwf
File Edit View Terminal Help
[Attempt]: 427 [Username]: user1 [Password]: matthew [Status]: Failed
-----
[Trying Password]: miller
[Attempt]: 428 [Username]: user1 [Password]: miller [Status]: Failed
-----
[Trying Password]: ou812
[Attempt]: 429 [Username]: user1 [Password]: ou812 [Status]: Failed
-----
[Trying Password]: tiger
[Attempt]: 430 [Username]: user1 [Password]: tiger [Status]: Failed
-----
[Trying Password]: trustno1
[Attempt]: 431 [Username]: user1 [Password]: trustno1 [Status]: Failed
-----
[Trying Password]: 12345678
[Attempt]: 432 [Username]: user1 [Password]: 12345678 [Status]: Successful [SESSION]

root@bt: /pentest/passwords/cwf# ls
crack_cookies.txt  crack-log.txt  crack_web_form.pl  cwf.v2.tar.gz  password.txt
root@bt: /pentest/passwords/cwf#
```

Figure 21 - Broken Password and Created Files

3.3 Cross-Site Scripting (XSS)

3.3.1 General Aspects

This is a kind of injection but specifically stated as another security risk, because it is widely found in web applications. Any input fields in forms or URLs, are enough to perform this attack. Especially, we should care about rich text editors, because it must accept HTML tags as in Figure 14. Generally, this risk endangers our session information and other sensitive data.

That kind of dangerous scripts can be executed via Flash and Java Applets too, so that we do not always need user's browser to perform this attack.

3.3.2 How Moodle Avoids the Risk

We covered this before in the "Injection" part (See 3.1.2 How Moodle Avoids the Risk) in general, but we did not mention other ways of executing this kind of dangerous scripts within Moodle.

Moodle aims to provide more interactive educational contents to their users who are students. Thus, teachers and admin users can add interactive contents from a public place on the WWW. This triggers a security risk and they should be more careful while getting this kind of interactive content.

3.4 Insecure Direct Object References

3.4.1 General Aspects

Simply, changing values that is obvious in the URL or hidden in the post data may cause a privileged function access by an ACL bug. Here, ACL gets importance. Since, Moodle has different types of users (student, teacher, admin, etc.) powerful mechanism is needed for Moodle.

3.4.2 How Moodle Avoids the Risk

Moodle has a very flexible role system. This is able to create new roles with assigning necessary permissions and edit existing roles to add/remove permissions. I think some definitions [16] from Moodle's document site will make its ACL system more obvious for you:

“A context is a "space" in the Moodle, such as courses, activity modules, blocks etc.

A role is an identifier of the user's status in some context. For example: Teacher, Student and Forum moderator are examples of roles.

A capability is a description of some particular Moodle feature. Capabilities are associated with roles. For example, mod/forum:replypost is a capability.

Permission is some value that is assigned for a capability for a particular role. For example, allow or prevent.”

Under the circumstances of these definitions, Moodle handles user operations properly.

I performed some kind of basic penetration tests using a student role to reach some admin authorization needed pages and took “*Access denied*” error message (Figure 22).

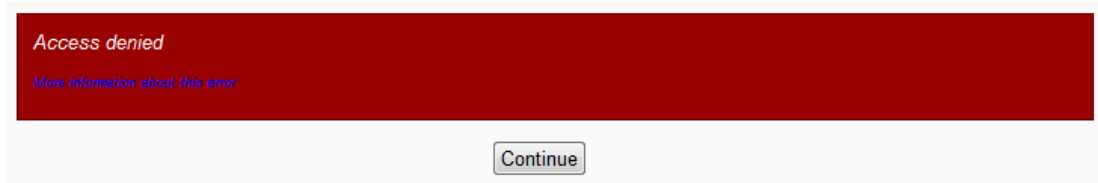


Figure 22 - Access Denied Error Page

Also, I tried to delete a user with and without given session key only for this operation and same error message displayed (Figure 22).

Moodle properly controls users’ access to any context or capability and checks permissions with respect to their roles.

3.5 Security Misconfiguration

3.5.1 General Aspects

This risk includes directory listing, folder permissions, HTTP headers, etc. Since server components are not configured properly, there will be some information leakage from this server. This leakage could be related to critical data (personal information), information about server components via HTTP headers (Figure 23), some specific code files, etc.

| Response Header Name | Response Header Value |
|----------------------|--|
| Status | See Other - 303 |
| Date | Fri, 09 Aug 2013 18:17:28 GMT |
| Server | Apache/2.2.22 (Debian) |
| X-Powered-By | PHP/5.4.4-14+deb7u3 |
| Expires | Thu, 19 Nov 1981 08:52:00 GMT |
| Cache-Control | no-store, no-cache, must-revalidate, post-check=0, pre-check=0 |
| Pragma | no-cache |
| Location | http://192.168.182.27/moodle24/login/index.php |
| Content-Language | en |
| Vary | Accept-Encoding |
| Content-Encoding | gzip |
| Content-Length | 398 |
| Keep-Alive | timeout=5, max=100 |
| Connection | Keep-Alive |
| Content-Type | text/html |

Figure 23 - A Sample HTTP Header from an Unsecured Server

3.5.2 How Moodle Avoids the Risk

Moodle do not have many options to prevent this risk. Basically, Moodle can hide error messages and it may send them to log files. Moodle warns administrators automatically if errors are set to be displayed. Last but not least, Moodle comes with a PHP info page and that page is only accessible by an administrator (See 3.4.2 How Moodle Avoids the Risk).

We must have components as secure as possible keeping them up to date. Also, we must encrypt all sensitive data, even their backup files to prevent information leakage.

In Chapter 5 a secure server installation guide from a software project [17] (See SERVER INSTALLATIONS) is proposed.

3.6 Sensitive Data Exposure

Some of the risks are very close to each other and interrelated to each other. Since, we called them risks; we can use same attack methods for our assessments. As a result of this, we are going in a cumulative way.

3.6.1 General Aspects

This risk is available with all attack methods that can help to reach sensitive data on your Moodle site. Thus, we can never say these sensitive data is under protection. It is important how to secure this kind of data.

The application of hashing is not important, because it is only a one way function and it can be broken using brute force method. We need some extra protection. Also, encryption is not enough without strong crypto key combinations.

3.6.2 How Moodle Avoids the Risk

Moodle provides extra protection method for passwords called password salting which encrypts passwords using your unique salt key.

Besides, Moodle data directory is located outside of your Moodle's installation directory and is forced to be located outside of your installation directory by Moodle installer.

3.7 Missing Function Level Access Control

3.7.1 General Aspects

Any web application should normally check the permission using an access function for the users' authorization. This will prevent some basic attack methods, but attackers never stop at this point and try to perform a CSRF attack (See 3.8 Cross-Site Request Forgery (CSRF)).

3.7.2 How Moodle Avoids the Risk

Moodle needs an extra session key for some specific operations (like deleting a user or course) in its URL which requests these operations. If session key is not found or does not match, then you see “Access denied” error page (See Figure 22). But, what happens if you try to access editing a course category page which does not need such session key variable. At this time, Moodle checks your permissions and returns an error page related to your permissions (Figure 24).

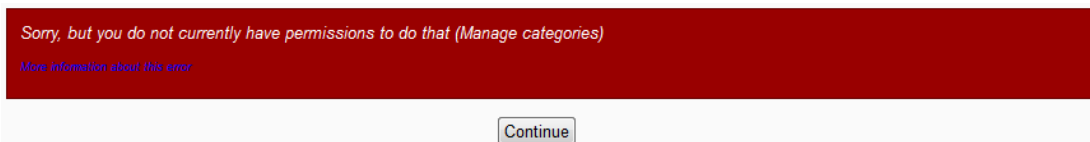


Figure 24 - Error Message Related with Permissions

3.8 Cross-Site Request Forgery (CSRF)

3.8.1 General Aspects

This is a way of avoiding authentication which tricks an authorized user (admin, teacher, etc.) to operate a privileged action like deleting a user. Attack is not identified immediately by an admin if there isn't any step like confirmation for critical operations.

3.8.2 How Moodle Avoids the Risk

Moodle is aware of the danger and takes necessary protection. When you login to your Moodle site, it also generates randomly a session key to use with such critical operations along the Moodle site (Figure 25).

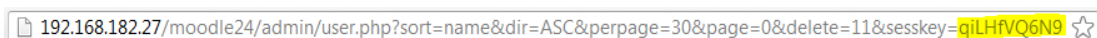


Figure 25 - URL with Session Key

Additionally, the attacker may sniff your network and get your unique session key to trick you to delete a user. Moodle needs an extra confirmation step for such critical operations to prevent this and deleting users by accident (Figure 26).

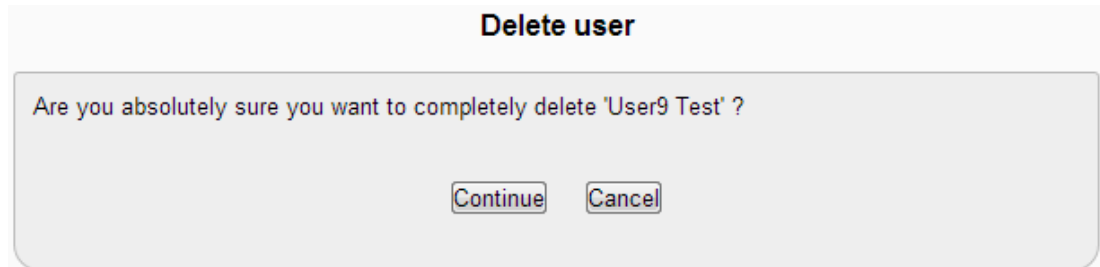


Figure 26 - Confirmation Step for Deleting a User

Nevertheless, I checked behind of this confirmation form (Figure 27). There may be another trick to bypass such protection.

```
▼<div role="main">
  <span id="maincontent"></span>
  <h2 class="main">Delete user</h2>
  ▼<div id="notice" class="box generalbox">
    <p>Are you absolutely sure you want to completely delete 'User9 Test' ?</p>
    ▼<div class="buttons">
      ▼<div class="singlebutton">
        ▼<form method="post" action="http://192.168.182.27/moodle24/admin/user.php">
          ▼<div>
            <input type="submit" value="Continue">
            <input type="hidden" name="sort" value="name">
            <input type="hidden" name="dir" value="ASC">
            <input type="hidden" name="perpage" value="30">
            <input type="hidden" name="page" value="0">
            <input type="hidden" name="delete" value="11">
            <input type="hidden" name="confirm" value="6512bd43d9caa6e02c990b0a82652dca">
            <input type="hidden" name="sesskey" value="qiLHFVQ6N9">
          </div>
        </form>
      </div>
      ▼<div class="singlebutton">
        ▼<form method="get" action="http://192.168.182.27/moodle24/admin/user.php">
          ▼<div>
            <input type="submit" value="Cancel">
            <input type="hidden" name="sort" value="name">
            <input type="hidden" name="dir" value="ASC">
            <input type="hidden" name="perpage" value="30">
            <input type="hidden" name="page" value="0">
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
</div>
</div>
</div>
```

Figure 27 - Codes behind the Figure 14

In Figure 27, I highlighted important parts. There is one more key called as “*confirm*” that is used to confirm before applying such a critical operation. This key is generated specifically for known operations and changes after each operation. Thus, the attacker needs to know this key to trick our authorized user. Despite everything, if the attacker performs successfully such an attack, he can only delete the “*user9 Test*” (Figure 26). The attacker needs “confirm” code of another user to delete his account.

3.9 Using Components with Known Vulnerabilities

3.9.1 General Aspects

If it is known that a vulnerable component is installed, and then immediate action has to be taken, otherwise, the attacker can damage your Moodle site, even if there are no security problems with Moodle itself.

There are periodic updates for specific server platforms or components (apache, MySQL, etc.). Some of them need special attention to get updated, because the server environment may be harmed by this update. These updates come with bug fixes or new features that you do not want to leave as it is. Your system administrator should take these into account and apply these updates or fixes without affecting server environment which is configured properly.

3.9.2 How Moodle Avoids the Risk

Moodle does not warn the Moodle and system administrators about server issues unless they are directly related to the Moodle application. The server side maintenance relies completely on the system administrator of the server. The server administrator has to be alert about security updates of the used components.

3.10 Invalidated Redirects and Forwards

3.10.1 General Aspects

Web applications like Moodle are based on sharing. This will let go some permissions like sharing another site's link inside the web application. However, we never guarantee this link is secure, because it may be linked to a phishing site by an attacker. We need to check all links with a trusted white/black list before redirection.

Also, there are forwards to the same web application. Invalidation of them may be resulted in an unauthorized access to a privileged feature.

3.10.2 How Moodle Avoids the Risk

Moodle does not have any control over redirections, so that authorized users like teachers and admins should continuously check their sites to remove malicious links manually.

Nor redirects, forwards are under control of Moodle. It is covered under the sections 3.4 Insecure Direct Object References and 3.7 Missing Function Level Access Control. Shortly, we can say that Moodle has a proper access control mechanism to protect privileged functions, objects, features, etc.

Chapter 4

SECURING MOODLE INSTALLATION

Moodle has an as easy as possible installation process. You can just click and leave all fields with their defaults and you need just to enter values for specific fields like passwords. This sounds good, but default values make our system more vulnerable. Thus, securing of our Moodle site starts from beginning of site setup.

Best way to describe such secure installation process is using a step-by-step guide and this is what will be elaborated in this chapter. The impact of each installation step will be illuminated as well.

4.1 Secure Server Installation

Yes, securing Moodle starts from its setup process, but it is impossible to say that the Moodle system is secure, unless we can guarantee that the server installation is secure. This arises to a very different subject, so that I did not go deeper. See **SERVER INSTALLATIONS**.

4.2 Secure Moodle Installation

4.2.1 Preparation

Since steps may differ with respect to your server environment, only general issues will be discussed. We assume that basic server administration skills are available. In the following the preferred tools, names, and settings are given in parenthesis.

Database Setup: There are different options like MySQL, PostgreSQL, Oracle, MS SQL, etc. Exemplarily we will use the MySQL database, as an open source database, which has no license fees. Use your available MySQL editing tool (e.g. phpMyAdmin) to create an empty database and called it as you wish (c1m24). Database collation is set to be “*utf8_unicode_ci*”.

Upload Your Moodle Installation Files: Use your FTP account to connect to your server and upload the installation Moodle files to your web root directory. Of course any other method of placing the installation files into the web root directory is also possible.

4.2.2 Start Your Installation

I used my own virtual server that is located on my computer, but my configuration allows assigning domain names via editing hosts file in windows. Thus, domain name, which I used, does not correspond to a real domain name.

Open your browser and browse to your domain name *http://www.yourdomain.com/* (*http://www.m24s.com/*) or *http://www.yourdomain.com/install.php* which is not important.

Choose a language: English

Confirm paths: The first two fields will be automatically set. The important part is to define data directory. This should be located outside of your web root directory (*./tmp/mdldata*). Default name for the data directory is “*moodldata*”. Rename it as you like (Figure 28).

| | |
|------------------|--|
| Web address | <input type="text" value="http://www.m24s.com"/> |
| Moodle directory | <input type="text" value="/var/www/clients/client1/web1/web"/> |
| Data directory | <input type="text" value="/var/www/clients/client1/web1/tmp/mdldata"/> |

Figure 28 – Path to Data Directory

If your default directory that is given by the installer is not writable, then contact your system administrator to locate your data folder outside of the web root with permissions 0777. Also, permissions of your Moodle installation folder is set to be 0755 and owner is set to be as root.

Choose database driver: Improved MySQL (native/mysqli)

Database settings: Enter your data and beware of the highlighted parts (Figure 29).

| | |
|-------------------|---|
| Database host | <input type="text" value="localhost"/> |
| Database name | <input type="text" value="c1m24"/> |
| Database user | <input type="text" value="c1db"/> |
| Database password | <input type="password" value="tw2KUhf0YqEC"/> |
| Tables prefix | <input type="text" value="emu_"/> |
| Unix socket | <input type="checkbox"/> |

Figure 29 - Database Settings

You must choose a strong password for your database. Although you do not need to use tables prefix name, Moodle has an option to vary this to prevent SQL injection attacks. Moodle default value is “mdl_” and you must replace it with another one.

Copyright notice: Click “Continue” button.

Server checks: You must see “Your server environment meets all minimum requirements.” at bottom of the page. If you have any problem with that page, contact your system administrator to solve it. Sometimes, there may not be all ok but some of them may not be needed for you and you can continue to installation if you like.

Installation progress: Database tables and other configurations have been completed. Click “*Continue*” button.

Admin account configuration: Enter your main administrator’s details.

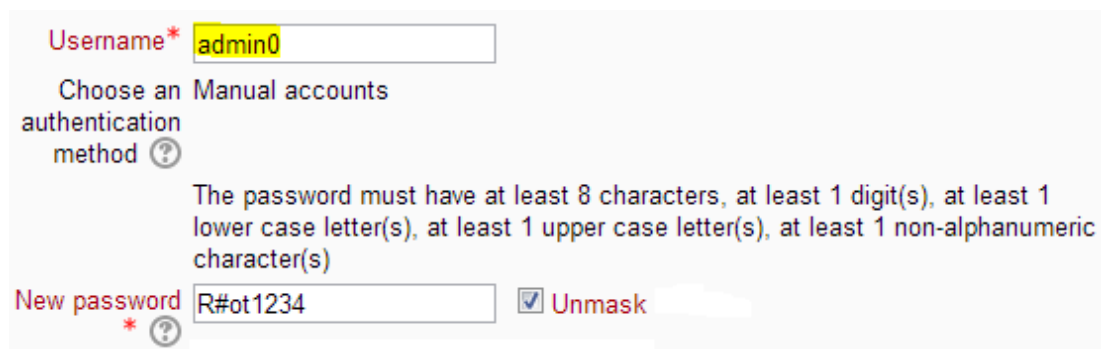
The image shows a screenshot of the Moodle installation 'Admin Details' configuration page. It features a light gray background with several input fields and labels. At the top, the 'Username*' field contains the text 'admin0'. Below it, there is a section for 'Choose an authentication method' with a dropdown menu currently set to 'Manual accounts'. A help icon (?) is next to this label. A password requirement message is displayed: 'The password must have at least 8 characters, at least 1 digit(s), at least 1 lower case letter(s), at least 1 upper case letter(s), at least 1 non-alphanumeric character(s)'. The 'New password*' field contains 'R#ot1234' and has an 'Unmask' checkbox checked. A help icon (?) is also present next to the 'New password*' label.

Figure 30 - Admin Details

Moodle has default username for admin that is admin too. You must change it with another username as you like (admin0). Also, you must assign a strong password, which is described above of the “*New password*” field in Figure 30. Enter information for other necessary fields and then click “*Update profile*” button where is located at bottom of the page.

Front page settings: Enter your site details.

Manage authentication: This is in the same page with front page settings and there is only one option here that is “*Self registration*”. Default value for it is “*Disable*”. If you allow user registrations with their email accounts you should limit by “*Allowed email domains*” setting to avoid spammers. I left it as it is.

After that you will be directed to the home page, logged in as your main admin user who is *admin0* in our case.

Cron Jobs: Cron jobs are periodically executed programs/scripts on the server for automated tasks. Thus, you must assign a cron job to run your cron job script for your Moodle site. URL of your cron job script is *http://site.yourmoodlesite.com/admin/cron.php*. If your server does not support for cron jobs, you can manually run it using your browser. Simply, locate your browser to URL of that script.

4.2.3 Configuration of Site Wide Settings

These settings are found in “*Site Administration*” part of “*Settings*” block which stays left.

Registration: You should register your Moodle site to *moodle.org* for getting security alerts before others.

Users/Permissions/User policies => Hide user fields: Protecting your users’ privacy is another security issue and you can select fields to hide the information from users that are not course teachers/admins.

Plugins/Filters/Manage filters: “*Email protection*” filter should be activated for preventing spammers to harvest them in a web page.

Security/Site policies => Cron password for remote access: Setup a password for blocking others to run cron jobs. Your cron job URL must be changed as *http://site.yourmoodlesite.com/admin/cron.php?password=your_password*.

Security/HTTP Security => Use HTTPS for logins: You must enable this feature to protect your users’ credentials. Otherwise, it is very easy to get users’ credentials for an attacker.

Security/Notifications => Display login failures to: This should be set at least for *Administrators*. They will be informed at such kind of situations.

Security/Notifications => Email login failures to: Choose your master admin to inform him via email.

Security/Anti-Virus: Contact your system administrator and take necessary information about setup your anti-virus protection for your uploaded files.

Server/Session handling => Timeout: Default is set to 2 hours, but you must decrease it less than 2 hours to protect your users’ sessions. My advice is set to 60 minutes, because less than this is not reasonable. Otherwise, your users will be annoyed.

Reports/Security overview: This is the last page that you visit to check if there is another issue for your Moodle site security. You can follow links to get informed about any issue that is listed or ask to solve your system administrator.

4.3 Suggestions

4.3.1 Change Your Default Theme

Your default theme can identify clearly your Moodle site version. You can use your own customized theme or select from listed ones from your site settings under Site administration/Appearance/Themes.

4.3.2 Take Regular Backups

Best option is set your web server to take full backups of your Moodle site. However, Moodle has backup options for *Front Page* and *Courses*. You should take these backups for enhancing your site security. If your full backups are infected, then you will need a fresh install. Thus, these backups will save your site.

4.3.3 Site Wide Settings

These settings are found in “*Site Administration*” part of “*Settings*” block which stays left.

Notifications: You may visit this page to observe if you missed something important to do. Additionally, you will find some other notifications like bad login attempts if you set to show you.

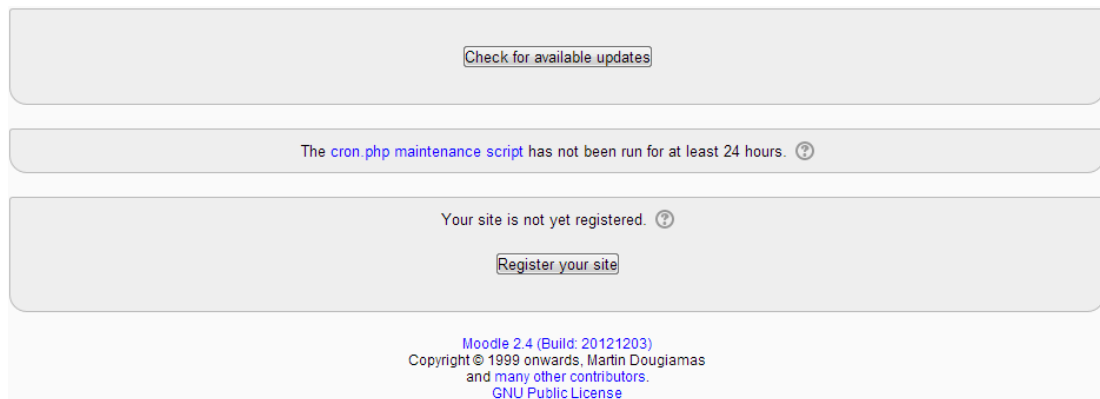


Figure 31 - A Sample Notifications Page

Advanced features => Web services: Do not enable this unless you really need to use it.

Plugins/Assignment plugins/File submissions => Maximum submission size: Default value is set to 1MB. Keep this value as low as possible.

Plugins/Authentication/Manage authentication => Self registration or Email-based self-registration: If you enabled these both, then you should activate reCAPTCHA for logins to protect your Moodle site against spammers. You can enable reCAPTCHA from *Settings* under *Site administration/Plugins/Authentication/Email-based self-registration*.

Plugins/Authentication/Manage authentication => Allowed/Denied email domains: Enabling self-registration is a problem, but assigning these will enhance your Moodle site security against spammers.

Plugins/Enrolments/Guest access: You may allow guest access to some courses. You can set to attempt a password which you distributed from your users to get access a course as guest.

Plugins/Filters/Manage filters: You may enable *Word censorship* filter and prevent your users to submit bad words.

Security/IP blocker: You can allow/block some IP addresses to protect your Moodle site known bad IP address or limit your Moodle side to specific IP addresses.

Security/Notifications => Threshold for email notifications: Default is set to 10. You may decrease it to 5.

Reports: It is an important part of your Moodle site, because you can see what happening/happened is on your site in details. Especially, “*Live logs*” is remarkable to have a look.

Reports/Spam cleaner: You can search specific certain strings along your Moodle site for all user profiles and delete those accounts which are obviously created by spammers.

Development/Experimental: Do not touch or use these experimental properties for your Moodle site stability.

Chapter 5

SERVER INSTALLATIONS

My aim is to secure the MOODLE environment completely. Thus, I tried to give a complete guide to install a server that can serve websites securely to the WWW. The most important part is to choose software components that will enhance security and I decided to choose widely used software packages over the world for frequent updates and to reach a very wide communities. Also, I provided an automated script that works on Debian Wheezy for basic LAMP (Linux, Apache, MySQL, and PHP) installation. I shortened it from “*Complete ISPConfig setup script for Debian 7*” [17] and rewrote with respect to “*Installing Apache2 with PHP5 and MySQL Support on Debian Wheezy*” [18], and “*The Perfect Server - Debian Wheezy (Apache2, BIND, Dovecot, ISPConfig 3)*” [19] articles. I tested both scripts on a virtual machine successfully. Installing Debian Wheezy is same for both scripts. After that, you may choose to install either my basic LAMP installation or “*Complete ISPConfig setup script for Debian 7*” that installs secure server.

5.1 Setting up Virtual Machine

I chose Virtual Box by Oracle to run and test my server installations. First, I will give you step by step guide how to configure Virtual Box to install Debian Wheezy. If you will set up a real server then you can skip this part.

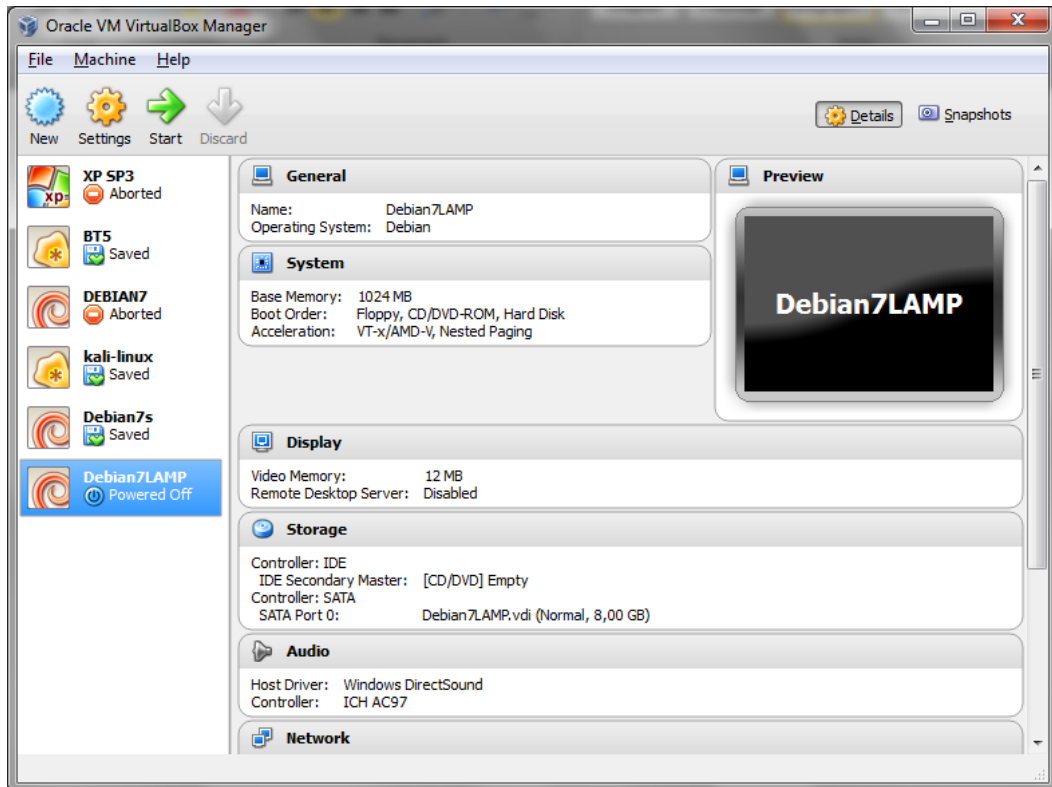


Figure 32 - VirtualBox Manager Window

Open VirtualBox and click “New” button (Figure 32).

Name and operating system: Give a name to your virtual machine (mine is Debian7LAMP) and if it includes the word “*Debian*” then others automatically adjusted. If not, please choose “*Type: Linux*” and “*Version: Debian*”.

Memory size: Adjust memory size depends on your system specifications.

Hard drive: Choose “*Create a virtual hard drive now*” and click “*Create*” button.

Hard drive file type: Choose hard drive file type. If you do not need to use it with other virtualization software, default is ok. Click “*Next*” button.

Storage on physical hard drive: I suggest choosing “*Dynamically allocated*” for saving some extra space in your hard drive.

File location and size: Give a name for your virtual hard drive file. Then, you can adjust its location and size (defaults are ok).

Your new virtual machine is listed on the left in the Virtual Box Manager window and Click “*Settings*” button while “*Debian7LAMP*” is selected (Figure 32).

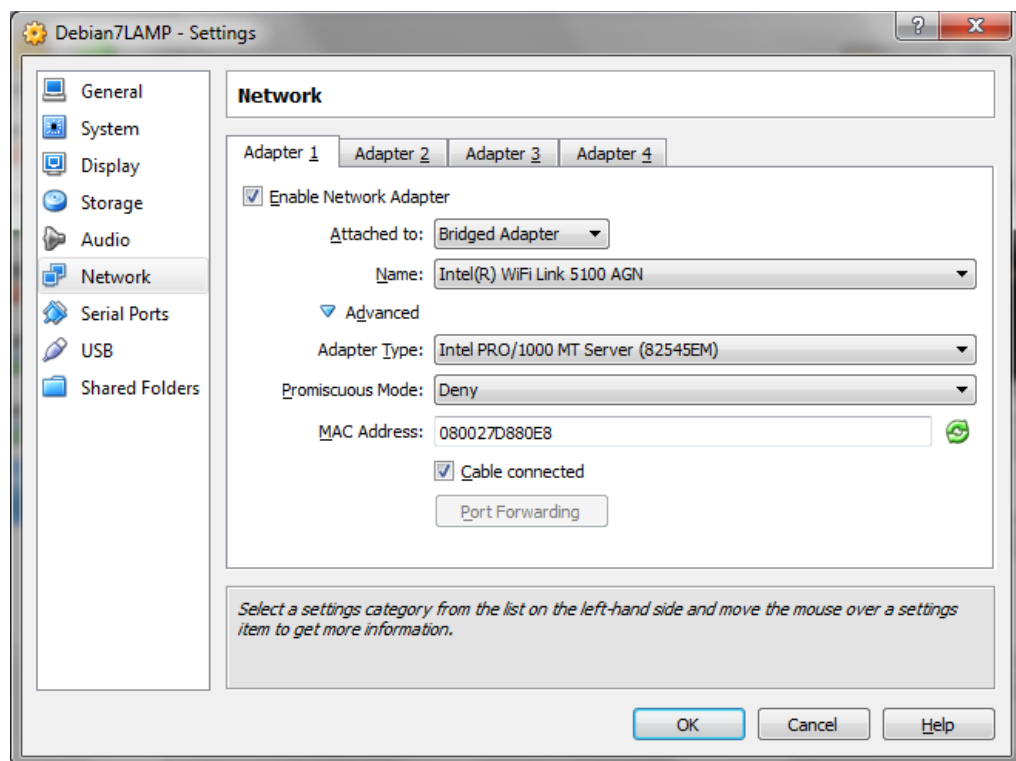


Figure 33 - Settings Window

Settings: Come to “*Network*” tab and enlarge “*Advanced*” part (Figure 33).

Network tab: Choose “*Bridged Adapter*” to use different IP from your host device for your virtual machine. The other settings do not need any attention, but you may check if you want. Then, click “*OK*” button.

We come back to Virtual Box Manager window (Figure 32) again. Now, click “*Start*” button while “*Debian7LAMP*” is selected. You will probably face some information messages and just click “*OK*” to close them. Also, you may check “*Do not show this message again*” to disable same kind of information messages.

Select start-up disk: Simply, click “*Start*” button.

Your virtual machine will try to start since you did not install any operating system, it will fail. Thus, you must insert your Debian Wheezy installation CD/DVD image file by choosing it from “*Choose a virtual CD/DVD disk file*” (Figure 34).

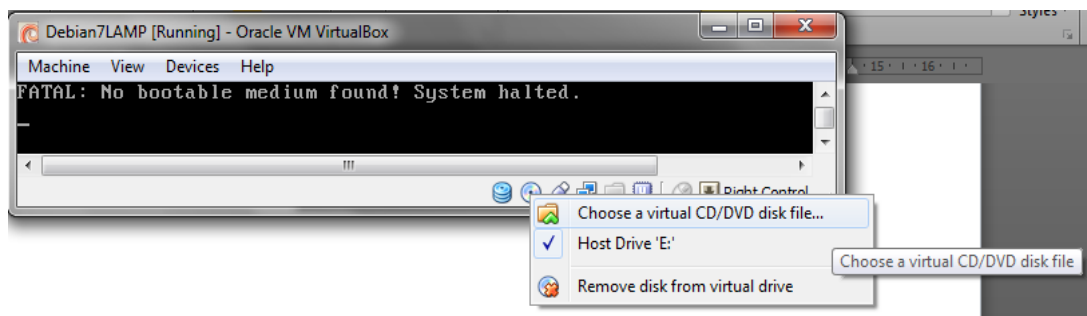


Figure 34 - Choosing Installation Medium

Your default host key for Virtual Box is “*Right Control*”. Use *Host Key* + *R* key combination to reset your virtual machine. Click “*Reset*” button to confirm your operation. Now, your system will boot up from your Debian Wheezy installation CD/DVD image file.

5.2 Installing Debian Wheezy

If you are about to install a real server, first you should insert your Debian Wheezy installation CD/DVD into your drive and boot your system from its drive. If you are just about to try this installation process via a virtual machine, then you can continue from previous section and if you see some information message boxes simply click

“OK” to close them or check “*Do not show this message again*” to disable same kind of information messages.

Debian GNU/Linux installer boot menu: Hit “*Enter*” while “*Install*” is selected.

Select a language: Choosing your language (after your selection, hit “*Enter*” button).

Select your location: Select your location. Mine is under other/Asia and it is Turkey.

Configure locales: I chose English as my language and Turkey as my location. In this kind of selections, system asks for to choose your locale. I selected “*en_US.UTF-8*”.

Configure the keyboard: Choose your keyboard layout.

The installer checks some parts (installation CD/DVD, hardware, etc...) and configures your network settings, if there is any DHCP server in your network.

Configure the network/Hostname: Enter your hostname. In my example, I called my system as “*server0.example.com*”, so I entered “*server0*” for my host name.

Configure the network/Domain name: Enter your domain name. In my example, it is “*example.com*”. Host name and domain name are getting important while setting up a real server. Otherwise, it depends on your choice.

Set up users and passwords: Setup your root user's password and verify it on the next window. But, root user has all privileges, thus system will create another account for you. In the next window, you enter your full name. After that, you will be prompted enter a new username and a password with a confirmation step for the password.

Partition disks: Choose your partitioning method. Mine is "*Guided – use entire disk and set up LVM*". This will provide me one volume group with two logical volumes, one for the file system and another one for swap. Actually, if you know what you are doing, then it depends on your choice. After that, hit "*Enter*" for next two steps.

When asked as "*Write to changes to disks and configure LVM?*", then choose "*Yes*" and hit "*Enter*".

To finish partitioning, hit "*Enter*". When asked as "*Write to changes to disks?*", then come to "*Yes*" and hit "*Enter*" again.

Partitions will be created and formatted. And the base system will be installed.

Configure the package manager: Since, we had minimal Debian Wheezy installation CD/DVD; we need to choose an online archive for our package manager.

When asked as "*Scan another CD or DVD?*" select "*No*" and hit "*Enter*" button.

When asked as "*Use a network mirror?*" select "*Yes*" and hit "*Enter*" button.

In my example, I chose *Turkey* and *ftp.metu.edu.tr* as Debian archive mirror. You can select another one depends on your locale.

You can leave blank HTTP proxy information. Also, you do not need to participate in the package usage survey.

Software selection: I will just select *SSH Server* and *Standard system utilities*, because I need to get full control over what installed is on my system.

Install the GRUB boot loader on a hard disk: Hit “*Enter*” button on “*Yes*” to install.

To finish installation, hit “*Enter*” button on “*Continue*”. System will restart and ask for your login details.

5.3 Installation of Other Components Using Automated Scripts

Before starting to run our scripts, we need to login our Debian Wheezy system as *root*. Thus, write *root* as username when prompted and supply your root password to the system.

Locate your directory to “*tmp*” folder (**root@server0:~# cd ../tmp**).

5.3.1 Running Basic Apache, MySQL and PHP Installation Script

You can find codes of “*LAMP_install.sh*” script in Appendix B: *LAMP_install.sh*.

```
root@server0:~# cd ../tmp
root@server0:/tmp# wget http://192.168.5.100/LAMP_install.sh
--2013-07-29 04:28:18-- http://192.168.5.100/LAMP_install.sh
Connecting to 192.168.5.100:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5101 (5.0K) [application/x-sh]
Saving to: `LAMP_install.sh'

100%[=====] 5,101 --.-K/s in 0s
2013-07-29 04:28:18 (122 MB/s) - `LAMP_install.sh' saved [5101/5101]

root@server0:/tmp# ls
LAMP_install.sh
root@server0:/tmp# bash LAMP_install.sh_
```

Figure 35 - Starting Script Installation

Using “*wget*” command I have downloaded my installer script which is called “*LAMP_install.sh*” to the “*tmp*” directory. Then, using “*bash*” command I have started to installation progress (Figure 35).

Server IP: Provide an IP address (*192.168.5.101*).

Hostname: Enter your hostname (*server0*).

Fully Qualified Hostname: Enter your domain name (*example.com*).

MySQL Root Password: Enter your MySQL Root Password.

The system will process under your supplied information above. At the end of this process, beware of the warning message and press “*Enter*” to continue installation.

Configuring phpMyAdmin: Enter your password of the administrative user for phpMyAdmin. Next, you will be prompted entering another password for phpMyAdmin to register with the database server.

Components that left are installed by the system and ready to be tested. Using the command “*ifconfig -a*” we can see our server IP address which is entered by me during script process.

I located my browser to <http://192.168.5.101/> to check Apache is running properly (Figure 36).

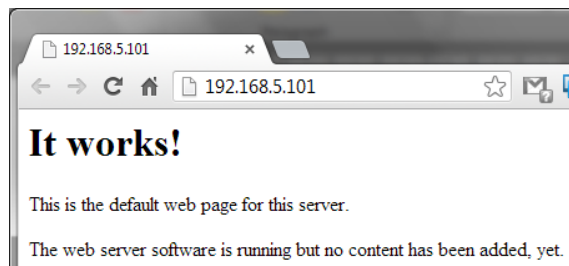


Figure 36 - Apache Works

To test phpMyAdmin is running properly or not, locate your browser <http://192.168.5.101/phpmyadmin>. You must see “*Welcome to phpMyAdmin*” page which includes login form. Then, I can login to phpMyAdmin with username *root* and my password that I created during script installation.

5.3.2 An Automated Full ISPConfig 3 Installer Script

This is a software project that is developing using GitHub by Drew Clardy [17]. Simply run the command to start installation script. Progress is very close to previous one, so that you can take it as a sample installation.

```
wget --no-check-certificate -O ISPConfig3.tgz
https://github.com/dclardy64/ISPConfig-3-Debian-Installer/tarball/master; tar zxvf
ISPConfig3.tgz; cd *Installer*; bash debian_install.sh
```

Chapter 6

CONCLUSION

In this thesis I have analyzed and discussed the general security risks for any web based Learning Management System and focused especially on the security risks of Moodle (version 2.4), because of the highly availability and applicability in the market. During this analysis I evaluated Moodle's approaches and processes to avoid these risks. In general Moodle manages these risks successfully, but of course, there will be still some security holes remaining that are generally fixed as soon as possible. In general, the observations made and tests conducted leads to the conclusion that Moodle is a secure LMS platform. Despite Moodle's being open source and as it is being widely used, therefore of major interest for hackers, it is secure in most cases.

The hardest part for any web application in general and for Moodle in detail is the handling of sessions. Moodle only provides SSL for login page which makes it very difficult for an attacker to grab the information. One of the weaknesses of Moodle found is that the cookie based brute force prevention system can be avoided by the usage of external scripts. But as this weakness is known, many protection mechanisms are available to remove this weakness.

Furthermore I have presented a secure installation process to ensure a proper installation against fraud attacks. Another issue in setup process is related to the server and its components. I provide an easy way to install a secure web server.

As a result, the Moodle platform manages the huge variety of functionality and numerous security measures quite successfully. I can say this system is secure enough to serve your online courses and has all the functionality you may need. Finally, this thesis discusses the security risks, how Moodle deals with it, and gives guidance for the installation process.

REFERENCES

- [1] “edX,” [Online]. Available: <https://www.edx.org/>.

- [2] “Coursera,” [Online]. Available: <https://www.coursera.org/>.

- [3] M. M. Waldrop, «Massive open online courses are transforming higher education — and providing fodder for scientific research,» *Nature*, cilt 495, no. 7440, pp. 160-163, 14 March 2013.

- [4] “Edudemic,” 27 October 2012. [Online]. Available: <http://www.edudemic.com/2012/10/the-20-best-learning-management-systems/>. [Accessed 5 September 2013].

- [5] “Moodle,” [Online]. Available: <https://moodle.org/stats/>.

- [6] “w3Techs,” [Online]. Available: http://w3techs.com/technologies/overview/programming_language/all.

- [7] “w3Techs,” [Online]. Available: <http://w3techs.com/technologies>.

- [8] “w3Techs,” [Online]. Available: http://w3techs.com/technologies/overview/operating_system/all.

- [9] “w3Techs,” [Online]. Available: <http://w3techs.com/technologies/details/os-unix/all/all>.

- [10] “w3Techs,” [Online]. Available:
<http://w3techs.com/technologies/details/os-linux/all/all>.
- [11] “w3Techs,” [Online]. Available:
http://w3techs.com/technologies/overview/web_server/all.
- [12] “ISPConfig,” [Online]. Available: <http://www.ispconfig.org/ispconfig-3/>.
- [13] OWASP Foundation, “OWASP Top 10 - 2013 Release,” OWASP (The Open Web Application Security Project), 2013.
- [14] Oracle, “bugs.mysql.com,” Oracle, 29 May 2012. [Online]. Available:
<http://bugs.mysql.com/bug.php?id=64884>. [Accessed 5 August 2013].
- [15] computersecuritystudent.com, “Computer Security Student,” 12 April 2012.
[Online]. Available:
http://computersecuritystudent.com/SECURITY_TOOLS/DVWA/DVWA_v1_07/lesson5/index.html. [Accessed 10 July 2013].
- [16] moodle.org, “Roles,” 28 January 2013. [Online]. Available:
<http://docs.moodle.org/dev/Roles#Definitions>. [Accessed 9 August 2013].
- [17] D. Clardy, “GitHub,” 26 July 2013. [Online]. Available:
<https://github.com/dclardy64/ISPConfig-3-Debian-Installer>. [Accessed 28 July 2013].

- [18] F. Timme, "HowtoForge - Linux Howtos and Tutorials," 27 May 2013. [Online]. Available: <http://www.howtoforge.com/installing-apache2-with-php5-and-mysql-support-on-debian-wheezy>. [Accessed 20 July 2013].
- [19] F. Timme, "HowtoForge - Linux Howtos and Tutorials," 7 5 2013. [Online]. Available: <http://www.howtoforge.com/perfect-server-debian-wheezy-apache2-bind-dovecot-ispconfig-3>. [Accessed 20 July 2013].

APPENDICES

Appendix A: Ettercap Attack Logs

Listening on eth1... (Ethernet)

eth1 -> 08:00:27:5D:6F:59 192.168.182.12 255.255.255.0

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file

Privileges dropped to UID 65534 GID 65534...

28 plugins

40 protocol dissectors

55 ports monitored

7587 mac vendor fingerprint

1766 tcp OS fingerprint

2183 known services

Randomizing 255 hosts for scanning...

Scanning the whole netmask for 255 hosts...

16 hosts added to the hosts list...

Host 192.168.182.34 added to TARGET1

ARP poisoning victims:

GROUP 1 : 192.168.182.34 00:13:77:AE:0F:01

GROUP 2 : ANY (all the hosts in the list)

Starting Unified sniffing...

HTTP : 192.168.182.27:80 -> USER: Admin PASS: R#ot1234 INFO:

http://192.168.182.27/moodle24/login/index.php

DHCP: [192.168.182.1] ACK : 0.0.0.0 255.255.255.0 GW 192.168.182.1 DNS

193.140.173.173 "mrtgdhcp.emu.edu.tr"

Appendix B: LAMP_install.sh

```
#!/bin/bash

#####

# Basic development LAMP setup script for Debian 7.

# Yılmaz Demirci - http://www.yilmazdemirci.com/

# 28.07.2013

#

# This script shortened from Drew Clardy's "Complete ISPConfig setup script for
Debian 7" and

# re-written with respect to Falko Timme's "Installing Apache2 with PHP5 and
MySQL Support on Debian Wheezy"

# article

#####

# Check if user is root

if [ $(id -u) != "0" ]; then

    echo "Error: You must be root to run this script, please use the root user to
install the software."

    exit 1

fi

back_title="Debian 7 Basic LAMP Server Installer"

assign_variables(){

while [ "x$serverIP" == "x" ]

do
```

```

serverIP=$(whiptail --title "Server IP" --backtitle "$back_title" --inputbox
"Please specify a Server IP" --nocancel 10 50 3>&1 1>&2 2>&3)

done

while [ "x$HOSTNAMESHORT" == "x" ]

do

HOSTNAMESHORT=$(whiptail --title "Hostname" --backtitle "$back_title"
--inputbox "Please specify a Hostname" --nocancel 10 50 3>&1 1>&2 2>&3)

done

while [ "x$HOSTNAMEFQDN" == "x" ]

do

HOSTNAMEFQDN=$(whiptail --title "Fully Qualified Hostname" --
backtitle "$back_title" --inputbox "Please specify a Fully Qualified
Hostname" --nocancel 10 50 3>&1 1>&2 2>&3)

done

while [ "x$mysql_pass" == "x" ]

do

mysql_pass=$(whiptail --title "MySQL Root Password" --backtitle
"$back_title" --inputbox "Please specify a MySQL Root Password" --
nocancel 10 50 3>&1 1>&2 2>&3)

done

}

debian7_install_basic (){

#Set hostname and FQDN

sed -i "s/${serverIP}.*/${serverIP} ${HOSTNAMEFQDN}
${HOSTNAMESHORT}"/ /etc/hosts

```

```

echo "$HOSTNAMESHORT" > /etc/hostname

/etc/init.d/hostname.sh start >/dev/null 2>&1

#Updates server and install commonly used utilities

cp /etc/apt/sources.list /etc/apt/sources.list.backup

cat > /etc/apt/sources.list <<EOF

deb http://ftp.metu.edu.tr/debian/ wheezy main contrib non-free

deb-src http://ftp.metu.edu.tr/debian/ wheezy main contrib non-free

deb http://security.debian.org/ wheezy/updates main contrib non-free

deb-src http://security.debian.org/ wheezy/updates main contrib non-free

# wheezy-updates, previously known as 'volatile'

deb http://ftp.metu.edu.tr/debian/ wheezy-updates main contrib non-free

deb-src http://ftp.metu.edu.tr/debian/ wheezy-updates main contrib non-free

# DotDeb

deb http://packages.dotdeb.org wheezy all

deb-src http://packages.dotdeb.org wheezy all

EOF

wget http://www.dotdeb.org/dotdeb.gpg

cat dotdeb.gpg | apt-key add -

apt-get update

apt-get -y upgrade

apt-get -y install vim-nox unzip

} #end function debian_install_basic

```

```

debian7_install_DashNTP (){

    echo "dash dash/sh boolean false" | debconf-set-selections

    dpkg-reconfigure -f noninteractive dash > /dev/null 2>&1

    #Synchronize the System Clock

    apt-get -y install ntp ntpdate

} #end function debian_install_DashNTP

debian7_install_MYSQL (){

#Install MySQL

    echo "mysql-server-5.5 mysql-server/root_password password $mysql_pass"
    | debconf-set-selections

    echo "mysql-server-5.5 mysql-server/root_password_again password
    $mysql_pass" | debconf-set-selections

    apt-get -y install mysql-server mysql-client

#Allow MySQL to listen on all interfaces

    cp /etc/mysql/my.cnf /etc/mysql/my.cnf.backup

    sed -i 's/bind-address = 127.0.0.1/#bind-address = 127.0.0.1/'
    /etc/mysql/my.cnf

    /etc/init.d/mysql restart

}

debian7_install_Apache2 (){

```

```

#Install Apache2

    apt-get -y install apache2

}

debian7_install_PHP5 (){

    apt-get -y install php5 libapache2-mod-php5

#Restart Apache

    /etc/init.d/apache2 restart

}

debian7_install_mysqlPHP5 (){

    apt-get -y install php5-mysql php5-curl php5-gd php5-intl php-pear php5-
    imagick php5-imap php5-mcrypt php5-memcache php5-ming php5-ps php5-
    pspell php5-recode php5-snmp php5-sqlite php5-tidy php5-xmlrpc php5-xsl

#Restart Apache

    /etc/init.d/apache2 restart

    apt-get -y install php-apc

#Restart Apache

    /etc/init.d/apache2 restart

}

debian7_install_phpmyadmin (){

    echo

    "=====

```

```

echo "You will be prompted for some information during the install of
phpmyadmin."

echo "Select NO when asked to configure using dbconfig-common"

echo "Please enter them where needed."

echo

"=====

echo "Press ENTER to continue.."

read DUMMY

echo 'phpmyadmin phpmyadmin/reconfigure-webserver multiselect apache2' |
debconf-set-selections

echo 'phpmyadmin      phpmyadmin/dbconfig-install      boolean false' |
debconf-set-selections

apt-get -y install phpmyadmin

#Restart Apache

    /etc/init.d/apache2 restart

}

#Execute functions#

if [ -f /etc/debian_version ]; then

    assign_variables

    debian7_install_basic

    debian7_install_DashNTP

    debian7_install_MYSQL

    debian7_install_Apache2

```



```
    debian7_install_PHP5
    debian7_install_mysqlPHP5
    debian7_install_phpmyadmin
else echo "Unsupported Linux Distribution."
fi
#End execute functions#
```