



UNIVERSITAS
INDONESIA

CEP-CCIT

FAKULTAS TEKNIK

WebApp Security Analysis

Group : 5

Name : Muhammad Armed Bintang Pradana

Nur Iqbal Maulana

Class : 3CS1

CEP CCIT

FAKULTAS TEKNIK UNIVERSITAS INDONESIA

2024

PROJECT ON

Introduction of CyberSecurity

Developed By

1. Muhammad Armed Bintang Pradana
2. Nur Iqbal Maulana

Batch Code : 3CS1

WebApp Security Analysis

Start Date : October 7, 2024

End Date : October 2, 2024

Name Of Faculty : Ivan Firdaus, S. T.

Name Of Developer :

1. Nur Iqbal Maulana
2. Muhammad Armed Bintang Pradana

Date Of Submission : October 21, 2024

CERTIFICATE

This is to certify that the report titled "WebApp Security Analysis", embodies the original work done by Muhammad Armed Bintang Pradana and Nur Iqbal Maulana. Project in partial fulfillment of their course requirement at CEP CCIT Faculty of Engineering Universitas Indonesia.

Coordinator :

Ivan Firdaus, S. T.

ACKNOWLEDGEMENT

The author expresses his gratitude to Allah SWT for all the abundance of grace and mercy. His mercy and grace, and do not forget the shalawat and greetings we send to the Prophet Muhammad SAW, so that we can complete this project with the title " WebApp Security Analysis" and without him we would not be able to complete this project on time. Time that has been calculated, and the author also wants to thank Mr Ivan Firdaus, S. T., as the supervisor who has provided suggestions and advice that are very helpful to the author in writing this project. Although there are many challenges and obstacles that we face in making this project, we can finally complete it. Finally, we were able to complete this project. The author realizes that this assignment is still far from perfection, and if colleagues and lecturers are willing to provide suggestions and criticism, then this assignment is not perfect. Supervisors are pleased to provide suggestions and criticism for the sake of the perfection of this project, and we as writers will be greatly helped. We, as writers, will be greatly helped by these suggestions and criticisms.

BACKGROUND

A WebApp (Web Application) is an application that runs on a web server and is accessed through an internet browser, providing ease of access and functionality for users without the need for local installation. However, WebApps also present a high security risk, especially when they are not properly protected. Various cyberattacks such as port scanning, SQL injection, and path traversal are often used by attackers to exploit weaknesses in web applications, causing serious damage or data theft.

The project will focus on Four main types of attacks that often occur in WebApps. First, port scanning, is used to find open ports on the server, which attackers can use to gain early access. Second, SQL injection, where insecure inputs allow attackers to interact with an application's database directly, potentially accessing or modifying critical data. Third, path traversal, which allows an attacker to access files or directories on a server that would otherwise be unreachable through file path manipulation. Fourth, File Upload, where attacker can upload shell or backdoor to the system directly from the website interface

In this project, we'll be checking a sample WebApp to determine if it's vulnerable to these attacks. Using tools in Kali Linux such as Nmap for port scanning, SQLMap for checking for SQL injection, Path Traversal, and file the project aims to identify potential vulnerabilities that may exist and provide mitigation recommendations to improve the security of web applications.

SYSTEM ANALYSIS

System Summary: Web applications, which run on web servers and provide ease of access, pose a high security risk when not properly protected. Cyberattacks like port scanning, SQL injection, and path traversal exploit weaknesses in web applications, causing damage or data theft. This project aims to identify potential vulnerabilities in a sample WebApp using tools in Kali Linux, such as Nmap for port scanning and SQLMap for SQL injection, and provide mitigation recommendations to improve the security of web applications.

System Processes:

1. Analyzing Security of The Sample WebApp: School 1 From VulnHub
 - a. Port Scanning
 - b. Path Traversal
 - c. SQLInjection
 - d. File Upload Vulnerability
2. Recommended Technique to Increase Security of WebApp from Discussed Attacks Type.

BASIC THEORY

1. WebApp

Web Application (WebApp) is a software application that operates on a web server and is accessed via a web browser over the Internet or an intranet. WebApps are characterized by their interactivity and dynamic nature, often built using technologies such as HTML, CSS, JavaScript, and various server-side languages like PHP, Python, or Ruby. Examples of WebApps include online banking systems, social media platforms, and e-commerce websites.

2. Vulnerability

Vulnerability refers to a weakness or flaw in a system, application, or network that can be exploited by attackers to gain unauthorized access or inflict harm. Vulnerabilities can be categorized into different types, including software vulnerabilities, which may arise from bugs or misconfigurations; network vulnerabilities, such as unsecured ports or protocols; and physical vulnerabilities, which result from inadequate security measures.

3. PortScanning

Port scanning is a technique utilized to identify open ports and services available on a networked device, serving as a means to assess the security of a system. The primary purpose of port scanning is to discover the services running on a host and to identify any vulnerabilities associated with those services. Common tools for port scanning include Nmap, Netcat, and Angry IP Scanner.

4. Path Traversal

Path Traversal, also known as Directory Traversal, is a security vulnerability that enables an attacker to access files and directories outside the intended directory structure of an application. This can result in unauthorized file access, information disclosure, or even system compromise. For example, an attacker might manipulate a

URL to access sensitive files by using a path like `http://example.com/app?file=../../etc/passwd`.

5. SQL Injection

SQL Injection (SQLi) is a code injection technique that exploits vulnerabilities in an application's software by inserting or "injecting" SQL code into a query. This can lead to unauthorized access to the database, data manipulation, or full system compromise. To prevent SQL injection attacks, developers are advised to use parameterized queries, prepared statements, and input validation.

6. File Upload Vulnerability

File Upload Vulnerability occurs when an application permits users to upload files without adequate validation or security measures, which can result in unauthorized file execution or system compromise. Attackers may exploit this vulnerability by uploading malicious files, such as scripts or executables, that could be executed on the server. To mitigate these risks, it is essential to implement file type validation, size restrictions, and store uploaded files outside the web root.

IP INVOLVED

1. IP of the machine : 192.168.1.69
2. IP of the target : 192.168.1.101

DISCOVERING THE WEBAPP TARGET

First of all, by using Virtual Machine that provide some WebApp and using netdiscover, we can know our target IP is 192.168.1.101.

Currently scanning: 192.168.10.0/16 | Screen View: Unique Hosts

23 Captured ARP Req/Rep packets, from 17 hosts. Total size: 1380

IP	At MAC Address	Count	Len	MAC Vendor / Hostname
192.168.1.1	f4:6f:ed:09:1e:c0	6	360	Fiberhome Telecommunication Technologies Co.,LTD
192.168.1.250	00:12:17:d5:1c:92	2	120	Cisco-Linksys, LLC
192.168.1.50	e8:48:b8:45:6a:1b	1	60	TP-Link Corporation Limited
192.168.1.51	b0:4e:26:51:90:fb	1	60	TP-LINK TECHNOLOGIES CO.,LTD.
192.168.1.52	d8:32:14:cd:85:70	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.54	b8:3a:08:9d:05:c8	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.55	b8:3a:08:9d:05:90	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.56	b8:3a:08:9d:05:b8	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.57	d8:32:14:cd:85:60	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.58	b8:3a:08:9d:05:80	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.59	b8:3a:08:9d:05:a8	1	60	Tenda Technology Co.,Ltd.Dongguan branch
192.168.1.61	90:d8:f3:6e:b9:2b	1	60	zte corporation
192.168.1.67	dc:41:a9:10:7c:0f	1	60	Intel Corporate
192.168.1.62	f2:3c:5e:9b:50:0b	1	60	Unknown vendor
192.168.1.101	08:00:27:fe:d5:80	1	60	PCS Systemtechnik GmbH
192.168.1.73	7a:70:b3:3f:16:5a	1	60	Unknown vendor
192.168.1.68	42:e3:67:7a:77:af	1	60	Unknown vendor

We use nmap command on same terminal to finding any port used on the webapp.

```
(root@kali)-[/home/kali]
# nmap -oA nmapscan -sT 192.168.1.101
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-19 14:23 EDT
Nmap scan report for 192.168.1.101
Host is up (0.0063s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
80/tcp    open  http
MAC Address: 08:00:27:FE:D5:80 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 14.05 seconds
```

There are some Port with “open” status. Here are the function of each ports

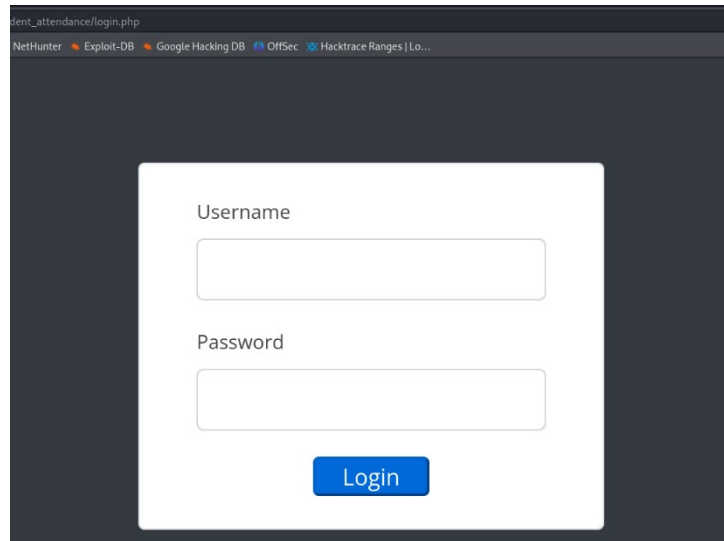
PORT 22:

PORT 23:

PORT 80:

It's very crucial mistakes from the developer of the WebApp because the SSH Port (Port 22) is critical port for developer for develop or maintaining their system.

We change to the internet browser and enter the IP of our target. It shows to us some login page, which means the IP is a Website Application. We don't know what's the Username and Password for login to the system. We will analyze the security of this WebApp more further.



PATH TRAVERSAL

Using dirbuster, we can find any directory listed on a webapp. Here, we found an interesting directory that we can access.

```
(root@kali)~[/home/kali]
# dirb http://192.168.1.101/student_attendance/ -w /usr/share/wordlists/dirb/big.txt

DIRB v2.22
By The Dark Raver

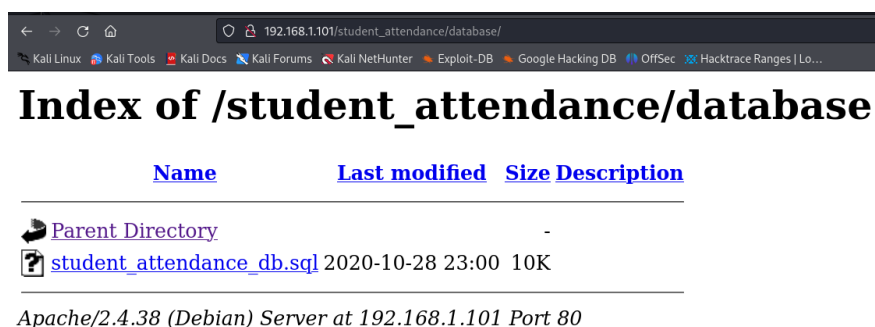
START_TIME: Sat Oct 19 14:36:29 2024
URL_BASE: http://192.168.1.101/student_attendance/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Not Stopping on warning messages

GENERATED WORDS: 4612

--- Scanning URL: http://192.168.1.101/student_attendance/ ---
=> DIRECTORY: http://192.168.1.101/student_attendance/assets/
=> DIRECTORY: http://192.168.1.101/student_attendance/database/
+ http://192.168.1.101/student_attendance/index.php (CODE:302|SIZE:14619)

--- Entering directory: http://192.168.1.101/student_attendance/assets/ ---
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)
=> DIRECTORY: http://192.168.1.101/student_attendance/assets/css/
=> DIRECTORY: http://192.168.1.101/student_attendance/assets/js/
=> DIRECTORY: http://192.168.1.101/student_attendance/assets/uploads/
=> DIRECTORY: http://192.168.1.101/student_attendance/assets/vendor/
```

This is inside of that directory. There is another interesting file (a database file) that we can download and access the content of the file.



Name	Last modified	Size	Description
Parent Directory	-	-	-
student_attendance_db.sql	2020-10-28 23:00	10K	

Apache/2.4.38 (Debian) Server at 192.168.1.101 Port 80

File has been downloaded, opened it and found hashes. If we crack the hashes we could get the password for administrator.

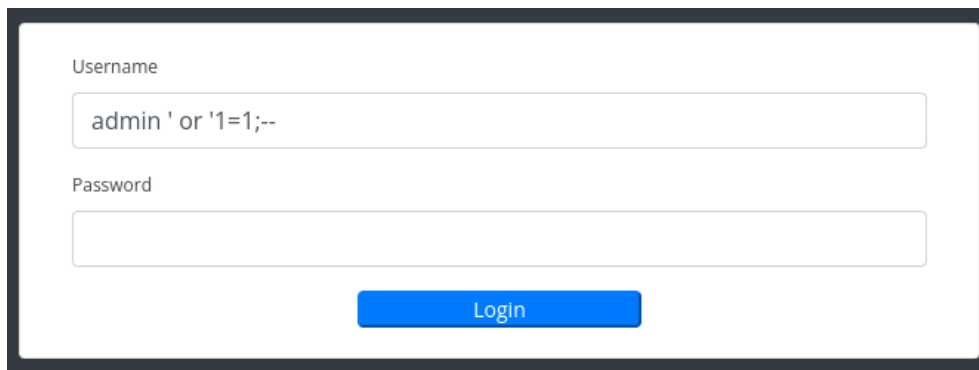
```
CREATE TABLE `users` (
  `id` int(30) NOT NULL,
  `name` text NOT NULL,
  `username` varchar(200) NOT NULL,
  `password` text NOT NULL,
  `type` tinyint(1) NOT NULL DEFAULT 3 COMMENT '1=Admin,2=Staff',
  `faculty_id` int(30) NOT NULL COMMENT 'for faculty user only'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- Dumping data for table `users`
--

INSERT INTO `users` (`id`, `name`, `username`, `password`, `type`, `faculty_id`) VALUES
(1, 'Administrator', 'admin', '0192023a7bbd73250516f069df18b500', 1, 0),
(2, 'John Smith', 'jsmith@sample.com', 'af606ddc433ae6471f104872585cf880', 3, 1);
```

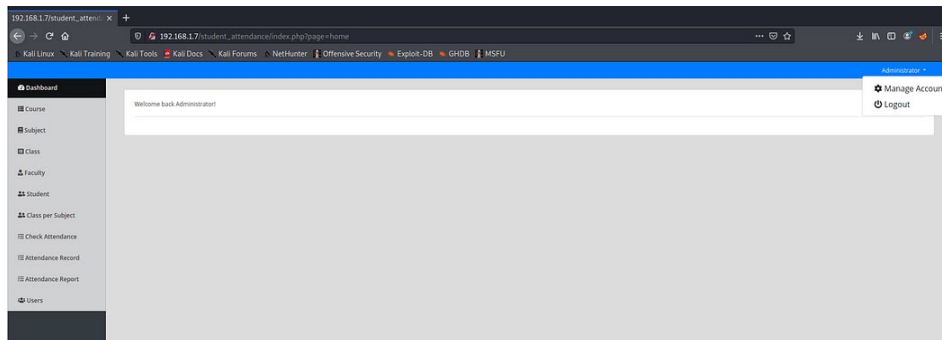
SQL INJECTION

There is another way that we can try to login without any credentials, the technique is called by SQL Injection. We can just try typing some payload into username field we can get logged in as admin user. To check whether the WebApp is having any SQL vulnerability or no. We try to type **admin' OR '1=1; --** to username field and type nothing on password field and we can login to the WebApp.



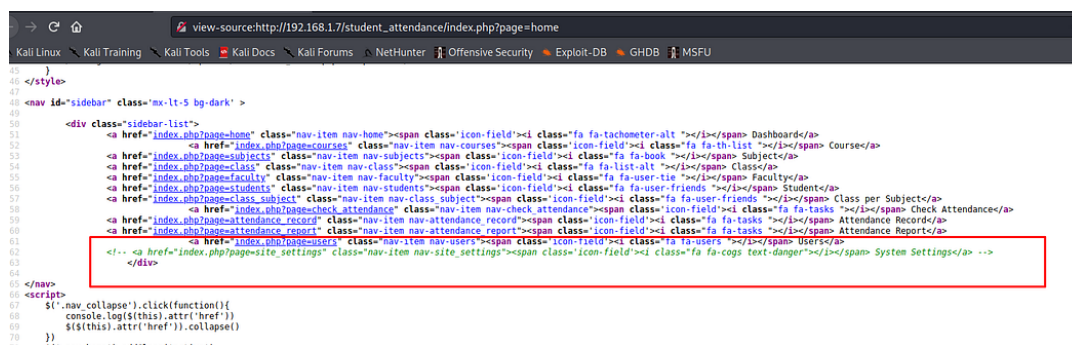
A screenshot of a web application login interface. It features two input fields: 'Username' and 'Password'. The 'Username' field contains the payload 'admin' OR '1=1; --'. The 'Password' field is empty. Below the fields is a blue 'Login' button.

Here is admin dashboard.



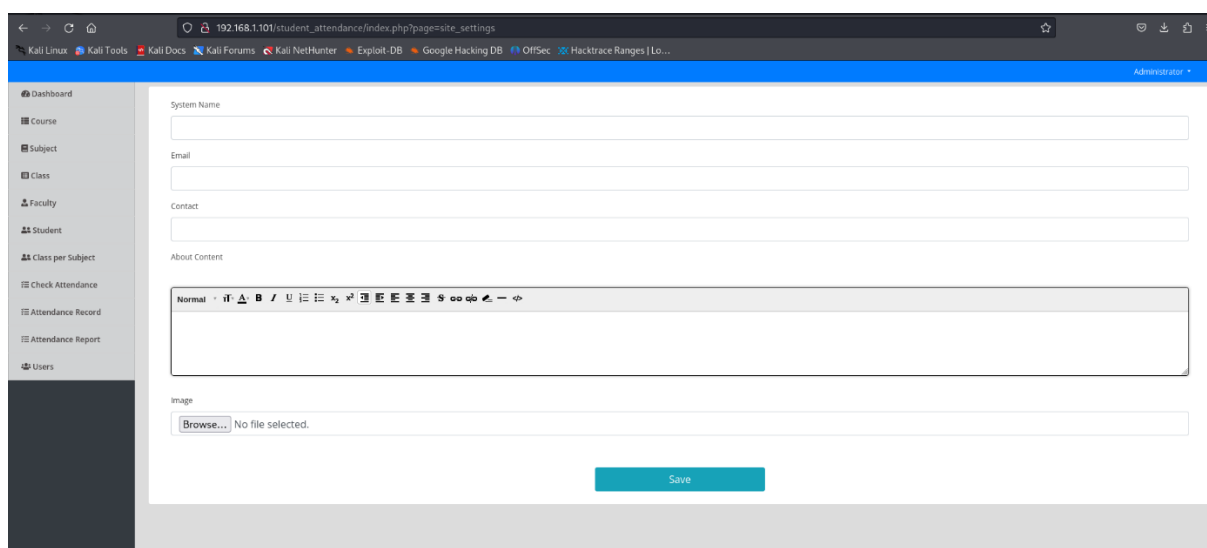
FILE UPLOAD VULNERABILITY

Let's visit page viewsource for any sensitive stuff. One interesting comment found inside the page, this link is not tie-up with dashboard so take a note of this and visit this page by typing into the URL.



```
view-source:http://192.168.1.7/student_attendance/index.php?page=home
<div class="sidebar-list">
  <a href="index.php?page=home" class="nav-item nav-home">Dashboard</a>
  <a href="index.php?page=courses" class="nav-item nav-courses">Course</a>
  <a href="index.php?page=subjects" class="nav-item nav-subjects">Subject</a>
  <a href="index.php?page=class" class="nav-item nav-class">Class</a>
  <a href="index.php?page=faculty" class="nav-item nav-faculty">Faculty</a>
  <a href="index.php?page=students" class="nav-item nav-student">Student</a>
  <a href="index.php?page=class_subject" class="nav-item nav-class_subject">Class per Subject</a>
  <a href="index.php?page=check_attendance" class="nav-item nav-check_attendance">Check Attendance</a>
  <a href="index.php?page=attendance_record" class="nav-item nav-attendance_record">Attendance Record</a>
  <a href="index.php?page=attendance_report" class="nav-item nav-attendance_report">Attendance Report</a>
  <a href="index.php?page=users" class="nav-item nav-users">Users</a>
  <a href="index.php?page=site_settings" class="nav-item nav-site_settings">System Settings</a>
</div>
```

It's opened and it has file upload functionality. This portal is created using php language.



System Name

Email

Contact

About Content

Normal

Image

Browse... No file selected.

Save

We try to upload some shells to the system and it worked. We can upload our shell to the system and we can apply a technique that called reverse shell and after we can enter the system, we can do a bunch of technique for many reasons.

System Name

3cs1

Data successfully saved.


Email

3cs1@gmail.com

Contact:

08212345678910

About Content:

Normal •  shell

Image

php-reverse-shell.php

CONCLUSION

As we did before, we can conclude that the **School: 1** WebApp is has Open Port Vulnerability, Path Traversal, Vulnerable to Sql Injection, and File Upload Vulnerability. Based on this analysis, we can advice to the developer so that their WebApp will be having stronger security system as below.

1. Close Unnecessary Ports

Port Audit: Conduct an audit to determine which ports are essential for the application's operation.

Close Unused Ports: Port 23 (Telnet) should be closed as it is insecure. Consider using SSH (port 22) for remote access.

Firewall: Implement a firewall to restrict access to open ports only to trusted IP addresses.

2. Secure Directory Access

Access Permission Settings: Ensure that directories containing sensitive credentials have strict access permissions. Only authorized users should be able to access these directories.

Maximize .htaccess File: Use a .htaccess file to restrict access to certain directories on the web server if using Apache.

Remove Sensitive Directories: If possible, delete or move directories containing sensitive information away from public access.

3. Prevent SQL Injection

Use Parameterized Queries: Ensure that all SQL queries use parameterized queries or prepared statements to prevent SQL injection.

Input Validation: Validate and sanitize user input to ensure that the data received meets expected criteria.

Security Testing: Conduct regular penetration testing to identify and remediate SQL injection vulnerabilities.

4. Secure File Uploads

File Type Validation: Implement strict validation for allowed file types. Only permit files with safe extensions (e.g., images).

File Storage: Store uploaded files outside the web root directory to prevent direct access.

Antivirus Scanning: Use antivirus software to scan uploaded files before further processing.

File Size Restrictions: Limit the size of files that can be uploaded to prevent Denial of Service (DoS) attacks.

5. Security Training and Awareness

Team Training: Provide cybersecurity training to the development team to raise awareness of best practices in secure application development.

Regular Updates: Ensure that all software and libraries used in the application are regularly updated to address known vulnerabilities.

6. Monitoring and Incident Response

Security Monitoring: Implement monitoring systems to detect suspicious activity and potential attacks.

Incident Response Plan: Prepare an incident response plan to quickly and effectively address security breaches.

By implementing these measures, developers can significantly enhance the security of the web application and protect data and systems from potential attacks.

SYSTEM REQUIREMENT

Hardware:

1. Lenovo Ideapad Slim 3

Software:

1. VirtualBox
2. Kali Linux Virtual Machine
3. School 1 Virtual Machine from Vulnhub
4. Microsoft Word

FILE PROJECT DETAILS	
Group 5 Paper.PDF	Paper File
PPT Group 5.PDF	Presentation File