‘Hack Me Please’ - VulnHub Write-up

Shrike InfoSec

2023-01-06

Contents

1 Assessment Overview

1.1 Introduction

This is an intentionally vulnerable machine designed to be used for OSCP preparation.

2 The Scope

The scope of this test is limited to a single public facing web server and any connected services or internal devices. This web server is hosted at the following address:

- 10.10.10.10

3 Systems Overview

The following systems were identified as part of the engagement:

System Name	IP Address/URL	Notes
Web Server	10.10.10.10	A public-facing web server.

4 Executive Summary

The client's public facing web server was compromised using information exposed in the website's source code. The source code included a comment describing a service running on the backend server that had a known vulnerability. The service is open-source, and the source code revealed some default credentials within one of the configuration files that was accessible to the outside world. This also included database credentials, which were then used to gain access to the admin account of this service. This account was then used to upload a web shell that was accessible via a browser, which was then used to create a reverse shell connection to the machine. A user account was identified on the machine which allowed privilege escalation with no password required, thereby allowing the machine to be compromised.

5 Tools Used

The following tools were used during this engagement:

Tool	Version
Kali Linux	2022.4
Nmap	7.80
Metasploit Framework	6.2.13

6 Findings Summary

6.1 Information Disclosure

Apache httpd/SeedDMS

Version information of a critical service disclosed.

Severity: High

Description: A comment in the source code of the website revealed the current version of the backend service running.

Impact: This information allows an attacker to identify potential vulnerabilities for the specified service, and gives a rough idea of how up-to-date the services/applications running on the backend might be.

Suggested Remediation: Any comments included in the source code should be removed when used in production.

6.2 Broken Access Control

Apache httpd

Incorrectly configured permissions allow for data to be exposed that otherwise should not be readable.
A01 - Broken Access Control

Severity: Critical

Description: An `.htaccess` file is accessible on the web server, which reveals a file containing sensitive information. This file (`settings.xml`) is publicly-accessible and exposes credentials for a MySQL Database used by the SeedDMS service.

Impact: This file being exposed allows for an attacker to gather sensitive configuration data (such as passwords for databases, API keys)

Suggested Remediation: Both the `.htaccess` and `settings.xml` files should have their permissions changed so that they are not accessible from the outside world.

6.3 Insecure Credentials

MySQL

The database was using insecure credentials that could be easily brute-forced or guessed.

Severity: Critical

Description: The MySQL database was using "easy to guess/brute-force" credentials.

Impact: Gaining access to the database allows an attacker to create user accounts, exfiltrate data, extract passwords, etc.

Suggested Remediation: A secure password that can not be easily guessed or brute-forced should be used. Database user permissions should also make use of the 'principle of least privilege'.

6.4 Password Re-Use

MySQL/User Accounts

The database contained a user account with a password that was re-used elsewhere.

Severity: Critical

Description: One of the user accounts present in the database had the same password for the web application and their account on the machine itself.

Impact: Once an attacker has access to the password from the database, it can then be used to log in as that user on the machine directly. This allows for additional enumeration, lateral movement and privilege escalation.

Suggested Remediation: Users should not re-use passwords and should make use of a password manager.

6.5 Password Re-Use

MySQL/User Accounts

The database contained a user account with a password that was re-used elsewhere.

Severity: Critical

Description: One of the user accounts present in the database had the same password for the web application and their account on the machine itself. This user has full `sudo` permissions and therefore can become the `root` user.

Impact: Once an attacker has access to the password from the database, it can then be used to log in as that user on the machine directly. This allows for additional enumeration, lateral movement and ultimately, privilege escalation to a `root` user.

Suggested Remediation: Users should not re-use passwords and should make use of a password manager.

7 Attack Narrative

7.1 Enumeration - Public Web Server

7.1.1 nmap scan

An initial nmap scan was done on the target IP address and revealed the following information:

Port	Service
80	Apache httpd 2.4.41
3306	mysql 8.0.25-0ubuntu-0.20.04.1
33060	tcpwrapped

This tells us the following information:

- There is a web server running on this machine on port `80`
- MySQL is present on the machine
- The machine is running Ubuntu 20.04 (inferred from the `mysql` service running)
- There is an additional service running on port `33060` that we do not have permission to communicate with.

7.1.2 Website Fingerprinting

The website content itself does not reveal any particularly useful information and appears to be using a template.

A look at the site's source code reveals a few JavaScript files in the `<head>` section.

An older version of the jQuery library is present (1.11.2) and is potentially vulnerable to an XSS attempt, but there is nowhere on this particular site where that becomes relevant, so it was ignored.

The `main.js` file includes a comment that reveals the version of a backend application being used on the server.

```
//make sure this js file is same as installed app on our server endpoint:
↪  /seeddms51x/seeddms-5.1.22/
```

None of the other JavaScript files reveal anything meaningful.

7.1.3 SeedDMS Enumeration

7.1.3.1 Exploit Search

In order to verify what sort of application SeedDMS was, an initial search online revealed that it is an open-source PHP-based document management system.

As the version was identified, a follow-up search for any existing vulnerabilities was done. This revealed a number of vulnerabilities for previous versions of the application, including a Remote Code Execution (or RCE) exploit.

While the application is not directly vulnerable to this exploit, it serves as a reference for the attack used later in this report.

7.1.3.2 Code Review

The application being open-source allows for code review to be done. The git repository was analysed for any potential weaknesses or areas of potential value and revealed a `conf` directory containing some important configuration files for the application - namely, the `settings.xml` file.

Upon navigating to the directory mentioned in `main.js`, we are met with a 404 error page.

`http://10.10.10.10/seeddms51x/seeddms-5.1.22/conf`

However, upon going one directory up, we receive a different error code and a 'Forbidden' message.

`http://10.10.10.10/seeddms51x/conf`

This reveals to us that there is likely an `.htaccess` file preventing us from accessing this resource.

Visiting the source code repository again, we can see an `.htaccess` file present in this directory. Upon inspection, it states that the `settings.xml` file in this directory should not be exposed to the outside world.

Navigating to this file, we are able to read the contents of it - revealing a significant amount of information regarding the configuration of this application. This includes (but is not limited to): - Database

Details (credentials, hostname, database name etc) - Supported file types for uploaded documents - Configured authentication and guest access permissions

`http://10.10.10./seeddms51x/conf/settings.xml`

This now gives us credentials to access the MySQL database that we discovered from our initial nmap scan.

7.2 MySQL Database Exploitation

Now that we have credentials for the database, we can connect to it and get a list of users and any other tables that might be of value.

Using the following query, we are able to enumerate a list of users, including one we haven't come across before:

```
USE seeddms; SELECT * FROM users;
```

This reveals a user called `saket` as well as a password for this user, which we will keep note of.

There is also an `admin` account, but appears to have an obfuscated password. The password seems to have the identifiers of the MD5 hashing algorithm, meaning we can try to replace the admin password with a new one of our own choosing.

We hash the password `admin` with MD5 and then update the table using the following query:

```
UPDATE tblUsers SET pwd = "21232f297a57a5a743894a0e4a801fc3" WHERE login = "admin"
```

We should now be able to log in to the web interface for SeedDMS with an administrator view.

7.3 SeedDMS File Upload

Now that we have access to the management interface for SeedDMS, we can upload a web shell to this as a document. Given we already know that SeedDMS is PHP-based, we can look to using a PHP-based web shell.

By navigating to the Documents section, we can upload a PHP file of our choosing. We can use an example payload from Pentest Monkey, which we customized to point to our attacking machine.

Once uploaded, we can access it via the following link:

`http://10.10.10.10/seeddms51x/data/1048576/<document_ID>/1.php`

Note: The `data/1048576` is a default directory that uploads are added to, and files are automatically renamed to `1.php`.

7.4 Reverse Shell

In order to utilize the reverse shell we uploaded, a new netcat listener is set up on our attacking machine using the following command:

```
rlwrap nc -lvnp 4444
```

Then, we can navigate to the page mentioned in the previous section, and we see a connection to our reverse shell.

We then stabilize the shell and do some additional enumeration of the system.

7.5 System Enumeration

The first thing we look at is the `/home` directory to see if any particular users stand out. We find that the `saket` user that was present in the MySQL database is also here.

7.6 Privilege Escalation

7.6.1 User

If we use the password we obtained previously, we can become the `saket` user with the following command:

```
su saket
```

The password is accepted, and we can now access any documents from this user.

7.6.2 Root

We can also check what permissions `saket` has for `sudo` access by running the following command:

```
sudo -S -l
```

As we can see, this user can run `sudo` for anything, so we can simply run the following command and we get `root` permissions.

```
sudo -i
```

8 Clean-up

As this was intended as a training exercise, no clean-up actions were necessary. However, in a real-world scenario, the following would be done:

- The `php-reverse-shell.php` file would be deleted from the file system.

9 Conclusion

Overall, this was a pretty fun exercise! I quite enjoyed doing the code review to identify potential vulnerabilities. Digging through the MySQL data and finding ways to bypass the MD5 hash was also fun and not something I've come across before.

I highly recommend checking out the box yourself over at: https://www.vulnhub.com/entry/hack-me-please-1,731/