

Hash auth

Description: Ok ok ok. They have learned there lesson. Instead of using confusing authentication functions there have switched to crypto functions. At least they think they have switched to crypto functions. Anyway - please help us again and hack this piece of code!

Level: Level 2

Type: Coding

Link: <https://ctf.securityvalley.org/dashboard>

So what we have is a `.py` file:

```
from hashlib import sha256

import sys

def validate_password(password):

    # be creative. it has something to do with SecurityValley ;-)

    if sha256(password.encode("utf-8")).hexdigest() ==
"f51f333ed26c41bedd99e1e483c0a15d2caeed7dc5a9ae02159f196799a74893":

        return True

    return False

def print_banner(payload):
```

```

print("that was great !!!")

print("run the following command to get the flag.")

print("curl -X POST http://ctf.securityvalley.org:7777/api/v1/validate -H 'Content-Type: application/json' -d
'{\"pass\": \"\"+payload+\"\"}'")

if __name__ == "__main__":

    print("let's do more python ;-)")

    password = input("please enter password: ")

    if validate_password(password):

        print_banner(password)

        sys.exit()

    print("wrong!")

```

We can see that their is a hash:

`f51f333ed26c41bedd99e1e483c0a15d2caeed7dc5a9ae02159f196799a74893`, but it gives us nothing with **hash cracker**.

They give us a hint: "be creative. it has something to do with SecurityValley ;-)",so we can try to make a `"wordlist"` file with some varient of "SecurityValley" like this:

```

securityValley
SecurityValley
securityvalley
SecuritlValley
securltyValley
Securtlvalley

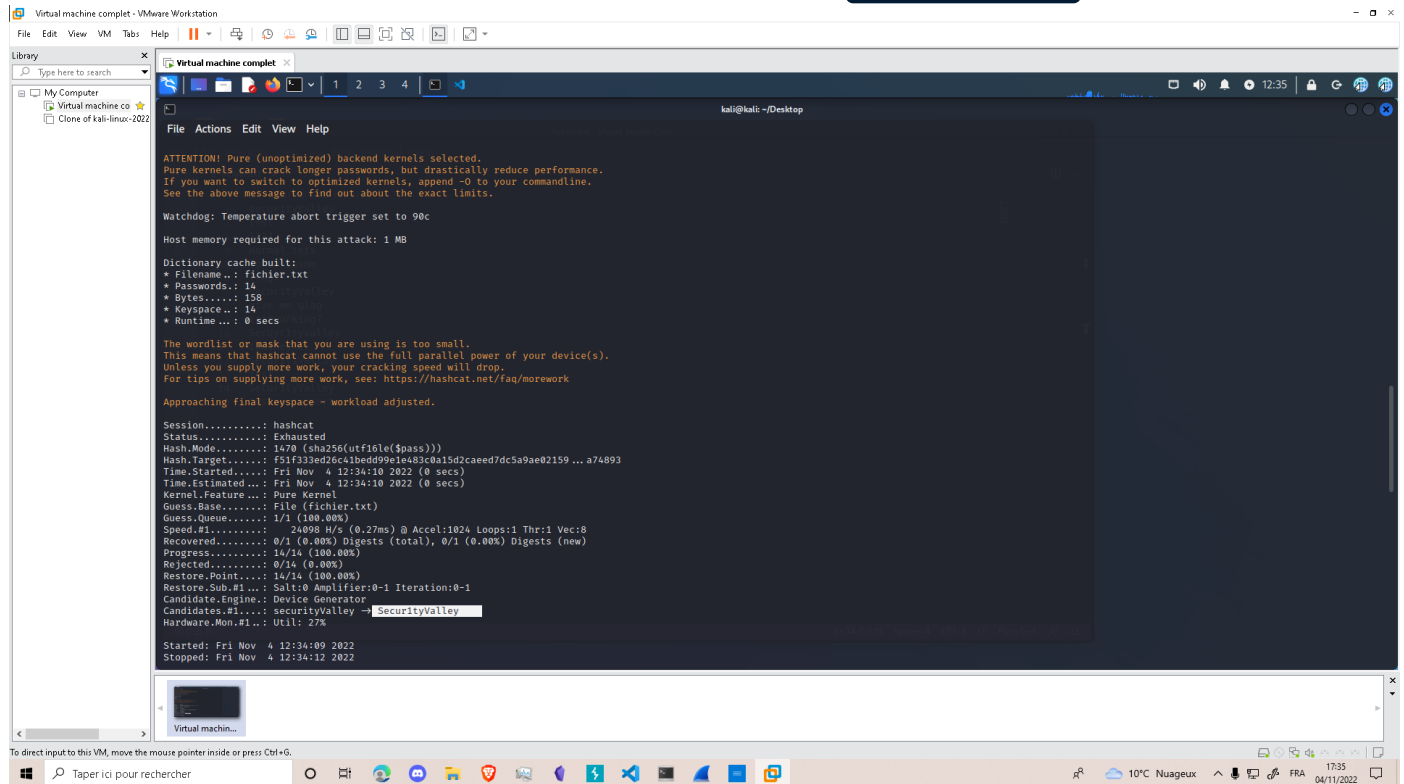
```

and use **Hashcat** to crack the hash and get the password: by using this command:

```
hashcat -a 0 -m 1470 f51f333ed26c41bedd99e1e483c0a15d2caeed7dc5a9ae02159f196799a74893
```

wordlist.txt -o --show

to crack the ashes with some word of our `wordlist.txt` file, and we can see that the first one to come is `SecurityValley`:



The screenshot shows a VMware Workstation window with a Kali Linux virtual machine. The terminal displays the output of a hashcat command. It starts with a warning about unoptimized backend kernels. Then, it shows the dictionary cache built from 'fichier.txt' with 14 passwords, 158 bytes, and 14 keyspaces. The wordlist is too small for full parallel power, so the workload is adjusted. The session details show the hash target as 'f51f33ed26c41bedd99e1e4830a15d2caeed7dc5a9ae02159...a74893'. The progress bar shows 0/1 digests recovered. The final output shows the candidate 'SecurityValley' as the correct password.

```
ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 1 MB

Dictionary cache built:
* Filename..: fichier.txt
* Passwords.: 14
* Bytes.....: 158
* Keyspace...: 14
* Runtime...: 0 secs

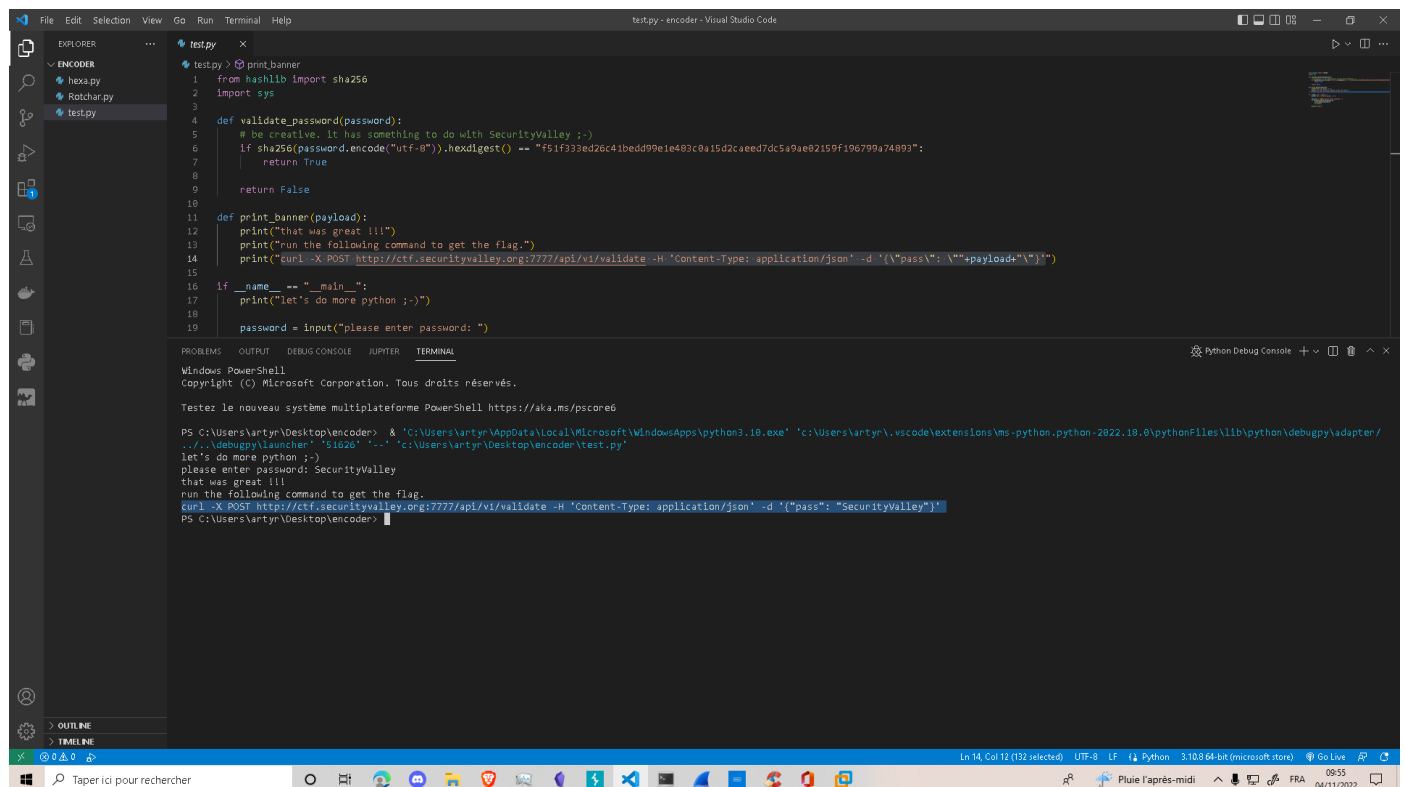
The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 1470 (sha256(utf16le($pass)))
Hash.Target.....: f51f33ed26c41bedd99e1e4830a15d2caeed7dc5a9ae02159...a74893
Time.Started.....: Fri Nov 4 12:34:10 2022 (0 secs)
Time.Estimated.....: Fri Nov 4 12:34:10 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (fichier.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 24098 H/s (0.27ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 14/14 (100.00%)
Rejected.....: 0/14 (0.00%)
Restore.Point.....: 14/14 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1.....: securityValley -> SecurityValley
Hardware.Mon.#1..: Util: 27%

Started: Fri Nov 4 12:34:09 2022
Stopped: Fri Nov 4 12:34:12 2022
```

And then when we enter `SecurityValley` in the python script, it gives us the command to run to get the flag:



The screenshot shows a Visual Studio Code window with a Python script named 'test.py' and its output in the terminal. The script defines a function 'validate_password' that checks if a password matches a specific SHA256 hash. It also defines a function 'print_banner' that prints a message and a command to get the flag. The script prompts the user to enter a password. The terminal output shows the user entering 'SecurityValley', which is accepted, and the command to get the flag is printed.

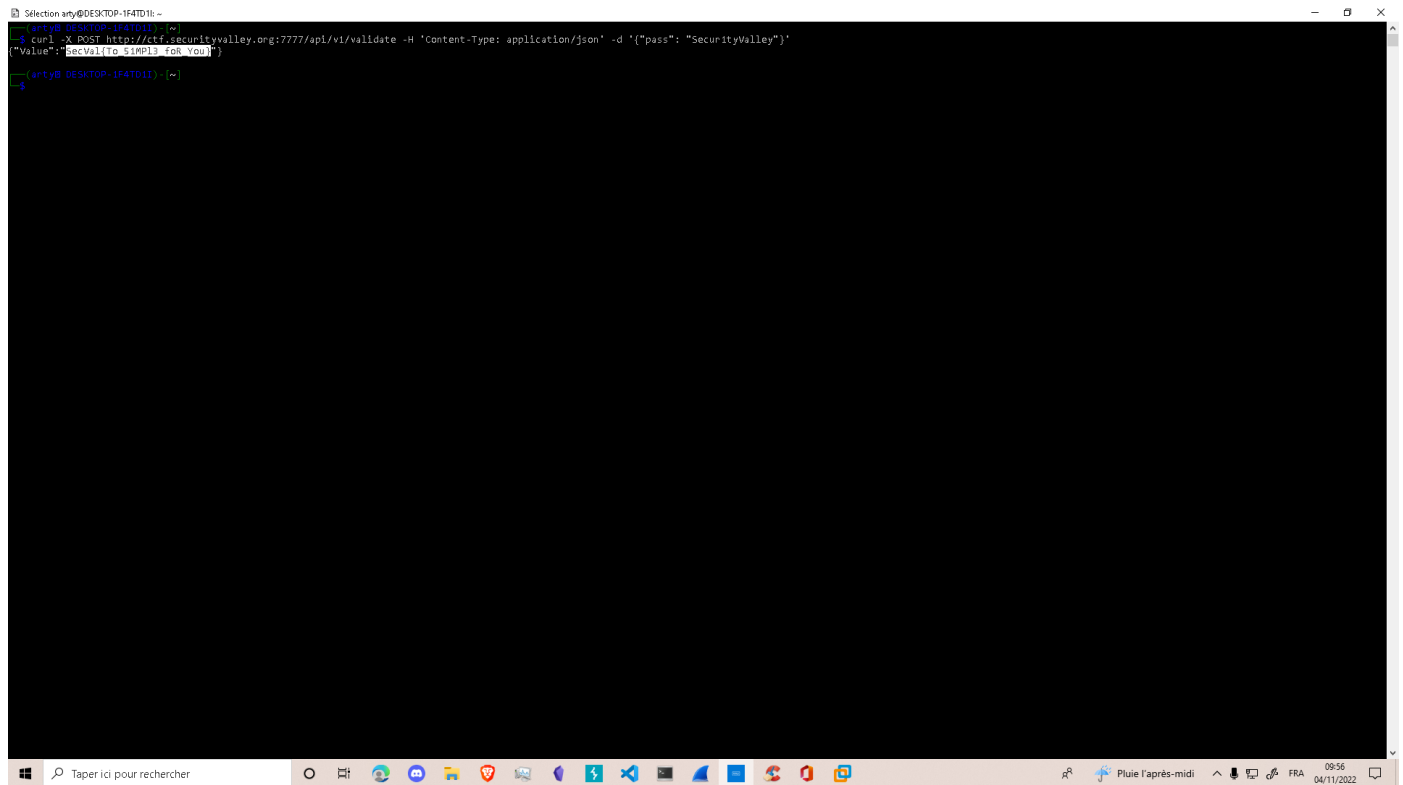
```
test.py > print_banner
1 from hashlib import sha256
2 import sys
3
4 def validate_password(password):
5     # be creative, it has something to do with SecurityValley :-))
6     if sha256(password.encode('utf-8')).hexdigest() == "f51f33ed26c41bedd99e1e4830a15d2caeed7dc5a9ae02159f190799a74893":
7         return True
8     return False
9
10 def print_banner(payload):
11     print("that was great !!!")
12     print("run the following command to get the flag.")
13     print("curl -X POST http://ctf.securityvalley.org:7777/api/vi/validate -H 'Content-Type: application/json' -d '{\"pass\": \"\"+payload}\"}")
14
15 if __name__ == "__main__":
16     print("let's do more python :-))")
17     password = input("please enter password: ")
18
19     if validate_password(password):
20         print_banner(password)
21         print("run the following command to get the flag.")
22         print("curl -X POST http://ctf.securityvalley.org:7777/api/vi/validate -H 'Content-Type: application/json' -d '{\"pass\": \"SecurityValley\"}")
23
24 if __name__ == "__main__":
25     print_banner("SecurityValley")
```

```
PS C:\Users\antyr\Desktop\encoder> & "C:\Users\antyr\AppData\Local\Microsoft\WindowsApps\python3.10.exe" "C:\Users\antyr\vscode\extensions\ms-python.python-2022.18.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher" "51626" "-c" "C:\Users\antyr\Desktop\encoder\test.py"
let's do more python :-))
please enter password: SecurityValley
that was great !!!
run the following command to get the flag.
curl -X POST http://ctf.securityvalley.org:7777/api/vi/validate -H 'Content-Type: application/json' -d '{"pass": "SecurityValley"}'
PS C:\Users\antyr\Desktop\encoder>
```

And when we run: ``curl -X POST`

`http://ctf.securityvalley.org:7777/api/v1/validate -H 'Content-Type: application/json' -d '{"pass": "Secur1tyValley"}'`

It gives us the flag !

A screenshot of a Windows terminal window. The title bar reads 'Selection artyb@DESKTOP-1F4TD1B: ~'. The terminal shows a command prompt where the user enters a curl command: `curl -X POST http://ctf.securityvalley.org:7777/api/v1/validate -H 'Content-Type: application/json' -d '{"pass": "Secur1tyValley"}'`. The output is a JSON object: `{\"Value\": \"SecVal{To_51MP13_for_You}\"}`. The terminal window has a dark background and is open on a Windows desktop with various taskbar icons visible at the bottom.

```
Selection artyb@DESKTOP-1F4TD1B: ~  
--(artyb@DESKTOP-1F4TD1B) --  
$ curl -X POST http://ctf.securityvalley.org:7777/api/v1/validate -H 'Content-Type: application/json' -d '{"pass": "Secur1tyValley"}'  
{\"Value\": \"SecVal{To_51MP13_for_You}\"}  
--(artyb@DESKTOP-1F4TD1B) --  
$
```

The flag is: **SecVal{To_51MP13_for_You}**